

## Final Year Project Report

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                           | <b>3</b>  |
| 1.1      | Project Overview . . . . .                    | 3         |
| 1.2      | Project Objectives . . . . .                  | 3         |
| 1.3      | Project Challenges . . . . .                  | 3         |
| 1.3.1    | Dataset Acquisition . . . . .                 | 3         |
| 1.3.2    | Time Management . . . . .                     | 4         |
| 1.3.3    | Technologies . . . . .                        | 5         |
| 1.4      | Structure of Document . . . . .               | 5         |
| <b>2</b> | <b>Research</b>                               | <b>5</b>  |
| 2.1      | Introduction . . . . .                        | 5         |
| 2.2      | Background Research . . . . .                 | 6         |
| 2.2.1    | Object Recognition . . . . .                  | 6         |
| 2.2.2    | Edge Detection . . . . .                      | 6         |
| 2.2.3    | Visual Recognition . . . . .                  | 7         |
| 2.3      | Alternative Existing Solutions . . . . .      | 8         |
| 2.3.1    | Subject Detection . . . . .                   | 8         |
| 2.3.2    | Object Classification . . . . .               | 9         |
| 2.4      | Technologies Researched . . . . .             | 10        |
| 2.4.1    | OpenCV . . . . .                              | 10        |
| 2.4.2    | TensorFlow . . . . .                          | 11        |
| 2.4.3    | Amazon Web Services . . . . .                 | 11        |
| 2.4.4    | Digital Ocean . . . . .                       | 12        |
| 2.4.5    | Flask . . . . .                               | 13        |
| 2.4.6    | Django . . . . .                              | 13        |
| 2.4.7    | MySQL . . . . .                               | 14        |
| 2.4.8    | SQLite . . . . .                              | 15        |
| 2.5      | Resultant Findings and Requirements . . . . . | 15        |
| 2.5.1    | Image Processing . . . . .                    | 15        |
| <b>3</b> | <b>Design</b>                                 | <b>16</b> |
| 3.1      | Design Methodology . . . . .                  | 16        |
| 3.1.1    | Feature Driven Development . . . . .          | 16        |
| 3.1.2    | Feature List . . . . .                        | 17        |
| 3.2      | Use Case Diagram . . . . .                    | 19        |
| 3.3      | User Interface Design . . . . .               | 20        |

|          |  |           |
|----------|--|-----------|
| 3.3.1    | Screen Wireframes . . . . .              | 20        |
| <b>4</b> | <b>Architecture and Development</b>      | <b>24</b> |
| 4.1      | Technical Architecture . . . . .         | 24        |
| 4.2      | Development . . . . .                    | 24        |
| <b>5</b> | <b>Testing and Results</b>               | <b>24</b> |
| 5.1      | Test Plan . . . . .                      | 24        |
| <b>6</b> | <b>Project Plan</b>                      | <b>25</b> |
| 6.1      | Original Plan and Changes . . . . .      | 25        |
| 6.2      | Key Differences . . . . .                | 25        |
| <b>7</b> | <b>Conclusion</b>                        | <b>25</b> |
| 7.1      | Key Learning Obtained/Findings . . . . . | 25        |
| 7.2      | Future Work . . . . .                    | 25        |
| 7.3      | Closing Statements . . . . .             | 26        |

# 1 Introduction

## 1.1 Project Overview

TODO: All Overview

## 1.2 Project Objectives

TODO: ADD mention of app The objective of this project is to make a full stack web application with features that analyse and edit user photos. This requires a number of smaller objectives to be completed which are listed below:

- Develop an algorithm for separating the subject of a photo from the background using image processing.
- Source a suitable dataset that contains a large number of wildlife images separated into different species
- Build and train a machine learning model that is capable of classifying the subject of a photo into a specific animal category.
- Create a website that allows users to upload photos for classification and background removal. A user feedback form will be made to enable collection of user evaluations and comments.

## 1.3 Project Challenges

This is a large-scale project with challenges and issues that were encountered throughout. It was important to identify risks at the beginning, so it would be possible to prepare and possibly avoid some problems entirely. Risks that are not accounted for can cost significant time which is the most valuable resource for this project. This section covers the identified risks and challenges. These include sourcing a dataset, time management and technology experience.

### 1.3.1 Dataset Acquisition

A large challenge is finding a suitable dataset. The classification of input images will solely rely on a model trained with a specific dataset as machine learning is the only technique used in the classification process. It's valuable

to compare the results from different splits from several datasets to observe how they influence the model. The aim is to have high accuracy classification and even if the training process is well thought out the training data is where it learns available features and their relationship with the categories.

Finding a relevant dataset is essential for the classification process. This meant a lot of time was invested in sourcing a dataset and a number of promising ones were found early on. In addition to these a number of viable methods have been thought of for sourcing suitable datasets.

A method that technically would work is scraping google image search. This would obviously be a very poor-quality dataset due to the amount of unsuitable or irrelevant images returned. To have any hope of being making the dataset viable all returned images would need to be extensively reviewed and filtered. While this could give a good resulting dataset, it is far too time intensive and therefore not feasible.

Datasets can often be sourced from educational institutions or from already completed studies. This ensures high quality images and possibly a large degree of relevancy if it is from a similar project. The last option considered for acquiring a dataset is the google dataset search tool which, despite still being in beta, looks to be a powerful tool and will likely be useful.

