

# **Design Documentation**

for

# **Sydney Wildlife Rescue and Care App**

**Version 2.0 approved**

**Prepared by Jonathan Wong, Sam Turner, Samuel Hickman and  
Andrew Bryant**

**Macquarie University PACE Group 9**

**27 May 2016**

## Table of Contents

Revision History.....	2
1. <u>System Design Document</u> .....	3
1.1.System architecture.....	3
1.2.Storage/persistent data strategy.....	3
1.3.Trade-offs and choices.....	4
1.4.Concurrent processes.....	4
1.5.Package diagram.....	5
2. <u>User interface layouts</u> .....	6
3. <u>Program Navigation Diagram</u> .....	8
3.1.Screen flow diagram.....	8
3.2.Login activity diagram.....	9
3.3.New user registration.....	10
3.4.Animal First Aid activity diagram.....	11
3.5.Program navigation site map.....	12
4. <u>Data definitions</u> .....	13
5. <u>ER Diagram</u> .....	14

## Revision History

Name	Date	Reason For Changes	Version
Sam	27/5/16	Fixed errors, added ER diagram and data definitions	V2.0

# 1. System Design Document

## 1.1. System architecture

This system will be primarily web based, so the hardware required for the function of the site will be a hosting server that has php and SQL modules.

The client hardware will be suited to any web capable device including computers, laptops and smartphones.

The software architecture for running of the system will be any integrated file and web hosting software needed for the server such as apache and some type of database interface package such as PHPmyAdmin.

The choice for using PHP is that pages can be dynamically created on the server before being pushed to the client's device. This is a great feature as pages can be modified depending on the user, to show a different set of information for each access level.

This also provides great security as users must be logged in to see any content. The php is all server side and the client computer cannot access or modify it, providing a secure environment.

## 1.2. Storage/persistent data strategy

The bulk of the data for the system will be stored on a relational database in SQL. This will hold all of the information for the running of the system.

The advantages of using a relational database rather than a flat file system include:

- Data is only stored once
  - No multiple record changes needed
  - More efficient storage
  - Simple to delete or modify details.
  - All records in other tables having a link to that entry will show the change.
- Advanced queries can be performed
  - This is the ideal reason for our app since we will need to view animal profiles where a set of criteria match, such as who is looking after that animal, if that animal has a pending status etc.
- Better Security
  - Because the tables of the database are separated out, only the tables that need to be accessed by the application can be use. For example when a user attempts to login, the system will only need to query the table that contains the user login information. That application (especially if compromised) will only have access to that user's record in that table, and no other information can be extracted.
- Allows for scalability and caters for future requirements.
  - The database can be scaled easily as it is stored on a web server that can be configured to different sizes.
  - In the future if additional tables are needed for a new feature, the data can 'plug into' the existing tables via primary keys.

There will also be an external media server which will host the animal images uploaded by clients. References to these resources will be stored in the database as mentioned above.

### 1.3.Trade-offs and choices

The other option would be to use a flat file database, similar to an excel spreadsheet to contain all information needed for the functionality.

This would be an extremely large and messy file with complex or no relations and lots of data redundancy.

The method of keeping the media and informational data separate allows the servers to be modified to work the best based on the information they are containing. If the media were to be stored where the information was being stored, it would slow down response time dramatically. The use of a reference rather than the actual file solves this issue.

### 1.4.Any concurrent processes and how they will be handled if any exist

There are two systems currently being used by the client. These systems are BatchGeo, and WildApricot. BatchGeo is an application that overlays contact details addresses onto a google map, which can be used to visually see where contacts are in a geographical area.

This is useful when an animal is reported in an area to see which rescuer is closest.

