



## Recommending photo filters via image content in a Web App & REST API



**Joshua Bridge**  
joshua.m.bridge@stu.mmu.ac.uk  
Student Number: 14032908  
BSc (Hons) Software Engineering

Unit: 6G6Z1101 Project  
Assignment Number: 1CWK100  
Supervisor: Dr John Darby

April 27, 2018

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed *Joshua Bridge*

## **Abstract**

*To be completed.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Report Structure . . . . .	2
1.2	Required Resources . . . . .	3
1.3	Aims . . . . .	3
1.4	Objectives . . . . .	4
<b>2</b>	<b>Literature Review</b>	<b>5</b>
2.1	Background Information . . . . .	5
2.1.1	Digital image processing . . . . .	6
2.2	Recommendation systems . . . . .	7
2.2.1	Collaborative recommendation . . . . .	8
2.2.2	Content-based recommendation . . . . .	8
2.2.3	Hybrid recommendation . . . . .	9
2.3	Image filtering system . . . . .	11
2.3.1	Instagram . . . . .	11
2.3.2	Personalisation of image enhancement . . . . .	12
2.4	Conclusion . . . . .	13
<b>3</b>	<b>Design</b>	<b>14</b>
3.1	Requirements . . . . .	15
3.2	Application architecture proposal . . . . .	15
3.3	API . . . . .	16
3.3.1	REST & JSON . . . . .	16
3.3.2	Web Framework . . . . .	18
3.3.3	Database Storage . . . . .	18
3.3.4	Image Filtering . . . . .	20
3.3.5	Image Analysis . . . . .	21
3.3.6	Recommendation . . . . .	22

3.4	Front-end . . . . .	24
3.4.1	Frameworks & Libraries . . . . .	24
3.5	Use Case & Class diagram . . . . .	25
<b>4</b>	<b>Implementation</b>	<b>26</b>
4.1	API . . . . .	27
4.1.1	Image handling . . . . .	27
4.1.2	Web Responding . . . . .	28
4.1.3	Image Analysis . . . . .	31
4.1.4	Recommendation . . . . .	32
4.1.5	Database . . . . .	33
4.2	Front-End . . . . .	34
4.3	Deployment . . . . .	36
4.3.1	API . . . . .	36
4.3.2	Front-end . . . . .	37
4.3.3	Database . . . . .	38
4.4	Testing . . . . .	39
<b>5</b>	<b>Evaluation</b>	<b>41</b>
<b>6</b>	<b>Conclusion</b>	<b>45</b>
6.1	Further work . . . . .	46
6.2	Author reflection . . . . .	46
<b>Appendices</b>		<b>50</b>
<b>A</b>	<b>Terms of Reference</b>	<b>51</b>
<b>B</b>	<b>Ethics documents</b>	<b>57</b>
<b>C</b>	<b>Presentation slides</b>	<b>68</b>
<b>D</b>	<b>Product Link</b>	<b>73</b>
<b>E</b>	<b>Questionnaires</b>	<b>74</b>

# List of Figures

2.1	An example of a photo filter . . . . .	7
2.2	System component architecture of Spotify’s ‘Discover Weekly’ . . . . .	10
3.1	Architecture flow from client to API and their interactions . . . . .	16
3.2	Example flow of API access from client . . . . .	17
4.1	Loading an image into Pillow from an external URL, and from NumPy . .	27
4.2	Python filter classes . . . . .	29
4.3	Flask image upload API method . . . . .	29
4.4	Python syntax for declaring a ‘Dict’ object . . . . .	30
4.5	The filter selection page of the front-end web app . . . . .	34
4.6	The filter selection page - scaled to a mobile screen size . . . . .	35
4.7	API Deployment - AWS EC2 instance status . . . . .	36
4.8	API Security - Domain SSL certificate information . . . . .	37
4.9	API Security - Secure connection to API . . . . .	37
4.10	MongoDB Atlas - Deployed cluster dashboard . . . . .	38
E.1	Participant 1 questionnaire . . . . .	74
E.2	Participant 2 questionnaire . . . . .	75
E.3	Participant 3 questionnaire . . . . .	75
E.4	Participant 4 questionnaire . . . . .	76
E.5	Participant 5 questionnaire . . . . .	76

# List of Tables

4.1	Participant responses to recommendation accuracy . . . . .	39
4.2	Participant responses to UI simplicity . . . . .	40

# Chapter 1

## Introduction

Online social media has become one of the fastest expanding industries in the last few decades, with Instagram (<https://instagram.com>) boasting over 700 million users (Instagram 2017). Facebook (<https://www.facebook.com>) says that its users upload one billion photos each week, with a total of 260 billion images currently being hosted on their site (Beaver et al. 2010). On Instagram users can apply photo filters before posting the photo to their profile, in an effort to personalise the images. Other photo filtering apps exist such as VSCO, Snapseed by Google, Afterlight, and many others. While all of these apps allow users to apply filters to the images along with a more fine-grained editing suite, none of them seem to offer a recommendation for which filter would be best suited for the image being edited.

In this project an application will be proposed which is made up of two major parts. The first is an online API which provides a service of photo filtering over the internet, which other applications or social media sites could potentially integrate to their photo sharing platforms. Other services of this nature do exist such as Pho.to (<http://developers.pho.to>) and Cloudinary (<https://cloudinary.com>) however to the authors knowledge, like the apps mentioned previously, these services do not offer a recommendation specifically for photo filters.

Late into the project development an app called askAI (VicMan LLC 2017) was discovered which does provide recommendations for photo filters. In contrast to the application proposed in this report the app is only available as a downloadable mobile application rather than a web app. Furthermore the app seems to only recommend Instagram-specific filters, rather than providing custom filters of its own and the recommendations are not available as an external API. The Pho.to API mentioned previously appears to be made by the same developers of the askAI app, though they do not appear to be connected in functionality.

## 1.1 Report Structure

The structure of the following report will first consist of a preliminary review of related literature. Once this has been completed a design will be proposed including justifications for each technology which is to be implemented into the final product. It will also define clear objectives which will be used to measure the success of the product. The next chapter will include details of how this design was implemented into a working application along with examples of performance and any issues encountered while creating the product. The penultimate chapter will be an objective evaluation of the products performance in relation to the objectives outlined in the design chapter. Finally, a conclusion will summarise the experiences and findings of the report. After this will be a bibliography containing all references in the report along with an appendix containing various project related documents.

## 1.2 Required Resources

- PyCharm
- Python
- WebStorm
- React
- Node Package Manager
- Git SCM
- Amazon Web Services account
- Atom text editor (for L<sup>A</sup>T<sub>E</sub>X)
- TeX Live (for L<sup>A</sup>T<sub>E</sub>X)

## 1.3 Aims

In this report the following goals are what is expected to be achieved in the creation of the product:

1. To reduce complexity, information overload, and time spent for users when trying to customise photographs - usually in the expectation that the photographs will later be shared on social media.
2. To create a secure platform for other applications to utilise the advancements made in this project.

## 1.4 Objectives

The objectives below describe in more detail a set of more specific goals which should be achieved in order to create a successful product:

1. Implement a simple online web application for filtering images using a framework or library designed for front-end development across multiple platforms.
2. Implement a back-end application which is able to process images including applying filters and creating thumbnails.
3. Research various recommender systems to determine the common technologies/solutions for solving the problem of recommendations.
4. Implement a recommender system which is able to suggest photo filters to users based on image content.
5. Implement a RESTful API which exposes both the recommender system and the filtering system to the front-end web app and allows developers to connect them into their own applications.