### **1.3.2 Time Management**

This project requires a large amount of work on researching, designing, implementing, evaluating and testing. Each of these stages requires a time investment and all must be completed within the timeframe of 8 months which is enough as long as there is preparation so that it's allocated well. This means planning out when to start on each section and the extent of time it is to be worked on. If targets are missed catching up quickly will be essential to avoid being overwhelmed when approaching the deadline.

The most important thing is to invest enough time into the project. Having time allocated throughout for project work while leaving time for all other work such as college modules. Following a plan consistently will ensure the project is kept on track. When encountering obstacles that prevent further development of the application, they need to be resolved swiftly. The best

approach in these circumstances is to seek help. Several lecturers have said they are willing to give advice if needed. **TODO: REWORD AND BE MORE SPECIFIC** - Also, I can ask for help from my project supervisor for certain areas.

### 1.3.3 Technologies

**TODO: REWORD** - There are technologies that I do not have previous experience with that are planned to be used in this project. Researching technologies is useful but it is an entirely different matter to learn them/ It will take time to become competent with each technology and put them into practice. There is also the possibility that any of them could have limited functionality in the desired area, be incompatible with other required technologies or too difficult to work with to justify the time put in.

These problems can only be avoided by researching thoroughly all these aspects before choosing a technology. If a problem is still encountered despite best efforts, researching should have brought up alternative technologies that can then be considered as a replacement.

## 1.4 Structure of Document

**TODO: All Structure**

# 2 Research

## 2.1 Introduction

This chapter deals with key areas of research that are incorporated in this project

**TODO: FIX:** Object recognition also referred to as object classification is mainly done using machine learning nowadays. This meant exploring a number of well developed methods that are available for object recognition. I decided to explore the general topic and see what methods were available. I also checked to see if any of the datasets they used were publicly available.

## 2.2 Background Research

### 2.2.1 Object Recognition

Baxter et al. [1] attempted object recognition based upon segmenting an image into regions and attempting to match supplied keywords to different region types. It used a dataset that was not named or made publicly available. The approach they proposed was invariant with regard to feature type but had the issue that unwanted bias tended to be present as the dataset provided had a lack of variety.

Belongie et al. used shape matching and shape contexts to recognise objects/shapes. [2] They used a variety of datasets. The result of the study demonstrated that their simple approach that relied on estimation of shape similarity and shape context provides great shape recognition with invariance to image transformations.

Barla et al. [3] used what is known as a Hausdorff Kernel for 3D Object Acquisition and Detection. Their approach uses a kernel that examines one object type at a time. It successfully identifies the smallest sphere in the feature space using the training data and this sphere can be used as a decision surface to classify new examples. It tended to have an impressive success rate for detecting 3D models.

Lowe [4] focused on “Object Recognition from Local Scale-Invariant Features”. Although he outlined an approach for finding a Scale Invariant Feature Transformation (SIFT) that can then be used for identifying additional features which then allows for robust recognition even with partial occlusion in cluttered images Lowe did not report on what data was used for the experiments or provide detailed experimental results.

### 2.2.2 Edge Detection

TODO: FIX: My initial idea for isolating the subject was to first focus on finding the edges in a photo. Edge detection typically works by identifying points at which image brightness changes drastically and organised into a set of line segments which are labelled edges. There are two main abstract methods, search based(5) and zero-crossing based.

There are two main abstract methods, search based(5) and zero-crossing based.

Search based calculates a scale for edge strength and highlights accordingly. It seems to be more a proof of concept rather than a well-developed functional process for identifying edges. There are almost no examples of it being used to develop an algorithm or approach for recording defined edges.

Zero-crossing is more popular with multiple well-known edge detection operators using this approach. Two important ones to examine are the Sobel Edge Detector[5] and the Canny Edge Detector.[6]

Sobel Edge Detection uses kernels for estimating the gradient along the x-axis and another for the y-axis and these can be used in a number of ways to approximate the magnitude of the gradient. It can even use simplified calculations to reduce accuracy but greatly increase processing speed.

Sobel is a straightforward method for detecting edges along with their orientation and can be optimised (by sacrificing a degree of accuracy) to give almost instant results. However, even without speed optimisation it has accuracy issues and is thrown off by noise in an image.[7]

Canny Edge Detection uses differentiation to mathematically calculate edges, requires pre-processing, typically smoothing using a Gaussian filter. Gradient magnitude is calculated in a similar fashion as was used in Sobel. Finally, thresholding is used along these edges.

Canny has in built functionality for removing noise. It has good localisation and has an impressively high signal to noise ratio. Getting the algorithm to run in real time is extremely difficult as the processing of this approach is extremely time consuming.

### **2.2.3 Visual Recognition**

Visual recognition research led to a paper about a yearly challenge that invited competitors to try get the best possible accuracy for a specific dataset with any approach they wanted.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[8] has been run annually since 2010. It is a benchmark with millions of images broken down into hundreds of categories that aims to provide the necessary base for object category classification and recognition. The premise was adopted from the Pascal Visual Object Classes Challenge[9].