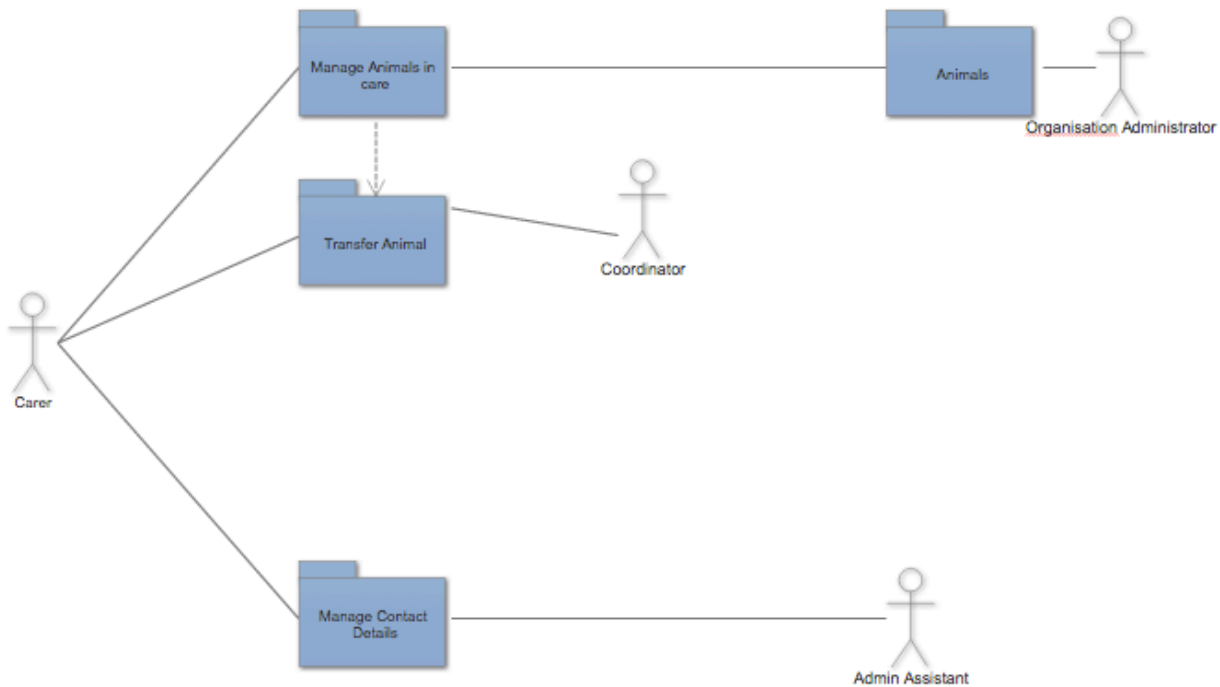
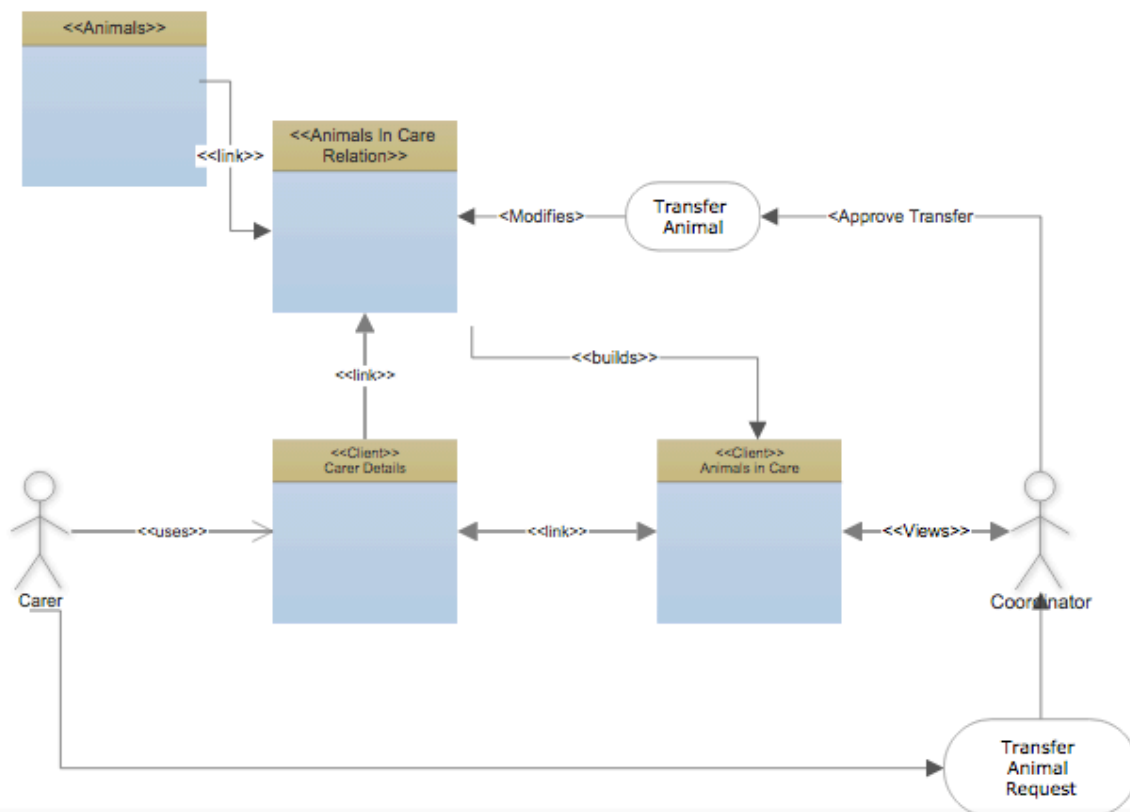
The other system WildApricot is a membership and directory application. The client currently uses it to keep track of it's members and keep contact details up to date.