# Chapter 2

## Literature Review

In order to understand the background of the technology involved in this project, it is necessary to complete a significant review into the current and past literature. This will help form a basis of knowledge from which the future development and analysis of the proposed system will utilise and build upon. This research will be vital in order to make use of the most optimal technology for any given problem.

The following review will be an investigation into the current knowledge of the more complex components of the proposed system, which are as follows:

- The recommendation system.
- The image filtering system.

### 2.1 Background Information

Before the main body of the review is carried out, the following paragraphs will aim to explain the basic premise of how digital images are stored and manipulated. This will serve as a basis of knowledge on which the rest of the report will rely upon.

### 2.1.1 Digital image processing

Image processing deals with the analysis and manipulation of image data. An example of image processing is the use of digital signal processing. This involves converting analog sensory data from a digital camera sensor into a computer-interpretable format.

#### Bitmap Images

Before you are able to analyse an image, you must first represent the data in a way that it can be interpreted by a computer, and a human. One basic form of doing this is via a bitmap image. A bitmap - as its name implies - is a simple spacial mapping of values (bits) along a horizontal axis ( $x$ ) and vertical axis ( $y$ ). Using a greyscale image as an example, a bitmap representation of this would contain a number of values (or ‘pixels’), the number of which is equal to the product of the sizes of the  $x$  and  $y$  axes. Therefore an image of size 200x200 would contain 40,000 pixels. Each of these pixels contains an integer value representing brightness, typically ranging from [0,255] (the total value range of an 8-bit integer). In this case ‘0’ would represent a completely black value and ‘255’ would represent a completely white value. Note that sometimes these values can be represented by a decimal number in the range [0,1].

A colour image follows a very similar format, except now each pixel contains three brightness values instead of one. Each of these values map to the brightness of the colours (or ‘channels’) red, green and blue - in that order. Therefore a pixel with values (0,255,0) would be entirely green and a pixel with values (0,0,255) would be entirely blue. It should be noted that when these colours are displayed on a computer screen their colour values are additive (i.e. they can mix together to form a different, brighter colour). A pixel with values (0,255,255) would therefore represent cyan, and finally a pixel with values (255,255,255) would represent white. Note that the RGB representation of colour is just one form, there are others such as HSV (Hue, Saturation & Value) which use a different method of representing colours.

## Photo filters

A photo filter is a way of describing the process of altering an image's colour data, in order to produce a variety of different visual effects on the image. For example one photo filter could increase the contrast (i.e. increase the variance between dark and light pixels), or give it a more red-shaded appearance by increasing the brightness values in the red channel only. An example of a photo going through a process similar to this can be seen in figure 2.1. The filter used in this instance is one of the filters implemented in the API defined in this report.

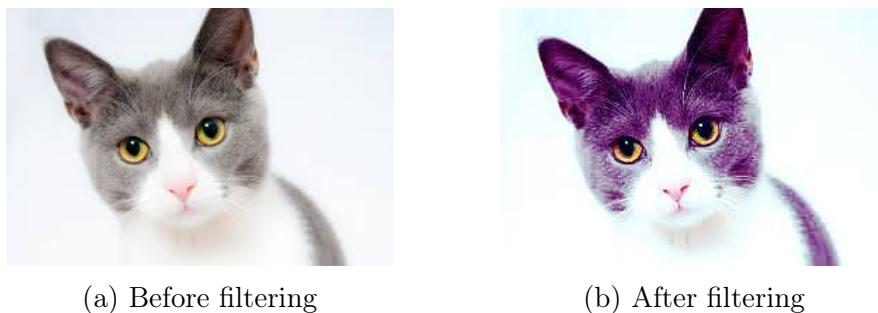


Figure 2.1: An example of a photo filter - original image from Pixabay (2013)

## 2.2 Recommendation systems

As defined in Ricci et al. (2011), recommending content is a problem which involves attempting to predict what a user may desire at any given time when using a system. In the ‘internet era’ applications of this are far-reaching, such as recommending products on Amazon.com Linden et al. (2003). Recommendation systems are a popular topic due to the “*abundance of practical applications that help users to deal with information overload and provide personalized recommendations, content, and services to them*” (Adomavicius & Tuzhilin 2005).

Jannach et al. (2010) describe three different methods for giving recommendations: Collaborative, Content-based, and Hybrid.

### **2.2.1 Collaborative recommendation**

When using an online streaming platform that serves video content such as Netflix or YouTube, a user will visit their site looking for something to watch. The problem faced by these sites is trying to find content that the user hasn't seen before, but will align with the user's interests enough to make them want to watch it (Davidson et al. 2010), (Gomez-Uribe & Hunt 2016). The information needed to carry out such a prediction can include the semantic value of the entire user's history of interaction with the system (often referred to as *metadata* (Duval et al. 2002)), from which preference can be extrapolated. If there is not enough metadata about the user from which to extrapolate preference with reasonable certainty, then preference can be inferred from other users of the system - especially users with a similar predicted preference.

### **2.2.2 Content-based recommendation**

There can sometimes be barriers to using collaborative recommendation, such as when the 'cold-start problem' is present. This problem describes when a user has just signed up to a website, it will not yet have enough information / user history to recommend them content (Schein et al. 2002). In this situation content-based recommendation could be used more effectively to build up more reliable recommendations (Lops et al. 2011). There are different kinds of content-based recommendation, as it can be applied to many different kinds of content. For example a system may work differently if the content is user-generated as opposed to internally or computer generated, where the information would likely be provided already. In a system with user-generated content, the information about that content would have to be inferred, or provided by the user.

## **Feature extraction for content analysis**

When content is user-generated and without a definition provided alongside it, its definition must be derived. For textual data this can be done by analysing the words and extrapolating a commonality between keywords to find a prevailing subject (Sanderson & Croft 1999). This type of process is often referred to as ‘Feature extraction’ (Guyon & Elisseeff 2006). When analysing raw data such as an image file or an audio file, this becomes a much more difficult task as you must first find a way of deciphering the content before deriving the subject or meaning.

Deep learning has recently become a common way of deciphering this content, including for image, audio, and video content types (Coates et al. 2011, Ciregan et al. 2012, Lee et al. 2009, Mobahi et al. 2009). Spotify has implemented this kind of deep learning to determine genre types from raw audio data - see section 2.2.3. This can be applied to image data also, for example in the context of a search algorithm, it would bring the most relevant images by content to the top of the results (Yee et al. 2003). This works similarly to how a recommender system will select the most relevant items for the user to see.

Google’s neural network implementation has proven to be a very reliable system for image classification (Krizhevsky et al. 2012). As it has a very high accuracy on most image types, it would be reasonably effective for providing a basis of feature extraction for use in a recommender system. In the system proposed in this report, scene classification along with prevailing colours in the image would be an ideal input into the recommendation system for determining optimal image filters.

### **2.2.3 Hybrid recommendation**

Generally, hybrid recommendation applies both of the previous methods together in order to draw up a more comprehensive list of recommendations, in order to avoid the individual

weaknesses of both types.