The group that got the highest accuracy used an object detection system that has a focus on discouraging the algorithm from ‘bad behaviour’ such as recognising duplicate instances in the same photo, false positives and failing to recognise objects. It provides possible detections along with confidence scores.

They found for categorisation that from their test with humans that even the best automated system fell below the human’s average by 1.7% in terms of success rate. However, an important note is that this was for humans that had undergone training, so it is likely that the top automated system would be fantastic for users as it would be more accurate than your typical person.

## **2.3 Alternative Existing Solutions**

### **2.3.1 Subject Detection**

Subject detection can be completed by identifying the background or the foreground and then it is simply removing the background from the original image.

It uses image segmentation which groups areas that share certain characteristics. A simple approach to this is thresholding where the image is split into two parts that are in a specific range of RGB values and those that are not. Two other methods for this are the GrabCut[10] algorithm and the Watershed[11] algorithm.

Thresholding is the most basic subject detection method. It is difficult to define a generic approach, it might be possible through thorough analysis of the image statistics but still it would probably need further processing.

Thresholding gives a well-defined separation between foreground and background but even with great image analysis will not be dynamic enough. The solution provided would most likely have a high failure rate if additional features for pre-processing the image were not used.

The GrabCut algorithm on its own doesn’t suit the project well but could be useful for testing purposes. A user specifies a rectangular region that contains all the foreground. Everything outside of this is marked as definitely background and is used to identify everything else that is background. A

result is shown, and the user can make small marks on regions that are incorrectly labelled and the algorithm does another run-through taking this into account.

#### TODO:Add grabcut diagram

GrabCut is extremely accurate given the right prompting but requiring user input slows down the experience, makes it less user friendly and adds a lot of difficulty in mass testing. Could potentially be used for giving feedback on a failed result for a different run of an algorithm.

The Watershed algorithm is an automatic process. It requires some pre-processing, so it begins by thresholding the image and denoting definite-foreground and definite-background. A distance transformation is applied to help split touching objects and the separated object areas are saved as an array. OpenCV then has an inbuilt watershed method that can be called.

Watershed is efficient, gives great results and has lots of fine tuning available for the pre-processing but can falter if there is little differentiation between foreground and background.

### 2.3.2 Object Classification

A research paper called Cats and Dogs[12] is a project based around identifying whether a picture contains a cat or a dog and then what breed the animal is. It uses a lot of techniques that will be beneficial to look at as they could pertain to my project. The project finds the face using a deformable part model as it can't be too strict and try focus on rigid shapes due to the countless factors that can distort an animals face as even a slight change in viewpoint would invalidate that kind of approach. There are then two different classification approaches that are compared against each other. Firstly, a two-tier approach where the animal is decided to be either a cat or a dog and then assigned a breed. Secondly, a simpler approach where the program only attempts to match a breed to the image without classifying it to a type of animal.

The separation between foreground and background is made by using the segmentation technique known as GrabCut, which usually relies on user input to get a rough bounding box around the subject of the photo but is

made automatic in this case by getting the estimate of the bounding box programmatically.

The goal of this project was making a system with the ability to accurately identify the breed and family of two specific animal groups. There were three different approaches, the last of which is a combination of the first two. Shape only gave an accuracy of 94.21%, appearance only was 87.78% and combining them resulted in an improvement over both with an accuracy of 95.37%. As expected, the more information provided about the image the better the result, this can also be noted for the appearance-only approach where accuracy improved as more localisation was done to the pet body.

## **2.4 Technologies Researched**

### **2.4.1 OpenCV**

OpenCV[13] was originally developed by the Intel Research Labs with the hope to improve the ease of access to hopefully advance vision research by providing open, well optimised code for computer vision basics. This would save a lot of time on the necessary setup for tackling image processing. They wanted their code base to be used as a standard so that knowledge and techniques would be more transferable. The OpenCV library was made available on as many platforms as possible so that it would be extremely portable. The license was made to be non-restrictive so that it could be used for more than just public open projects.

There are many benefits of using OpenCV, as mentioned above it is quite portable so it's easy to incorporate it into different code bases. It is free to use which is always a positive and this paired with its great accessibility results in high usage in the industry and a lot of experienced users writing.

It has low ram usage and is very fast for its C/C++ implementations. There is also a python library that while slower, can be preferable due to having a simpler code base and more readable scripts. OpenCV-Python works with Numpy which is an extremely efficient library that has a wide range of mathematical computations. Numpy is used by a large number of libraries so its implementation can improve compatibility with other technologies.

### 2.4.2 TensorFlow

TensorFlow was developed by the Google Brain Team and is heavily used in the realm of machine learning. It has in built functionality for running computation on CPUs, multiple GPUs and TPUs and on devices such as mobile, desktop and server clusters. This allows users to minimise deep learning training time by sharing the load.

It works with Python 3 and has some image processing abilities, however most of these are designed for formatting images to be used as training data. It has many useful APIs, for example keras. Keras is well suited for fast prototyping as it is designed based on user interaction rather than to fully describe machine learning logic. This user-friendly design paired with its modular design make it a great tool for making an implementation quickly that has the ability to be expanded upon later.

It is well suited for deep learning and is perfect for neural networks with lots of layers and strange topology. TensorFlow has great tools for visualisation that assist in debugging and optimising applications. It is a widely used and freely accessible framework that is extremely well documented with hundreds of tutorials.