We plan to investigate a way to connect WildApricot to our system to insure the contact details are kept synced with our system's contact details. This will allow the client to keep using the same methods to keep details up to date without having to do any extra entry.

There is currently an export feature from WildApricot which we could automate and import into our system when required.

However Ideally we can use a WildApricot API or LDAP service to use the existing user accounts in our system, so users only have to register in one place. This avoids having multiple logins for different systems for each user.

## 1.5. Package diagram showing the subsystems you will use

**Use Case Package Diagram: Animal Management System****UML Class Diagram: Animal Management System**

## 2. User interface layouts

The following screenshots are parts of the prototype that are functional.

### 2.1. Member login page

Sydney Wildlife PACE **Home** Profile Decision Tree Contact Profiles [Logoff](#)

[Search](#)  
[Transfer](#)

## Login

Email:

Password:

Currently logged out.

If you don't have a login, please [register](#)

[Forgot Password?](#)

Log On  
Register  
Nav7  
Nav8  
Nav9  
Nav10

**sydney** wildlife pace

### 2.2. Animal Profiles





A page showing all animals that have been recorded and their status

Sydney Wildlife PACE **Home** Profile Decision Tree Contact Profiles [Logoff](#)

## Animal Profiles

Currently in care of / has status of / all view

All profiles below are pulled from a database, and will be sortable based on query etc.  
We will also implement a list view etc.

 <p><b>grey-headed flying fox</b> bat baby, wing injury Status: resolved by another organisation/vet</p>	 <p><b>red belly black snake</b> reptile juvenile, unsuitable environment Status: relocated Animal ID: 2</p>	 <p><b>brushtail possum</b> marsupial baby, collision with motor vehicle Status: in care Animal ID: 3</p>	 <p><b>magpie</b> bird juvenile, fallen from nest</p>
---	---	---	--

## 2.3. Decision tree

Sydney Wildlife PACE

## PACE Example

Welcome

This is the Decision Tree ALPHA.


Do you know what is wrong with the animal?

I Do

I Do Not

Back

Restart

 This page built with Zingtree.

Footer Text

## 2.4. Decision tree results page

Sydney Wildlife PACE

## PACE Example


Medium Viability - Assessment

If you have reached this answer, the animal has the possibility of recovering, though it could be quite low. Further advice from a coordinator/expert carer as well as a vet is highly recommended.

Conditions such as your personal caring facilities, experience as well as accessibility to treatments and medications plays a large factor in this animals recovery.