## Spotify - Discover Weekly

Discover Weekly is a playlist made every week for every user of Spotify. Instead of getting humans to curate a playlist for each and every user, an algorithm is used to try and find a set of songs which it predicts the user will like, but hasn't listened to before. As described by Popper (2015) it does this by mixing collaborative and content-based recommendation. The first is done by analysing all playlists on spotify and determining their similarity to playlists made by the user. The next step is looking for songs in those playlists that the user in question hasn't heard yet.

## Discover Weekly Data Flow

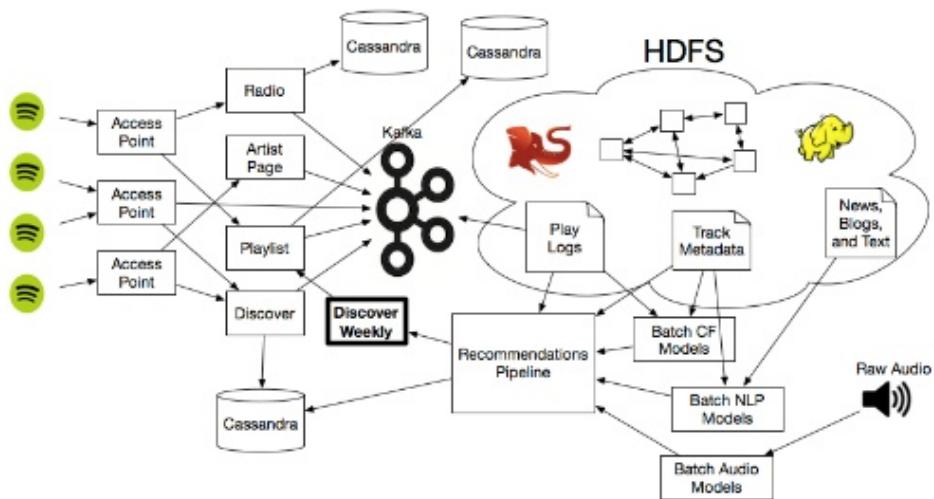


Figure 2.2: System component architecture of Spotify's 'Discover Weekly' - diagram re-used from Johnson (2015)

The second, more complicated part of this process is the content-based recommendation. Spotify's algorithms actually analyse the music data in several different ways to

determine a users ‘taste’. The first is by defining the genres a user listens to, which it does by scanning music blogs for what genre a certain piece of music is described as being. Using this as a search, spotify can then look for other music in that genre to recommend. The arguably more complicated part of this process is the auditory analysis of raw music data. As shown in figure 2.2 the raw audio is converted into ‘batch audio models’ where they can be used as a basis for recommendations.

## 2.3 Image filtering system

There is little to no research into the area of developing new photo filters, likely due to the fact that filters can be very subjective in terms of what effect they have on an image. Furthermore the relation of that effect to how many people choose that filter for a certain type of photograph would require a large data-set. However it is possible to determine the most common types of photographs posted online, which will guide the process of creating filter templates.

### 2.3.1 Instagram

In order to create image filters that users would want to apply to their photographs, an assessment must first be made about that the type of photos people tend to post online. Instagram is a popular online photo-sharing platform that attracts over 700 million users (Instagram 2017). An investigation by Hu et al. (2014) revealed that the types of photos posted on Instagram could be roughly categorised into eight types: “*self-portraits, friends, activities, captioned photos (pictures with embedded text), food, gadgets, fashion, and pets*”. The study clarifies that ‘self-portraits’ (or ‘selfies’) made up 24.2% of the images posted, and ‘friends’ made up 22.4%, totalling 46.6% of the images analysed. From this it can be hypothesized that creating filters which are complimentary for skin-tones would be a contributing factor to success when developing image-filters.

Hu et al. (2014) however did not study a wide proportion of the Instagram user-base, in fact they only closely studied 50 users, out of an initial 95,343 ‘unique seed users’, from a user-base of more than 150 million when the study was taken place. Furthermore the study only analyses pictures posted online, to Instagram, with public profiles, and from users which had “*at least 30 friends, 30 followers, and had posted at least 60 photos*”. The system proposed in this report would not be limited in terms of where the resulting image would be posted, and has no requirement that the image be posted to an online website (other than the system itself). Therefore the findings in the study should only be used as a loose guideline for popular image categories when defining image-filter styles.

### 2.3.2 Personalisation of image enhancement

There has been some work done on the mapping of image enhancement styles to personal taste, although the field appears to be still in early stages. Two studies in particular have heavily contributed to the development in this field. Kang et al. (2010) found that when given a set of images, participants would deviate into different styles when asked to choose between two options of enhancement (e.g. lighter or darker). Furthermore, many participants preferred their own enhancement when compared to an averaged enhancement, suggesting that personalised enhancements would be a valuable tool.

Using the work done by Kang et al. (2010) as a basis, Caicedo et al. (2011) developed a method of personalised image enhancement that focused on putting users into groups (or ‘clusters’) that had similar tastes within image enhancement. Therefore when attempting to apply personalised enhancement on a per-user basis, instead of determining a specific users preference from scratch, the user’s preference would only have to be mapped to a pre-defined group. Using this method the authors suggest that personalisation of image enhancement could be scale-able to large amounts of users.

## 2.4 Conclusion

Posting images online is fast becoming one of the biggest trends in recent history, as evidenced by Instagram's user-base expanding to over 700 million users (Instagram 2017). While not much public research seems to have been carried out into the reasons why users pick certain styles of image-filters before posting, it certainly appears viable to aid them in making that process easier. The findings of this review suggest that feature extraction could be a useful tool in developing recommendations for filters. This would be carried out in the form of image classification, most commonly to determine whether a person is present in a photograph which would thereby influence the algorithm's decision of what filters to recommend to the user for a certain image.

The findings of this review also suggest (mostly from Kang et al. (2010), Caicedo et al. (2011)) that the field of personalised image enhancements (or filters) is a relatively new field of study, therefore there is a lot of room for collecting new findings by extending upon previous works in a new implementation.

# Chapter 3

## Design

In this chapter a detailed description of the product design will be laid out, including explanations for each choice of technology and how it will enable the product to achieve its goal. The product can essentially be split up into two major components - a RESTful API & a web-based front-end. Creating an API is an important factor in the implementation as it defines how the flow of the client application will work. As a result, technologies must be chosen which easily make use of this type of application-flow such as a framework which integrates well with AJAX calls - a common way of connecting to APIs. The front-end development is also a crucial part of the implementation as it acts as a testing interface for the API. If the API is not structured well then designing a front-end around it is much more difficult, meaning any future developers wishing to use it might be dissuaded by its complexity and end up choosing a different service. Developing both at the same time makes it much easier to see how clients will access the service, therefore making it much easier to remove any complexity it may contain while the service is still being built.

### **3.1 Requirements**

The application proposed in this report would be best utilised on multiple platforms & devices, as the service it would offer is platform-independent (i.e. all you need is an image file). For this reason the primary focus will be making an online web-app, which connects to a separate REST (Fielding & Taylor 2000) API which performs the main functions of the application, without the clutter of the front-end mixing in with it. This would also allow other apps and services to utilise this API and integrate it with other apps, such as on social media apps when sharing an image.

When designing a user-facing website and an API, there are certain considerations that have to be met, such as data security and ease of use. The following features define what the final product should achieve:

- A web-based user interface.
- A user-history (metadata) tracking mechanism.
- A RESTful API which can be utilised by any application.