### 2.4.3 Amazon Web Services

Amazon Web Services (AWS) is a cloud services platform that offers functionality such as content delivery, computing power and database storage[14]. AWS had a bare-bones launch in 2002. Its focus is on providing businesses solutions that increase flexibility, scalability and reliability.

The amount of technologies it is able to support is astounding. It has ways of incorporating almost every technology researched in this section apart from rival cloud services of course. It is easy to use and deploying basic applications to the cloud can be achieved quickly by following a range of their well laid out tutorials. The scalability solutions mean there are no limits on capacity and the current capacity can be upgraded as needed. Other services force businesses to select specific tiers meaning money is wasted if future use is over estimated, while AWS lets businesses pay for the storage used.

Flexibility means you can get access to needed services quickly. Business in need of technical expansion typically have to plan acquisition and setup of

hardware that could take days before forward progress can be made. AWS allows access to these services almost immediately and with tools for elastic load balancing business can match their exact demand.

AWS has good security that doesn't require any extra effort on businesses end to maintain. All data is kept on their private secure servers and protection is size invariant, so vulnerabilities don't arise for even the largest databases.

A big disadvantage of the platform is their technical support fees. The service is meant to be almost completely automated for the user so it should be rare that support is needed but for moderately sized businesses, monthly support fees can be hundreds or even thousands.

**TODO: ADD Pythonanywhere**

#### **2.4.4 Digital Ocean**

DigitalOcean define themselves as “a values-driven organization” that believe they should, “Start by defining the problem. Have the courage to approach a problem from a different perspective. Plan ahead, but act decisively. Consider that the simplest approach is often the best.” [15] They have a large focus on developer experience and this influences their user interface design. This can be seen in the control panel for their service, where a lot of time consuming tasks are made simpler with fast compute server creation, reliable object storage and management tools for infrastructure.

The DigitalOcean cloud is SSD-only. This is rare in the cloud services market as it's cheaper to use slower mechanical hard drives in the creation of a cloud platform. The performance benefits of the system are available to all users allowing server launch in less than a minute. This paired with the control panel makes server configuration quick and easy.

The quality of the hosting is incredible and has a low barrier of entry as it is less than €10 a month. This means it is a tool that is feasible for both a lone developer or a giant business. Learning to use digital oceans service is easy as there are many tutorials available as well as a well-developed Q & A Section.

DigitalOcean have many Linux distributions available but not even a single kind of windows distribution is available which can be limiting depending

on developers experience and preferences. Also they don't use centralised storage, meaning backups need to be handled by the user if the data loss risk is to be mitigated.

#### **2.4.5 Flask**

Surprisingly, Flask was created as an April fools joke[16] and had its initial release in 2011. Despite this odd start, it has grown to be one of the most popular web frameworks for non-enterprise developers. This is most likely thanks to its versatility and lack of boilerplate meaning it is very pythonic and not too abstract.

Flask is a microframework that supports hundreds of extensions so the features that can be implemented suit the majority of users needs. This means you're not locked down to a specific database or templating engine, it gives developers a lot of freedom.

It is extremely easy to get started with flask, an application can be made in less than 10 lines. These basic apps can then be expanded upon to improve understanding and start to develop into useful web apps. This is something to be valued as some web frameworks can take many hours of learning to even get a proof of concept app running reliably. Flask is a great tool for any developers that enjoy the hands-on approach for learning a new technology.

A downside to the versatility means developers can find themselves in uncharted territory if they are using a rare combination of extensions. Even with an active community there is going to be many blind spots in the documentation for a framework such as Flask.

#### **2.4.6 Django**

"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development"[17] They also make the important note that the framework is free and open source.

Django is designed to speed up the creation of web applications. It is referred to as a batteries included framework meaning that developers have what the tools they need to power their web application out of the box. Even though

many of the features may go unused they are wide ranging enough that most elements of a project can be created just using the default packages. This means after successive applications have been created with Django a developer can quickly deploy their arsenal of tools on a new project and reach completion faster.

This also highlights how Django developers can be transferred far more seamlessly than other framework developers. The tools being ubiquitous among Django projects ensures that developers can jump right into a project at any point. Also, with it being a scalable framework that means there are many approaches for handling larger projects such as running separate servers for each element of the application or load balancing.

The main negative point for Django is how it can be quite obtuse for smaller projects. It can be annoying having such a large project structure for a single page web personal website. Of course, this doesn't make Django any less effective, just a bit overkill when it comes to simple applications.

#### **2.4.7 MySQL**

“MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.”[18] It is a high performance database management system that is compatible with a lot of different setups.

It is quite proficient at handling multi-user interaction unlike the previous technology. It is a fast and stable server that is well suited for programs that handle large numbers of concurrent additions and edits such as order or booking systems.

It is incorporated in the LAMP stack used in many web-based applications. LAMP (Linux, Apache, MySQL, PHP/Perl/Python) is widely used throughout the industry and is one of the leading open source web platforms.

The only disadvantage is the requirement of a running server for access, however this is the case for all but a few database implementations.

### 2.4.8 SQLite

SQLite is one of the most basic database implementations possible. It is entirely self-contained, doesn't require a server or even any configuration. It is different to the SQL databases that require a server process as it reads and writes directly to disk.