Back

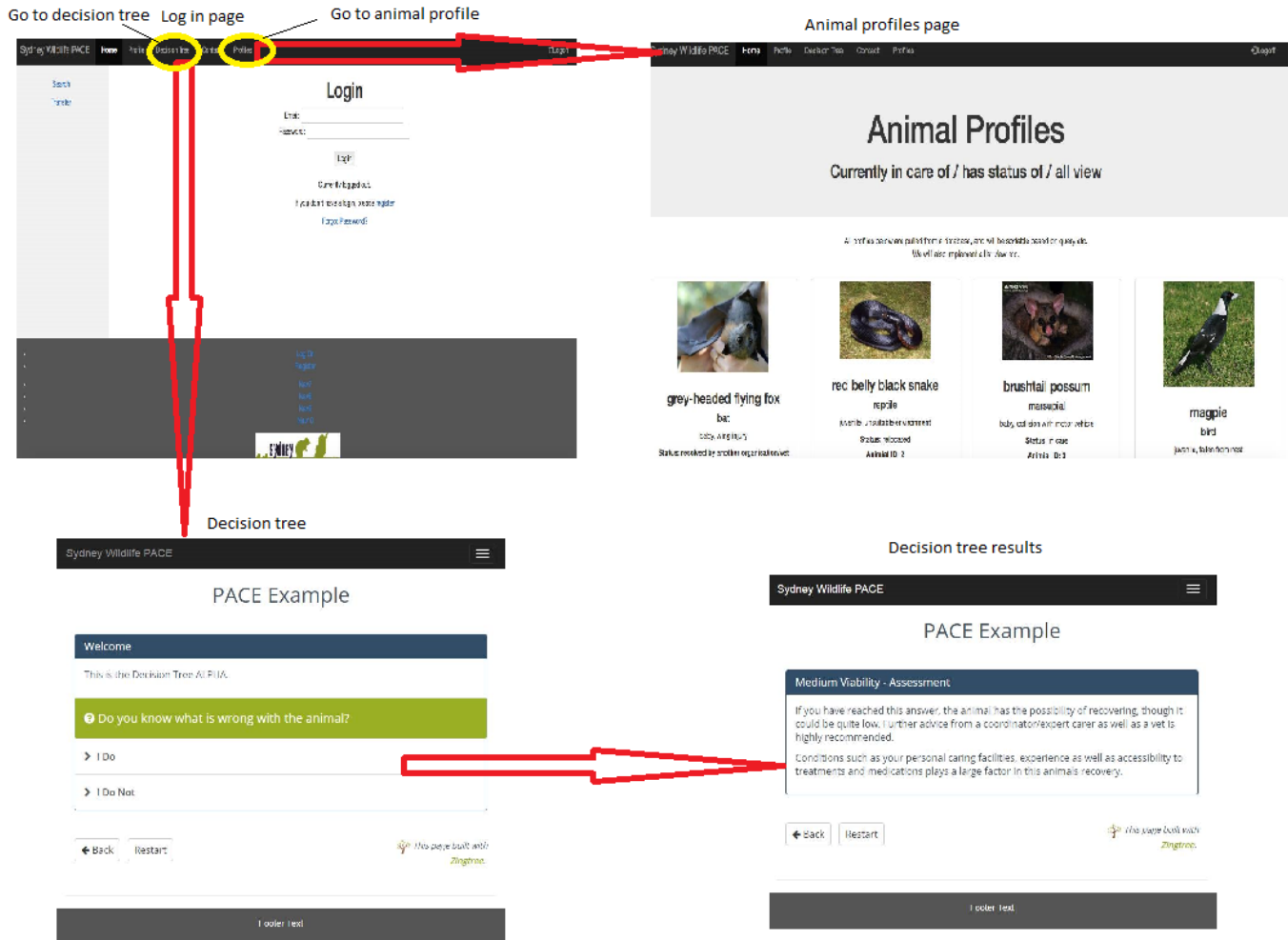
Restart

 This page built with Zingtree.