Following these features, these are some of the things that should be taken into consideration when implementing these features:

- The user interface should be easy to use.
- It should not store any more information about users than it has to.
- It should maintain a reasonable level of security to prevent user-data from being stolen.

### **3.2 Application architecture proposal**

Figure 3.1 shows the proposed architectural flow for the product being created using an example image from Pixabay (2013). It shows how an image is uploaded to Amazon

S3 before being analysed, then the filtered previews are uploaded to S3 before the API response is generated.

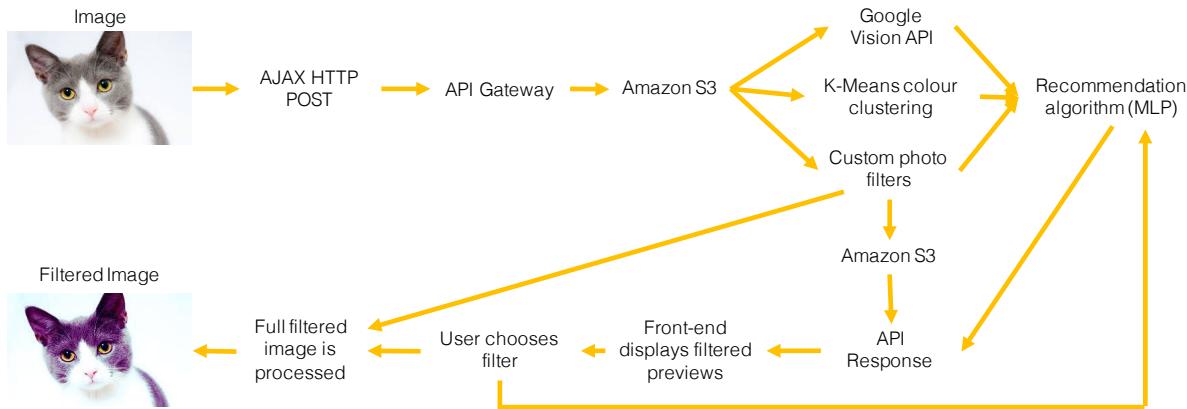


Figure 3.1: Architecture flow from client to API and their interactions

The final version of the API and front-end components will be hosted separately on AWS due to its high configurability, with support for multiple different platforms and programming languages.

### 3.3 API

The API is to be written in Python (<https://www.python.org>) due to its high suitability for dealing with image data, web-access and recommender algorithms. The application is able to be completely self-contained as it can perform all of the functions required in a single project.

#### 3.3.1 REST & JSON

When developing an API, one of the biggest considerations which must be made is how the end user will be interacting with it. If an API is too complex and performs against a developer's expectations, it could make it much more likely that developer might abandon it in favour of another competing product. Following an established architectural guideline makes it much easier to develop a clear API while spending more time focusing

on developing core features. REST is a guideline developed by Fielding & Taylor (2000) which describes a resource-oriented method of designing APIs. This is the guideline that the API in this report will be attempting to follow.

As many of the interactions with other services in this project are required to be completed using JSON as an interchange format, it stands to reason that it would be a good idea to also use this format for API responses. This is not just because it is popular, but also that converting data between different formats could be seen as unnecessary complexity within the application.

## URI design

As there are not many operations which are required to be completed by the API, the number of endpoints can be kept to a relative minimum. Figure 3.2 illustrates an example of how a client would connect to the API where the orange rows indicate a server action and the white rows indicate a client action.

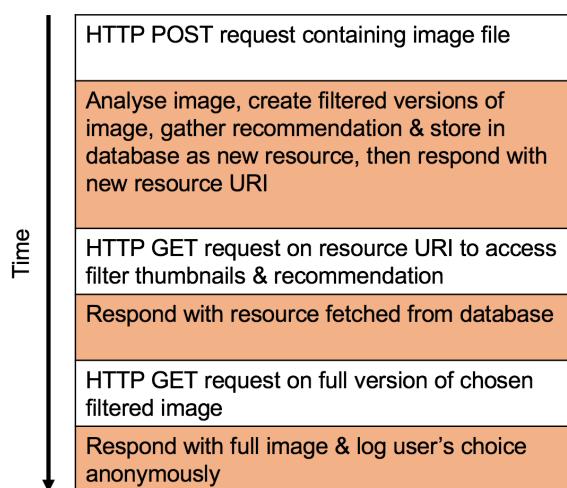


Figure 3.2: Example flow of API access from client

Using figure 3.2 as the expectation for what interactions will happen on the sever, a simple URI of '/images/' can be created for the first HTTP POST request. Then the new resource URI in the accompanying response could be an extension of this, con-

taining a generated unique id for the resource. For example ‘/images/1234’ can be used to access the resource. The full version of the image then can be available at ‘/images/1234-filter-name.jpg’, with that URI contained within the original resource body so the client can easily access it.

### 3.3.2 Web Framework

A simple framework for handling HTTP access to the API endpoints is critical when creating an application which is meant to be able to handle high-volume traffic. Flask (<http://flask.pocoo.org>) performs exactly this purpose as it is a very lightweight framework which provides very easy methods for transferring data in different formats such as JSON.

Other frameworks could be considered for this purpose, however few match Flask in terms of low complexity and ease-of-use. Django (<https://www.djangoproject.com>) is another very common web framework for Python which comes with many useful features such as a built-in user system. Many of the features, however, are not required within this project and would only increase the complexity of the code in the application. When applying the Lean Software Development methodology (Poppdieck & Poppdieck 2003) this would come under the heading of eliminating waste - for example spending less time learning a complex framework and more time actually developing core features & meeting requirements. Therefore Flask is a good foundation upon which to build a RESTful API that is easy to use & easy to develop.

### 3.3.3 Database Storage

A database is a crucial component within an application therefore choosing a suitable one is an important step in creating an application. The choice of database technology falls into a few categories such as SQL/NoSQL (or Relational/Non-Relational). There are

several differences between these database types such as ability to scale and the types of data which can or should be stored by them. Non-relational databases are useful when the data to be stored is not of a strong structure, where relational databases are more suited for structured, related information. For example a relational database would be well suited for storing products on an online store, where different data such as products or orders can be stored in different tables, with connecting foreign keys. SQL databases are also very good at managing consistency e.g. when data is inserted into a table, there will definitely only be one entry input into the database. This is in comparison to NoSQL where horizontal scaling causes there to be a higher chance of duplicate entries, in an effort to ensure no data is lost while maintaining fast transaction speeds.

NoSQL databases are often easier to host than SQL databases due to their ability to scale horizontally (Cattell 2011) rather than just vertically (i.e. adding more servers to handle load rather than upgrading from one machine to another higher power machine). This is applicable in the context of this project as a requirement (see section 3.1) is that the application should be able to scale well with more users.

When considering the type of data which needed to be stored by the application a NoSQL database seems to be best due to the non-relational types of data to be stored such as API responses. Furthermore speed is a key part as every bit of processing has to be done fast enough for the client not to time out waiting for a response. MongoDB (<https://www.mongodb.com>) is a popular NoSQL database with easy connectivity to python using the library pymongo (<https://api.mongodb.com/python/current/>). MongoDB is very fast with access speeds of up to 10 times faster than MySQL (Han et al. 2011) and also comes with a strong query language therefore it is highly suited for use in the application.