The main advantages and disadvantages for SQLite are directly related to scale. For a simple database with only a few tables it is perfect for the task, easy setup, simple operations and no requirement to run a server process. For a large database accessible by many users simultaneously there will be a large drop in speed. This is due to the database being locked during access meaning users need to queue for access. That means there is a point at which an SQLite database base will become unusable with no way of improving scalability.

## 2.5 Resultant Findings and Requirements

### 2.5.1 Image Processing

The above research covered the image processing library OpenCV. Its Python library is well documented, and has been shown to be quite intuitive.

**TODO: REWORD section below, too much first person language** Since starting this project, I have had a lot of practice with OpenCV-Python and become quite proficient.

The main downside to OpenCV-Python is its lower speed compared to the C++ version. The use case for image processing in this project is to remove the background from user submitted photos before providing the isolated image back to the user. From my experience, even heavily processing a single image rarely takes more than a second or two. I decided to pick OpenCV-Python as I believe my familiarity with it outweighs the speed benefit of the C++ for the required use case.

**TODO: All Resultant**



## 3 Design

### 3.1 Design Methodology

#### 3.1.1 Feature Driven Development

Feature driven development (FDD) is defined as an iterative and incremental software development process. It is a variation of agile methodology that has a big focus on end user experience. It has five main iterative activities that repeat during the development process.

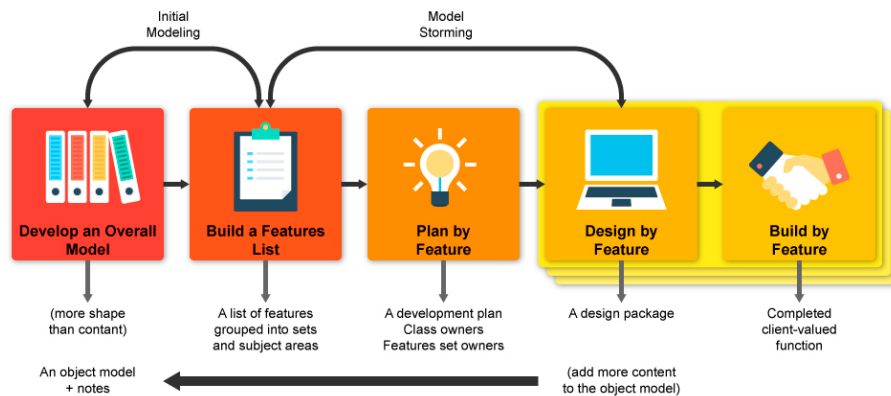


Figure 1: The steps of feature driven development

**Develop an Overall Model:** First of all there is a high level overview of the system to understand components priority and how they will fit together. Next, models are created for different domain areas and are reviewed. These models will slowly merge into a single comprehensive model.

**Build a Features List:** Domain areas are broken down into their possible interactions and this allows features to be identified. Each feature can be expressed in the format "action, result, object", for example 'Filter images in gallery'. It is important that no feature takes more than two weeks to complete. If an activity will most likely take longer than that time frame it should be broken down into smaller more manageable steps.

**Plan by Feature:** Priority is decided for each features based on factors such how mandatory are they for system operation, are they a prerequisite for other features and do they improve system usability. These are taken into account and a development plan is made.

**Design by Feature:** The necessary functions for each feature are defined and a sequence diagram is made which will be implemented in the next step after a design inspection.

**Build by Feature:** The feature is developed and implemented and unit tests are made for it.

FDD is typically for development teams so it nee

Reasons that FDD is suitable for this project:

- The straight forward five step process makes it easy to get started quickly.
- Risk of not having a finished product is reduced as a base application is built early to slot features into.
- Easy to adapt and add new features as new needs arise.

Reasons that FDD is unsuitable for this project:

- On smaller projects only one person creates models.
- No written documentation.

### 3.1.2 Feature List

This section lists and describes the features that were decided on for the application

- Image classification - Any user can upload an image that the system will attempt to classify and return that label to the user.
- Background removal - Users can upload an image which the system will analyse and attempt to remove the background. The resulting image with a transparent background will be provided back to the user.

- Submit feedback - After having an image classified and isolated the user can leave feedback details such as label accuracy, isolation quality and general comments.
- View gallery - Users can navigate to the gallery page to see all images uploaded by users that have been given mod approval.
- Filter gallery - Users can filter the gallery by animal species or type of image e.g. original or isolated image.
- Theme selection - By default the website has a dark theme but users can change it at any time to a different colour scheme.
- Account registration - Users can create an account for the application.
- Account login - Once a user is registered they are able to login to their account.
- View own uploads - Registered users have access to a personal gallery where they can see the results of all the images they uploaded regardless of whether they have mod approval or not.
- Mod approval - Admins have access to an admin approval page where they can approve, deny or delete any user uploads.