Footer Text

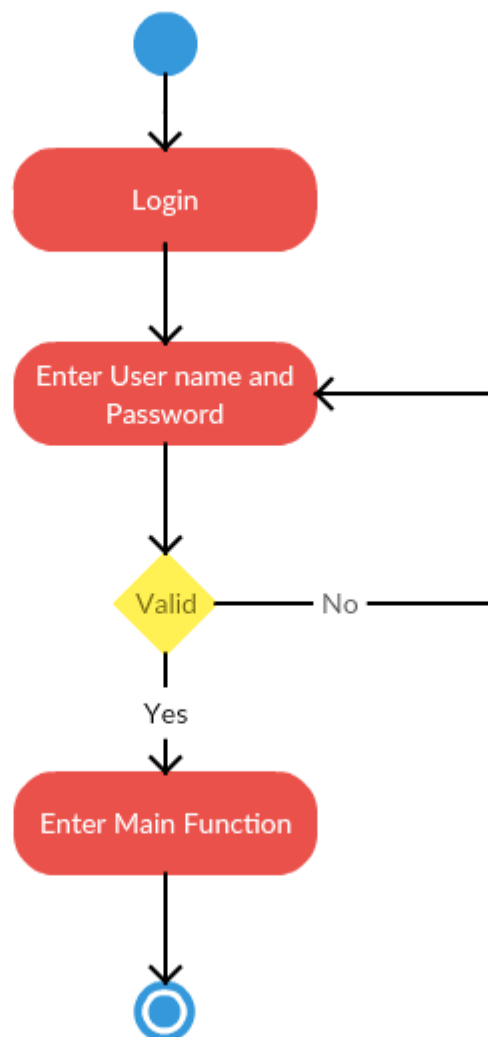
### 3. Program Navigation Diagram

#### 3.1. Screen flow diagram

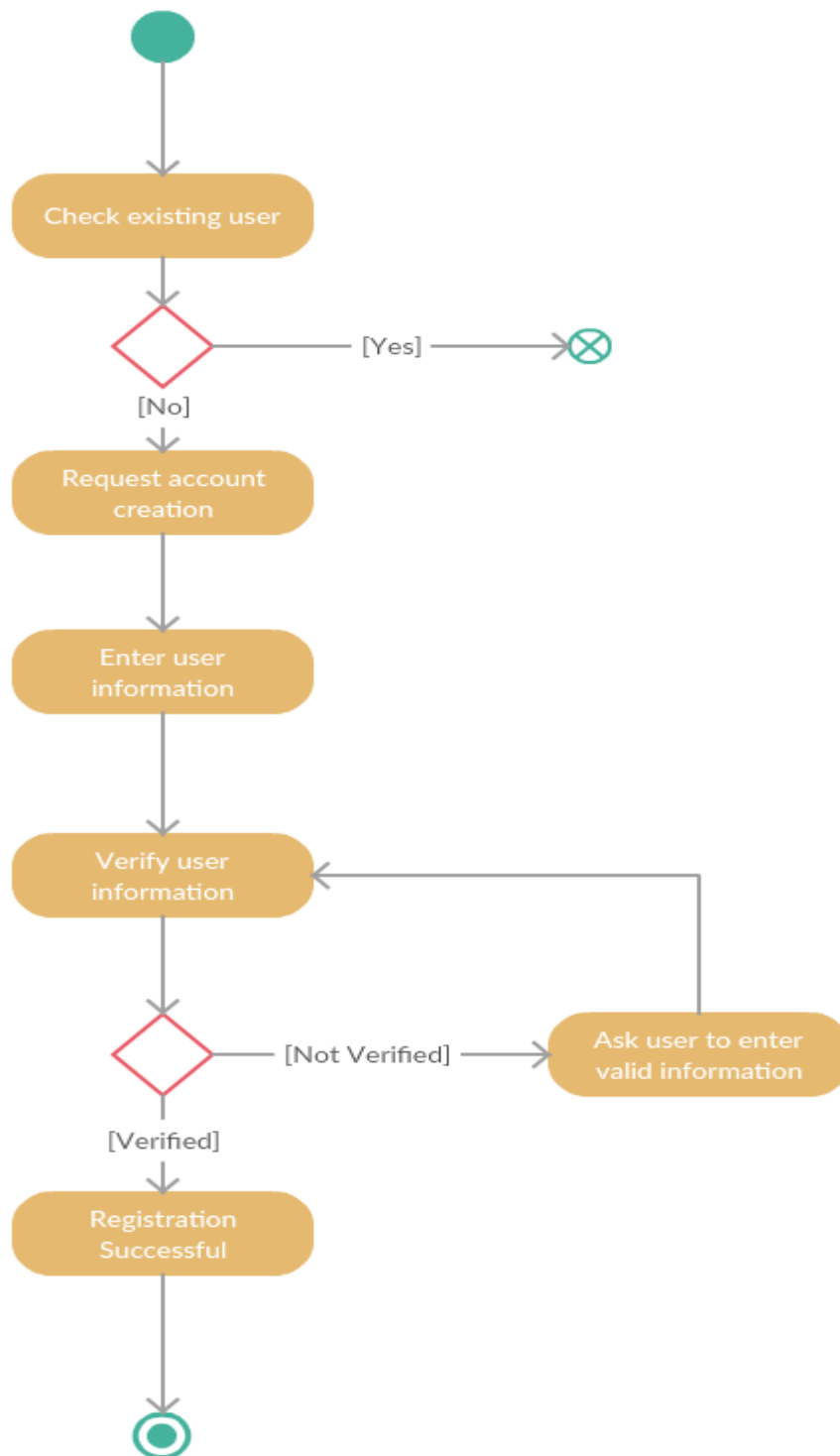




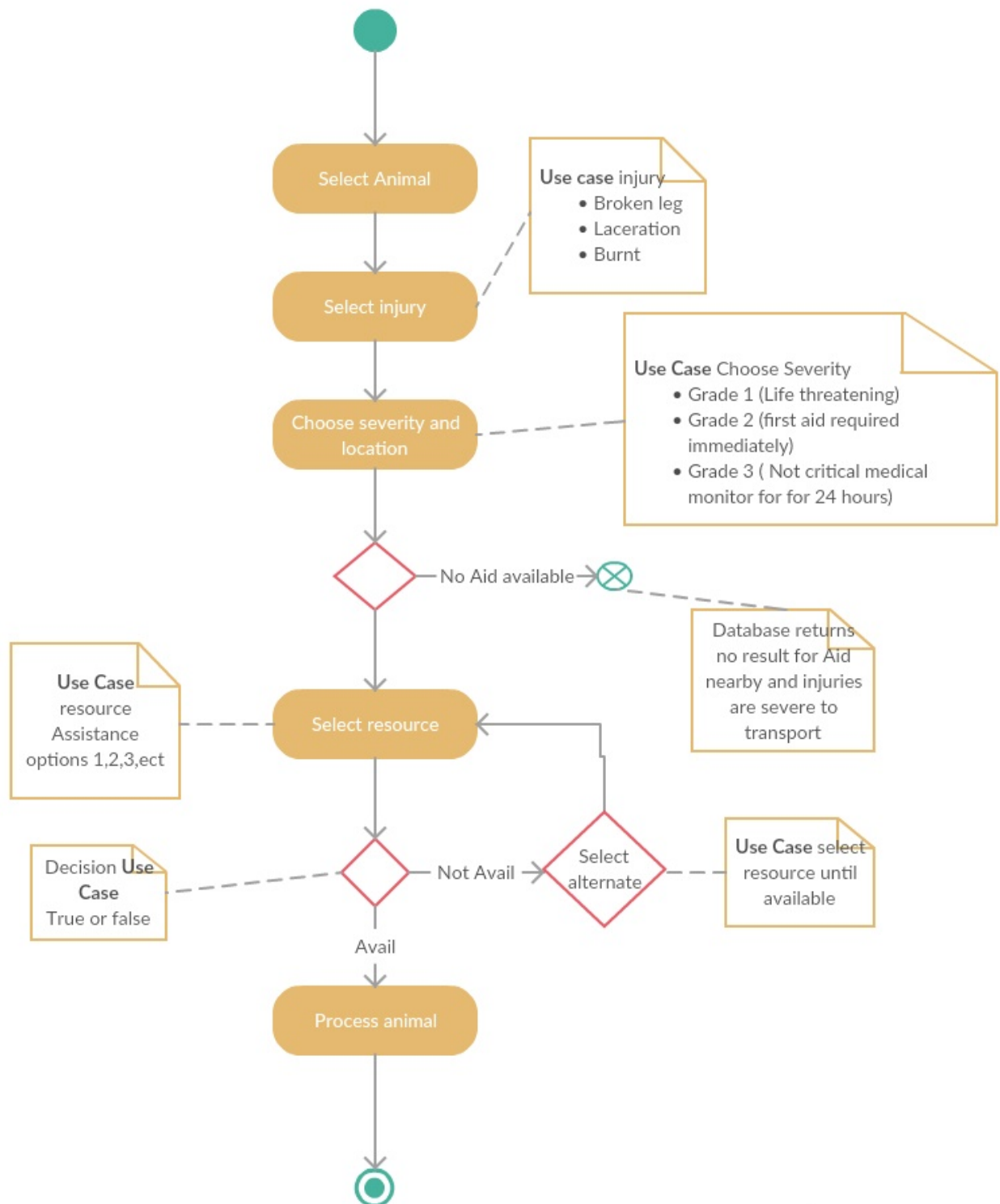
## 3.2.Login activity diagram



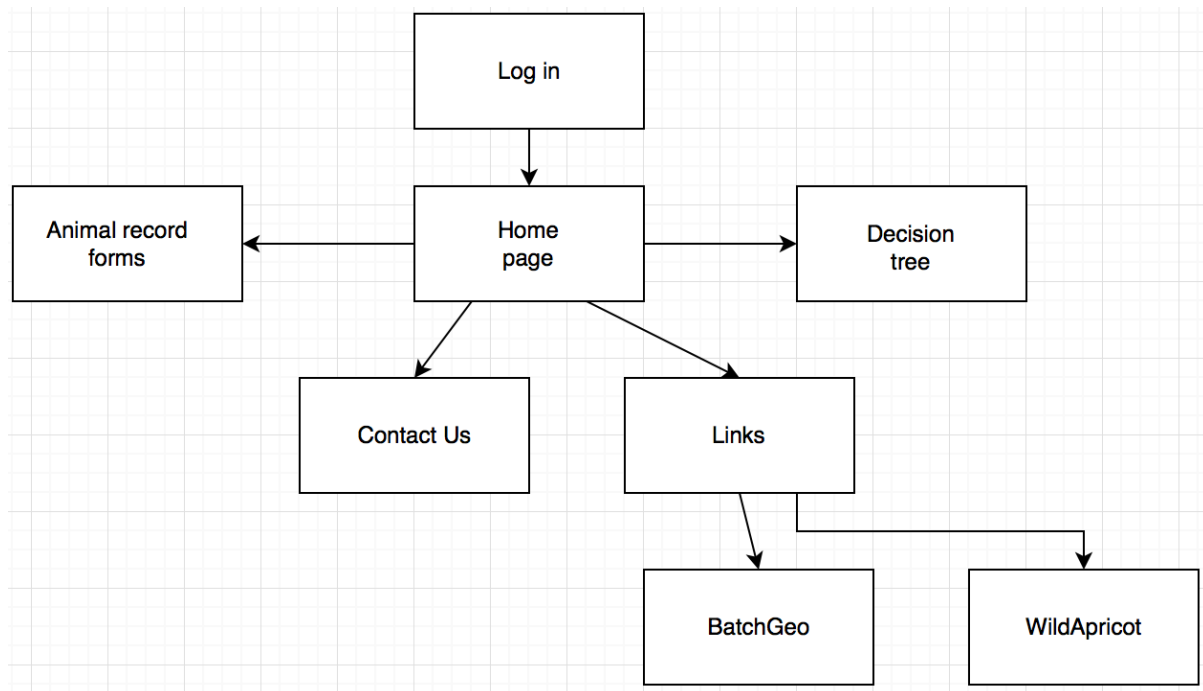
## 3.3.New user registration



## 3.4. Animal First Aid activity diagram



## 3.5. Program navigation site map



## 4. Data definitions

Member Table			