### 3.3.4 Image Filtering

The main bulk of processing done in the API can be broken down into three main components which are filtering, analysis, and recommendation. Filtering is the first step and will involve work to create multiple different photo filters which can be applied to any image which gets submitted to the API. Due to pythons ability to manipulate image data well using several different libraries such as NumPy (see section 4.1.1) and Pillow (see section 4.1.1).

When an image enters processing it can be loaded into python using Pillow which creates an array containing each pixel value within the image. The pillow library provides many default methods for manipulating images such as changing brightness or contrast. Furthermore it contains methods which allow it to be transferred to different datatypes such as a NumPy array, where other kinds of mathematical operations can be applied to the pixel data such as linear interpolation. The image can also then be split into three different arrays with one for each colour channel of red, green, & blue and then operations can be applied individually to create a colour effect on the image. For example if the red channel only was brightned by increasing each pixel value by a certain amount, then when the channels were put into a single image again this would give the image a red tint, which could be desirable for use as a filter.

Due to the nature of the application trying to recommend filters, it would stand to reason that the filters in the application should be easily distinguishable to ensure maximum seperability between them. When being entered to the AI component (see section 3.3.6) it is much better to have high seperability between classes (or filters) in order to find the relationship between input data and its classification.

### 3.3.5 Image Analysis

The second stage of processing (though not related to the first stage) is to analyse the image in order to gather input information for the recommendation algorithm. Providing raw pixel data would not be ideal when processing larger images as this will likely increase processing time greatly. When trying to build a dataset for use in training the recommendation algorithm it would not be ideal to store every picture input into the application as this violates one of the requirements in section 3.1. Furthermore the raw pixel data will not provide extracted meaningful information such as dominant colours and face detection. Therefore, the best way to proceed is to extract and store information about the image itself without storing any actual image content. As found in section 2.3.1 self-portraits are a highly common type of image posted online so it could be meaningful to extract whether an image contains one or more faces in it. This information could affect how someone chooses which filter they would like to apply to an image so it is vital in this application to include this information. A link between a person in a photograph and the filter being chosen could exist if one of the filters was more complimenting to skin tones for example. In order to find the dominant colours in the image and apply face detection two different methods will be applied and are described below.

#### Dominant colours: K-Means

K-Means clustering (MacQueen et al. 1967) is a form of machine learning which performs a mathematical analysis on un-labeled data in order to find common groupings within that data. The number of clusters (or groupings) is first specified and the algorithm then splits the data points into that number of groups and then finds an average value for each cluster. K-Means has been used for image segmentation (Coleman & Andrews 1979, Shi & Malik 2000) which is an important part of computer vision. In this case K-Means clustering will be used to find the dominant colours within the image along with their amount of use within the image as a percentage. The colours found from this process

will likely not be exactly as they appear in the image due to the averaging of each colour cluster to find the most common groups of colours.

The K-Means algorithm used in this application will not need to be implemented as there is one provided by the scikit-learn library (<http://scikit-learn.org/stable/>) which meets all requirements and makes it easier to implement K-Means directly into the application.

### **Face detection: Google Vision API**

The Google Vision API (<https://cloud.google.com/vision/>) is a useful tool for performing a variety of image analysis steps such as face detection and logo detection. It can be accessed via a REST API using JSON as the request language therefore it should integrate into the application with relative ease using the libraries available in python such as ‘requests’ (<http://docs.python-requests.org/en/master/>). Google has not published what technology exactly powers the vision API other than saying in a blog post that it is run using machine learning and TensorFlow (<https://www.tensorflow.org>) (Google 2015). However, due to the complex nature of face and logo detection it can be assumed that the systems behind the API are running some very complex machine learning models which require a lot of processing power to run. With this in mind it will be much simpler to integrate this API into the application rather than try to implement a custom version of face detection into the application.

#### **3.3.6 Recommendation**

The final stage of processing within the application is to get the recommendation for which filter would be best applied to the image. As discovered in section 2.2 a good way of doing this is to use neural networks in order to build a relationship between input data and a possible classification. Within this application a form of hybrid recommendation will be used whereby the input to the algorithm will be the results from image analysis

as discussed in the previous section. In order to find a relationship between this data and a filter, some data will have to be provided by some initial prototype users in which they will choose a filter for a set of photos of their own choosing.

Using the stored image analysis data as parameters and the chosen filters as classifiers, a machine learning model can be built which hopefully finds a significant link between image content and the filters which could get applied to them. This is a form of hybrid recommendation as the users interests are powering the recommendation but that information is linked directly to the content of the image, thereby mixing collaborative and content-based recommendation. However, this is not strictly including true collaborative recommendation as the users are not being recommended based on tastes of similar users, but more that the collective users are helping to power the content-based recommendation.

## Multi-Layer Perceptron

Multi-layer Perceptrons (Minsky et al. 2017) are a type of neural network which can be used for classifying non-linearly separable data. A multi-layer perceptron consists of 1 input layer, 1 output layer and 1-to-many hidden layers which contain neurons that perform an activation function on their inputs. In the case of this application the multi-layer perceptron will be used as a recommendation algorithm by ‘learning’ the connection between image content and image filters. While it may not be best suited for recommendation in place of deep learning for example (section 2.2), the MLP is already implemented in scikit-learn (<http://scikit-learn.org/stable/>) and therefore will be much easier to integrate within the application. Furthermore supervised deep-learning is more suited to dealing with pixel data, which this application aims to not keep therefore training a deep-learning model would be much more difficult.

## 3.4 Front-end

A front-end is an ideal way to create a testing interface for the API and is a good way to advertise to potential users of the API. Therefore one should be built which is capable of running on multiple platforms including mobile phones and desktops. This is so that it is accessible to as many people as possible, by removing as many impedences to accessing the application as possible. The most suitable way of achieving this would be to make the front-end a web-based service which could scale to various screen sizes, and thereby making the application accessible to all with no downloads required.

### 3.4.1 Frameworks & Libraries

To achieve the best results in terms of aesthetics, functionality and work-effort, a front-end JavaScript framework is ideal for achieving the best results in the shortest time. Their pre-made code structures often eliminate a lot of the grunt work from creating reactive user interfaces from scratch, therefore they are an obvious choice.

While there is little in the way of academic comparisons between JavaScript frameworks as told by Graziotin & Abrahamsson (2013), a subjective comparison can be made to find the most suitable framework for this application. Two commonly used libraries/frameworks are React and Angular which are discussed briefly below.

#### React Vs Angular

React (<https://reactjs.org>) is a JavaScript library for creating user interfaces which was created by Facebook. As it is entirely in JavaScript and introduces a component structure which updates when any data state changes, it is a simple way to add lots of functionality in a short time. Angular (<https://angularjs.org>) is another JavaScript library however it is not the recommended version which is Angular 4 (<https://angular.io>). Angular 4 is a framework rather than a library and uses TypeScript which could

introduce complexity, and it comes with a lot of baggage which could bring a much steeper learning curve as discussed by Neuhaus (2017).

### **Choosing React**

ReactJS is the library which will be used in this application due to its flexibility and ease of use with a relatively small learning curve. React Native (<https://facebook.github.io/react-native/>) is a variant which can be used to run the JavaScript code as a native mobile app. This is commonly used to build mobile apps, and has been done by some such as Instagram (as stated on the React Native website). While React Native will not be used in this report, it is important to bear in mind the code would be the same, and that React in general is highly optimised for mobile development. Due to these factors React is highly suited for the requirements of this application, as long as the final product is able to easily scale between different browser sizes including desktop and mobile browsers.