## 3.2 Use Case Diagram

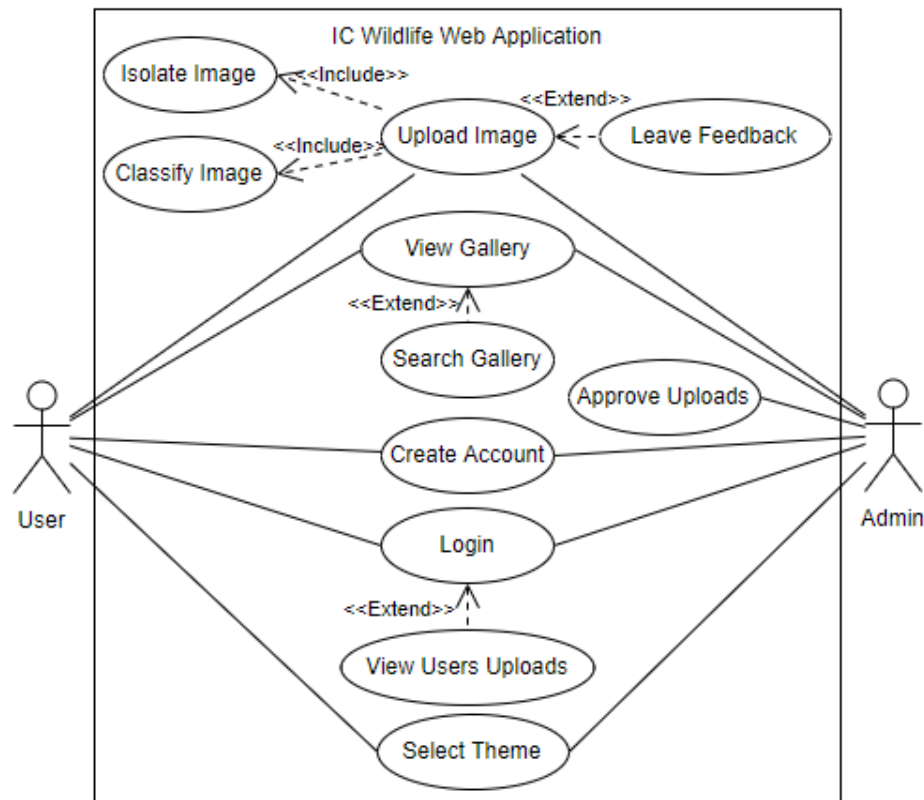


Figure 2: Use Case diagram for IC Wildlife web application

The use case diagram above shows the base functionality of the IC Wildlife web application. The main use case is Upload Image and it is key to the application as it includes the two most complex use cases and extends another. There are two different kinds of actors that will use the application, users and admins. Admins have access to all the same use cases as base users except they also have access to approving uploads to appear in the public gallery.

### 3.3 User Interface Design

When making a user interface there are a number of best practices that should be followed.[19] These include keeping interfaces simple, using common UI elements, and being purposeful in page layout. These were all kept in mind while creating the initial screen wireframes.

#### 3.3.1 Screen Wireframes

The initial designs were made assuming they would be viewed on desktop screens, however they are placed in a loose grid layout so elements can easily be scaled or moved to adjust for smaller screens.

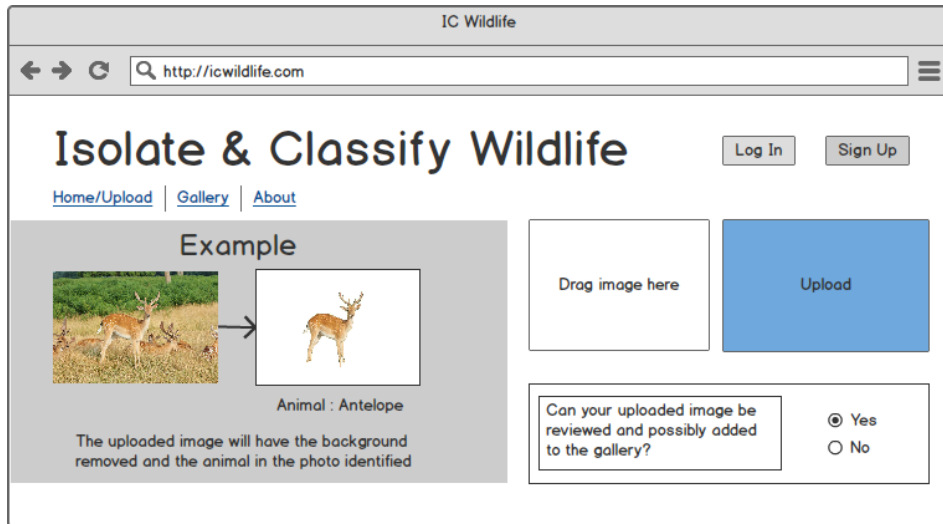


Figure 3: Home page wireframe

The navigation bar is at the top of the page and contrasts the background. Log in and sign up are easily visible but further away due to the fact they shouldn't need to be used more than once a visit. An example of what the site does is displayed prominently on the page. This visually shows the user that an image of an animal going in will result in a label being added and the background of the photo being removed.

The upload section has clearly labelled buttons with a more prominent colour on the upload button. The buttons along with a permission input are grouped

together in a square. An example of how display will be handled for smaller screens is that this square would be shifted to underneath the example images to fit the different scale.