Attribute	Type	Length	Rules if applicable
memberID	CHAR	8	PK
firstName	VARCHAR	20	
lastName	VARCHAR	20	
email	VARCHAR	50	
contactNumber	NUMBER	12	
address	VARCHAR	50	
ubd	NUMBER	3	
availability	VARCHAR	20	
specialisation	VARCHAR	50	

Caller Table			
Attribute	Type	Length	Rules if applicable
callerID	CHAR	8	PK
firstName	VARCHAR	20	
lastName	VARCHAR	20	
contactNumber	NUMBER	12	
locationFound	VARCHAR	50	
date	DATE		
time	TIME		

Animal Record Form Table			
Attribute	Type	Length	Rules if applicable
recordID	CHAR	8	PK
animalID	CHAR	8	FK1
callerID	CHAR	8	FK2
memberID	CHAR	8	FK3
animalSpecies	VARCHAR	50	
date	DATE		
time	TIME		
animalInjuries	VARCHAR	255	
animalFate	VARCHAR	150	

Animal Table			
Attribute	Type	Length	Rules if applicable
animalID	CHAR	8	PK
animalType	VARCHAR	15	
animalSpecies	VARCHAR	50	
animalDescription	VARCHAR	255	
animalStatus	VARCHAR	150	
animalPictureFile	VARCHAR	50	

## 5. ER Diagram