## **3.5 Use Case & Class diagram**

# Chapter 4

## Implementation

In this chapter the implementation for the application proposed in this report will be explained. Both the API and the front-end will be developed concurrently to ensure that each of them are able to interact with each other in the easiest way possible. For example if any issues were found when accessing the API using the prototype front-end then they could easily be fixed by changing how the API returns its data.

Once the code for both the front-end and API has been finished, the resulting applications will be deployed to AWS (<https://aws.amazon.com>) and put under the domains of ‘<http://inflex.co>’ and ‘<https://api.inflex.co>’. Some prototype versions will also be tested throughout for hosting on AWS to ensure the final deployment will be successful.

## 4.1 API

In this section the implementation details for the API including filtering, recommendation and web responding will be explained.

### 4.1.1 Image handling

While python was chosen for its suitability for image handling, there is not much guidance online for how to handle images in the context of trying to apply photo filters to them. In order to do this a more fundamental understanding of how photo filters worked was required. Essentially it was found that colour effects can be applied individually to each channel (the red, green, and blue channels) in order to adjust the colour palette for that image.

In order to do this an image must first be loaded into memory, which it was found can be done using the Pillow library (<https://pillow.readthedocs.io/en/5.1.x/>). Figure 4.1 shows the code required to import an image into memory from an external URL using the Requests library (<http://docs.python-requests.org/en/master/>). This is then stored as an ‘Image’ object for which there are various methods within the Pillow library for performing enhancements such as sharpening, brightness adjustment, and contrast adjustment (though these are not per-channel adjustments, rather they applied globally to every channel).

```
    @classmethod
    def from_url(cls, url):
        response = requests.get(url=url, stream=True)
        response.raw.decode_content = True

        return cls(Image.open(response.raw))

    @classmethod
    def from_array(cls, numpy):
        return cls(Image.fromarray(numpy))
```

Figure 4.1: Loading an image into Pillow from an external URL, and from NumPy

In order for per-channel adjustments to be made it was found that using NumPy

(<http://www.numpy.org>) enabled much easier handling of pixel values, as they are treated as a multi-dimensional array with x and y dimensions along with an RGB value dimension. Note that most functions require the colour values to be in the range [0,1] as a floating point number rather than a [0,255] integer. The RGB dimension can be split into 3 different arrays for each channel and processed by increasing the channel values by a certain amount. Furthermore other operations can be completed such as adjusting the colour values using linear interpolation. Basically, this allows for fine tuning of colour values in that lower/darker values may get adjusted differently to higher/brighter values.

In figure 4.2 two of the filters which ended up being implemented are shown in the form of a python class. The classes extend an interface type class called ‘`BaseFilter`’ which contains the method stubs required of the filter. The ‘`apply()`’ method is the actual filtering method, which takes in another custom parameter called ‘`filterable`’. Most filters use the Pillow ‘`Image`’ object contained within that parameter and perform some contrast and brightness adjustments. The ‘Black and White’ filter uses only provided Pillow methods to convert the image into a black and white form of the image, as seen in figure 4.2b. After the adjustments have been made, the separate channels are then merged into a single array, where they can be converted back to a Pillow ‘`Image`’ object and returned as a ‘`Filterable`’ object for further processing.

#### 4.1.2 Web Responding

Once the code had been implemented for adding filters to the images, the next step was to introduce a handler for HTTP API calls, where images can be uploaded to the application and then filtered using the aforementioned code. The Flask library was used to handle the HTTP requests by adding a python method for each URI address which needed to be accessed. It is quite easy to specify using Flask which HTTP method should be used for each URI and there are many built-in methods which handle file uploads to the server, along with restricting which file types can be uploaded. Figure 4.3 shows how a URI

(a) ‘Joker’ filter class

```
class Joker(BaseFilter):
    @staticmethod
    def num():
        return 1

    @staticmethod
    def id():
        return "joker"

    @staticmethod
    def name():
        return "Joker"

    @staticmethod
    def apply(filterable):
        sharp = ImageEnhance.Sharpness(filterable.pillow).enhance(2.0)
        contrast = ImageEnhance.Contrast(sharp).enhance(1.2)

        r, g, b = array_to_rgb(pillow_to_array(contrast))

        final = merge_channels(
            r,
            linear_interpolate(g, BRIGHTER_LOWER),
            increase_channel(b, 0.1))
        return Filterable.from_array(float_to_uint8(final))
```

(b) ‘Black and White’ filter class

```
class BlackAndWhite(BaseFilter):
    @staticmethod
    def num():
        return 0

    @staticmethod
    def id():
        return "black_and_white"

    @staticmethod
    def name():
        return "Black and White"

    @staticmethod
    def apply(filterable):
        black_white = ImageEnhance.Color(filterable.pillow).enhance(0.0)
        contrast = ImageEnhance.Contrast(black_white).enhance(1.3)
        return Filterable(contrast)
```

Figure 4.2: Python filter classes

structure can be defined using an annotation above a regular Python function.

```
@application.route('/images', methods=["POST"])
def upload():
    request_uid = create_uuid()
    print(str(request_uid))

    try:
        validate_upload(request)
    except Exception:
        return jsonify({_uid:request_uid, error="Invalid file"}, status.HTTP_400_BAD_REQUEST)

    original_uploaded_url = upload_original(request.files["user_file"])

    imid = create_uuid().hex[:5]

    do_filter(imid, original_uploaded_url, request_uid)

    return jsonify(im_url=BASE_URL + "/images/" + imid, 201, {'location': "/images/" + imid})
```

Figure 4.3: Flask image upload API method

Due to Python being able to easily handle both image processing and web requests, it was relatively easy to link both of these functions together. Once the URI routes had been set up all that was required to link them was creating a function which loops through each of the filter classes, passing in the original image each time. This was made possible due to the code design implemented in the previous section. As each filter was a subclass of ‘BaseFilter’, the built-in Python method ‘`__subclasses__()`’ returns any objects which are subclasses of that class. All that is then required is to loop through the returned list and run the ‘`apply()`’ method on each of them.

Once the filters have been processed then it is possible to create thumbnails and previews of the resulting images using the methods on the ‘Filterable’ object mentioned previously. These thumbnails and previews will need to be uploaded somewhere in order to provide a link which a user can access to retrieve them. Amazon S3 (<https://aws.amazon.com/s3/>) was used as a remote hosting site due to its provided Python libraries which handle uploads to the service. For the images to be uploaded they must first be converted into a format such as a JPEG. To do this the built-in ‘BytesIO’ class was used along with the Pillow ‘save()’ method which writes the in-memory image array to a binary format.

```

data = {
    'original': {
        'id': 'original',
        'name': 'Original',
        'full_url': original_uploaded_url,
        'thumb_url': original_thumb,
        'preview_url': original_preview,
    },
    'filtered': {
        'all': filtered_images
    },
    'properties': {
        'vibrant_colour': vibrant,
        'dominant_colours': dominant,
        'vision_api': api_response,
        'size': {
            'full': {
                'width': o_width,
                'height': o_height
            },
            'preview': {
                'width': p_width,
                'height': p_height
            },
            'thumb': {
                'width': t_width,
                'height': t_height
            }
        }
    },
    'timestamp': datetime.datetime.now().timestamp(),
    '_uuid': str(request_uid),
    'imid': imid
}

```

Figure 4.4: Python syntax for declaring a ‘Dict’ object

Due to the syntax of Python it was very easy to define the JSON responses for each of the web methods. The Python ‘Dict’ object can be written in a very similar syntax to JSON and can be seen in figure 4.4. Further to this, the Flask library offers a JSON serializer method called ‘jsonify()’ which automatically converts its parameters into a JSON HTTP response.