The wireframe shows a web browser window titled "IC Wildlife" with the URL "http://icwildlife.com". The page has a navigation bar with links for "Home/Upload", "Gallery", and "About", along with "Log In" and "Sign Up" buttons. The main content area is divided into two columns. The left column, titled "Uploaded before and after", shows a before-and-after image of a black dog with the label "Animal : Dog" and a prompt "Please give feedback below". The right column features a "Drag image here" box, an "Upload" button, and a feedback question: "Can your uploaded image be reviewed and possibly added to the gallery?" with radio buttons for "Yes" (selected) and "No". Below these is a "Feedback Form" section with three parts: a required question "Was the image labelled correctly?" with "Yes" and "No" radio buttons; another required question "How effective was the background removal?" with radio buttons for "Perfect", "Good", "Poor", and "Completely Wrong"; and two optional text input fields for comments on labeling/background removal and general website suggestions, followed by a "Submit" button.

IC Wildlife


← → ↻ http://icwildlife.com

# Isolate & Classify Wildlife

Log In Sign Up

[Home/Upload](#) | [Gallery](#) | [About](#)

## Uploaded before and after



Animal : Dog

Please give feedback below

Drag image here

Upload

Can your uploaded image be reviewed and possibly added to the gallery?

☒ Yes  
☐ No

## Feedback Form

\* Required

Was the image labelled correctly?

☐ Yes  
☐ No

\* Required

How effective was the background removal?

☐ Perfect  
☐ Good  
☐ Poor  
☐ Completely Wrong

Optional

Any comments on the labeling or background removal?

Optional

Comments or suggestions for the website/project.

Submit

Figure 4: Home page after upload wireframe

After an image is uploaded the example is replaced with the before and after images. This prevents the screen from getting cluttered by having two similar elements stacked on top of one another. The upload buttons remains so that additional uploads can be made.

An optional feedback form appears below. The most important feedback has simple radio button inputs as a too involved process might discourage users from leaving feedback. The ratings for background removal are a bit subjective so a button to show example images for each rating will be added. The comment input boxes are non mandatory as many users will simply not have any comments they wish to say. The submit button will have it's size and positioning adjusted to make it more prominent.

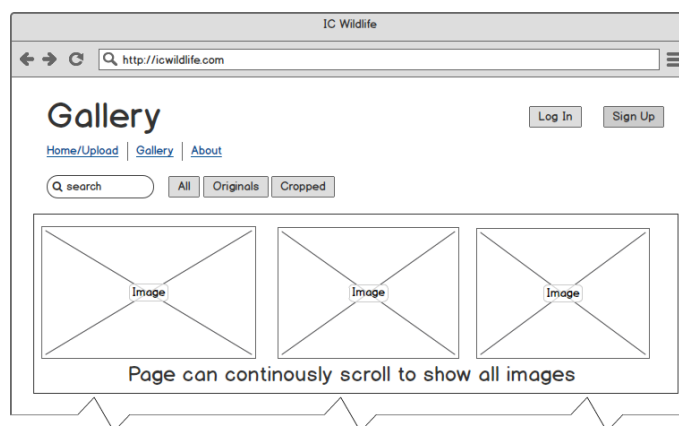


Figure 5: Gallery page wireframe

The gallery page by default displays all images and can be filtered by searching a species of animal and specifying original or cropped images. The images are displayed along with their labels and the page will continually scroll to show all images.

The my uploads page is not shown as it is identical to the gallery page except only images that the logged in user uploaded will be displayed.

The wireframe shows a web browser window titled "IC Wildlife" with the URL "http://icwildlife.com". The page has a navigation bar with links for "Home/Upload", "Gallery", and "About". The main heading is "Log In". There is a "Sign Up" button in the top right corner. The login form consists of two input fields: "Email Address:" and "Password:". Below these is a blue "Log In" button. At the bottom, there is a link "Don't have an account?" and a grey "Sign Up" button.

Figure 6: Login page wireframe

A basic form for logging in. It is done in a simplistic style as there is no need to over complicate straight forward pages. Client side error checking will be used to inform users if invalid input such as a blank password is used on a login attempt and server side validation will inform of incorrect login information. A link to the sign up page is included at the bottom for convenience and also to indicate to users who may have misclicked sign up that this is the wrong page and this button can get them to the correct one.

The wireframe shows a web browser window titled "IC Wildlife" with the URL "http://icwildlife.com". The page has a navigation bar with links for "Home/Upload", "Gallery", and "About". The main heading is "Sign Up". There is a "Log In" button in the top right corner. The sign up form consists of three input fields: "Email Address:", "Password:", and "Confirm Password:". Below these is a blue "Create Account" button.

Figure 7: Sign up page wireframe

The sign up page follows the same simplistic styling as the login page. Client



side validation will check that the password fields match while server side validation will check if the requested username is available.

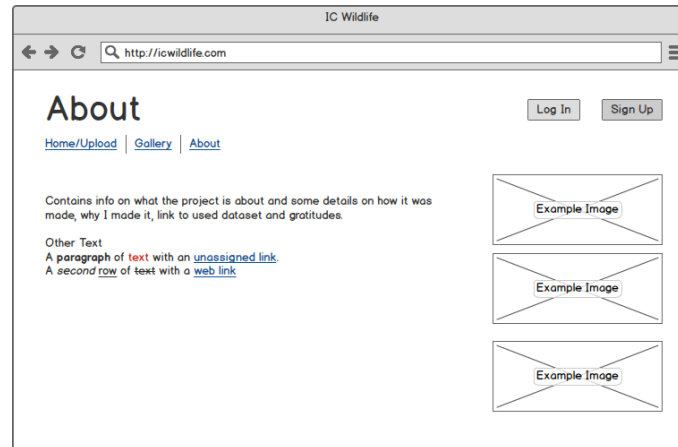


Figure 8: Sign up page wireframe

The about page has basic information on it and is only a display page with no unique functionality. Images are added to the side of the page to make it more visually pleasant.