Finally, the REST constraints for the application were able to be met by using ap-

propriate HTTP response codes for the API responses. An example of this is seen at the bottom of figure 4.3 where the HTTP code ‘201 CREATED’ is returned along with the URL for the newly created resource.

### 4.1.3 Image Analysis

The two main methods for performing image analysis are using the Google Vision API and performing a K-Means colour clustering operation on the image to find the most prevailing colours within it. The K-Means algorithm is already implemented in scikit-learn therefore there was no need to implement it manually, which would have just introduced unnecessary work that goes against the ‘Eliminate waste’ principle of LSD (Poppendieck & Poppendieck 2003).

A value of 5 clusters was used in this instance of K-Means, which seemed to give a decent range of colours within the image without taking too much processing time. Also only a 100x100 sized version of the image is input into the algorithm to further ensure a fast enough response time for the API, at the risk of losing some colour detail. The algorithm requires a NumPy array, which was easy to supply by converting the Pillow image thumbnail to a NumPy array and re-shaping the array to the ideal shape of 2 dimensions.

After the K-Means analysis some further analysis is implemented on top of this in order to find the (subjectively) most vibrant colour out of the 5 colours found by K-Means. By using the HSV colour space (rather than RGB) which provides values of saturation and brightness it was easy to order the colours by the product of these two values, and then simply selecting the top item from the ordered list which gives arguably the most vibrant colour. This information could serve to be valuable for the recommendation algorithm as it could be possible to find a link between different vibrant colours and which filter a user chooses to apply.

The next step of including a response from the Google Vision API is relatively simple.

Using the API documentation by Google (2018) as a guide, a request can be made in which an image URL is supplied along with the type of analysis to be conducted. Once again this request is made in JSON format which is easy to do in Python. In this case the ‘FACE\_DETECTION’ identifier is sent to the API and the response contains various information about any faces detected in the image. The detail from that response is not used however, as the only information which is desirable in this case is the number of faces in the image which can be found by adding up the number of objects in the ‘faceAnnotations’ array within the JSON response.

#### 4.1.4 Recommendation

Using the data provided from the image analysis components described in the previous sections, the recommendation algorithm has been implemented with relative success. The Multi-Layer Perceptron is also implemented by scikit-learn therefore it made sense to use this implementation rather than create one manually. In order for the MLP to train itself on the data, the data needs to be in the form of a 2D array with normalised values along with a 1D array containing the corresponding classifications for each row in the first array. In order to provide such data a series of functions were implemented which convert the stored JSON data about the images (containing the image analysis results) into an array of raw values. More specifically, these raw values correspond to HSV colour values for the dominant colours, their proportional percentage values, and the number of faces in the image.

The MLP is required to be trained manually by collecting all of the previous responses and formatting them in bulk to be supplied to the algorithm. With this in mind 10 users were asked to use a prototype version of the application and upload as many photos as possible, choosing the filter which they found most appropriate for each image while their choice was logged automatically, though completely anonymously. As a result of this 472 instances were then able to be used to train the MLP using a default train/test split of

75/25% respectively. The structure used for the MLP was 3 hidden layers, each with 30 neurons, and a maximum training time of 1000 epochs.

The results from the MLP were relatively poor, as the results after training multiple times appeared to differ greatly. Furthermore the average classification accuracy did not go above 40% in any test. This could be due to a relatively low number of training instances, as well as the data having a poor link between input data and associated classifications.

The trained model used in the system was selected due to its comparatively high average classification accuracy of 37%, and was saved by serialising the MLP Python object using the Pickle library (<https://docs.python.org/3/library/pickle.html>). This model is then able to be loaded into memory when the application starts, in order to be used for classifying new instances supplied by the API code.

#### 4.1.5 Database

The MongoDB database was used for a few different purposes within the implementation. The first is simply to store JSON responses so that when a user accesses an image endpoint for example like '/images/1234', it means that the response can be accessed multiple times with the heavy processing only having to be done once. The second use was to store the results from the image analysis stage so that the results could be used later for input into the recommendation algorithm implemented in section 4.1.4. The database performed well at this task and using a MongoDB turned out to be very beneficial due to its compatibility with JSON. As JSON was utilised heavily within the implementation and storing everything in it meant virtually no explicit conversion between data structures had to be done (other than going into the recommendation algorithm - as described in the previous section).

## 4.2 Front-End

The front-end web app implementation was relatively simple in terms of complexity however there was a decent learning curve to using React. Having never used a framework/library like this before it took a while to get used to the component-based architecture and how information should flow from component to component.



Figure 4.5: The filter selection page of the front-end web app

The front-end is comprised of two major components which are the upload / image display area, and the filter thumbnail bar across the bottom of the page. A lot of effort was put into making the site being able to scale to different screen sizes, which turned out to be more difficult than expected. As seen in figure 4.5 the image in the center of the page fills most of the negative space between the header and footer. In order to make the center image scale proportionally some code had to be implemented which calculates the correct ratio for the image size and automatically calculates the next best size as soon as the window is re-sized. The result of re-sizing the window to a size more comparable to a mobile screen can be seen in figure 4.6.

A library called React Slick (<https://react-slick.neostack.com>) was used to power animations between different images on the front-end filter selection page. The library has many features built-in which were very useful such as being able to control the state

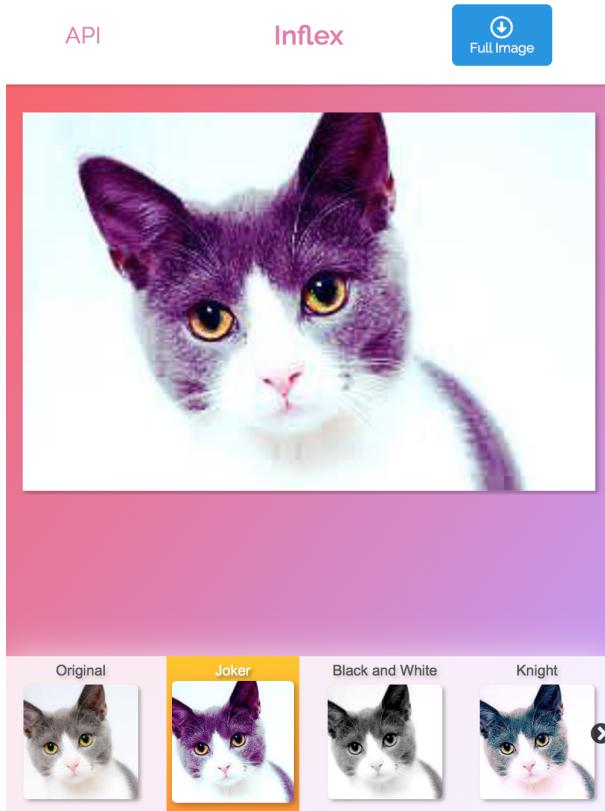


Figure 4.6: The filter selection page - scaled to a mobile screen size

of one image carousel using another - in this case using the thumbnail bar to control the current image in the central area. Both areas are also touch-optimised so on a mobile device it would be possible to quickly swipe between different filters.