TODO: Add conclusion ??????

## 4 Architecture and Development

### 4.1 Technical Architecture

TODO: All Technical

### 4.2 Development

TODO: All Development

## 5 Testing and Results

### 5.1 Test Plan

TODO: All Test

## 6 Project Plan

### 6.1 Original Plan and Changes

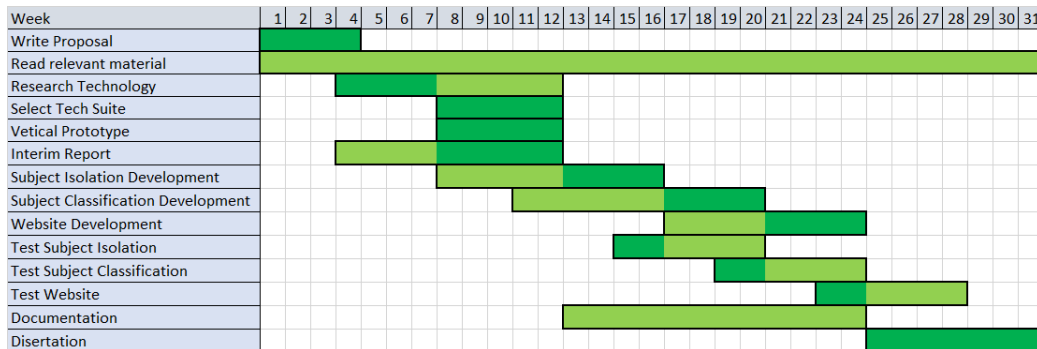


Figure 9: Original Gantt chart

The Gantt chart above shows the initial timeline for this project. Dark green indicates that for that week the associated task is high priority while light green is low priority. The three code deliverables are the subject isolation program, the subject classification program and the website.

TODO: REWORD section from interim, too much first person language

### 6.2 Key Differences

TODO: All Key

## 7 Conclusion

### 7.1 Key Learning Obtained/Findings

TODO: All Key

### 7.2 Future Work

TODO: All Future

## 7.3 Closing Statements

TODO: All Closing Statements

## References

- [1] Baxter R, Hastings N, Law A, Glass E. Object Recognition as Machine Translation Learning a Lexicon for a fixed Image Vocabulary; 2008.
- [2] Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts; 2008. Available from: <https://apps.dtic.mil/docs/citations/ADA640016>.
- [3] Annalisa B, Francesca O, Alessandro V. Hausdorff Kernel for 3D Object Acquisition and Detection; 2002. Available from: <https://pdfs.semanticscholar.org/cf26/a593ceef822f666e1b472253614f946a255e.pdf>.
- [4] Lowe D. Object Recognition from Local Scale-Invariant Features; 1979.
- [5] Gao W, Yang L, Zhang X, Liu H. An improved Sobel edge detection; 2010.
- [6] Canny J. A Computational Approach to Edge Detection; 1986.
- [7] Kittler J. On the accuracy of the Sobel edge detector; 1983. Available from: <https://www.sciencedirect.com/science/article/pii/0262885683900069>.
- [8] Russakovsky O, Deng J, Su H, Krause J, et al. ImageNet Large Scale Visual Recognition Challenge; 2015.
- [9] Everingham M, L VG, Williams C, Winn J, Zisserman A. The pascal visual object classes (VOC) challenge; 2015.
- [10] Rother C, Kolmogorov V, Blake A. “GrabCut”: interactive foreground extraction using iterated graph cuts; 2004.
- [11] Osma-Ruiz V, Godino-Llorente J, Sáenz-Lechón N, Gómez-Vilda P. An improved watershed algorithm based on efficient computation of shortest paths; 2007.

- [12] Parkhi O, Vedaldi AV A Jawahar. Cats and Dogs; 2015.
- [13] Pulli K, Baksheev A, Korniyakov K, Eruhimov V. Real-time computer vision with OpenCV; 2012. Available from: <http://dl.acm.org/citation.cfm?doid=2184319.2184337>.
- [14] Amazon Web Services Inc. About AWS; 2018. Available from: <https://aws.amazon.com/about-aws/>.
- [15] Digital Ocean LLC. About DigitalOcean; 2018. Available from: <https://www.digitalocean.com/about/>.
- [16] Ronacher A. Opening the Flask; 2011. Available from: <http://mitsuhiko.pocoo.org/flask-pycon-2011.pdf>.
- [17] Django Software Foundation. The Web framework for perfectionists with deadlines; 2005. Available from: <https://www.djangoproject.com/>.
- [18] Oracle Corporation. About MySQL; 2018. Available from: <https://www.mysql.com/about/>.
- [19] Jesse G. The Elements of User Experience; 2010. [Accessed 03 Jan. 2019]. Available from: <http://ptgmedia.pearsoncmg.com/images/9780321683687/samplepages/0321683684.pdf>.