In the top right of figures 4.5 & 4.6 a blue button can be seen which is the one that a user must click in order for the API to receive a response that a user has chosen to download an image with a certain filter. Care was taken to make the button stand out from the rest of the page to ensure a user would immediately see it and know what its purpose was. In addition to this, in the bottom of figures 4.5 & 4.6 one of the filters has a coloured background which is there to indicate that this is the recommended filter. Also when the user uploads their photo and the filters appear on screen, the recommended filter is automatically shown as the central image.

## 4.3 Deployment

The final step for implementing the product proposed in this report is to deploy it to an external server and connect it to a public domain name. The deployment will be carried out in three different sections, which are the API, the front-end and the MongoDB database.

### 4.3.1 API

In the prototype stages of implementing the API in Python, temporary deployments were run in order to test whether an application developed locally would run as planned on a remote server. This ended up causing a lot of frustration but was also very helpful as it outlined the steps which had to be taken and the configuration which had to be used in the final deployment to ensure it ran successfully. One of the problems was that the application would not start up properly when deployed on a server running Python version 3.6. In response to this a test deployment was tried using Python version 3.4 and it successfully ran the application, and since then development and every deployment has been done using Python 3.4 as the language version.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
Inflex-api-2	i-0cf57e9b5d35709d7	t1.micro	eu-west-1c	running	2/2 checks ...	None

Figure 4.7: API Deployment - AWS EC2 instance status

The deployments on AWS were done using the Elastic Beanstalk product which allows for relatively easy set-up. The specific server type which the application is deployed to is a `t1.micro` server running Python 3.4 on 64bit Amazon Linux version 2.6.4. The server information and running status can be seen in figure 4.7 from the AWS EC2 Dashboard page.

In order to have an API which could be considered as secure, an important way of achieving this would be to have the service running securely over HTTPS. In order

Certificate Details: PositiveSSL <span style="background-color: #2e6b2e; color: white; border-radius: 5px; padding: 2px 5px;">✓ ACTIVE</span>	
for <b>api.inflex.co</b>	<b>COMODO</b> Creating Trust Online®
Validity	Dec 04, 2017 - Dec 04, 2018
Validation Level	Domain Validation (DV)
Certificate Authority's ID	108812757
Namecheap Order ID	31273938
Total Domains	1 domain
	<a href="#">SEE ORDER</a>

Figure 4.8: API Security - Domain SSL certificate information

to achieve this an SSL certificate was purchased from the domain provider NameCheap (<https://www.namecheap.com>). The certificate provided was then uploaded to the AWS application instance in order for it to start serving traffic over HTTPS. Details about the certificate are shown in figure 4.8 and evidence of connecting using this secure protocol is shown in figure 4.9.

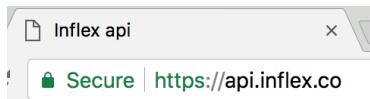


Figure 4.9: API Security - Secure connection to API

### 4.3.2 Front-end

Deploying the front-end was comparatively simple after deploying the API. The React code can be built using the ‘`react-scripts build`’ command, which creates a set of files which can be uploaded to a static file hosting server such as AWS S3. Using S3, it is possible to configure the static host to serve the files like a web host and all that needs to be done after that is to point the domain at the static host. As the React code is all based in JavaScript, the resulting front-end is processed entirely on the client therefore a dedicated web service is not required for hosting it.

Unfortunately, an HTTPS configuration was not achieved on the front-end deploy-

ment. However, this is not a huge problem due to the fact that all front-end HTTP calls are made on the client through AJAX, and every call to the API is done over HTTPS as shown in the previous section. This means that any file uploaded to the server is still protected however when downloading the static JavaScript files for the front-end the connection is not secure.

### 4.3.3 Database

Throughout most of the development stages of the application, a local instance of MongoDB was run for testing the application's interactions to the database. Once it was time to deploy a database to an external server an initial effort was made over a week or so to manually set up an instance of MongoDB on an AWS server. Unfortunately the time spent on this ended up being wasted as after many attempts it was still not possible to connect to the MongoDB instance from the API server. After this, it was discovered that MongoDB themselves offer a service called MongoDB Atlas (<https://www.mongodb.com/cloud/atlas>) which provides a free instance of cloud-hosted MongoDB (coincidentally hosted on AWS).

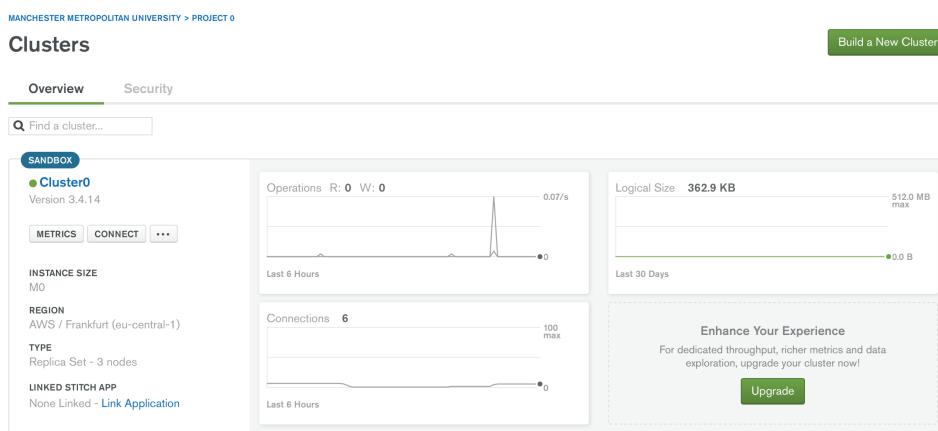


Figure 4.10: MongoDB Atlas - Deployed cluster dashboard

The instance provided by MongoDB Atlas includes a primary server and two replica servers which handle back-ups of data. The dashboard information about the deployed

instance used in the application is shown in figure 4.10. Once the database service was up and running it was then trivial to change the connection details in the API code to then point at the new service instead of the local service. No further set up was needed to start using the service due to the fact that MongoDB has no constraints on what document structures can be stored on the database. If for example an SQL server was used it would have required running lots of SQL scripts to make sure the server was configured correctly before any data could be inputted into it.

## 4.4 Testing

In order to fully evaluate the success of the recommendation system and the front-end web app, a short questionnaire was given to a set of 5 participants after given the chance to use the final implementation of the product. These questionnaires can be found in appendix E.

The main factor in determining an accurate recommendation for a user is simply to find out if they would have chosen that recommended filter, whether or not it was not shown to them as recommended. The results from this question in the questionnaire are summarised in table 4.1. It can be seen that only 2 out of 5 participants thought the recommendation was accurate.

Participant Number	Accurate Recommendation
1	No
2	Yes
3	No
4	No
5	Yes

Table 4.1: Participant responses to recommendation accuracy

Due to time constraints a full usability questionnaire could not be given out, however one question was included in the questionnaire which asked users to rate their experience

of how simple they found the UI to navigate. These results can be seen in table 4.2. Note that the scores are in the range [1,5] where a score of 1 indicates a poor user experience and 5 indicates an excellent user experience. It can be seen that most participants found it to be quite usable, with the most popular score being 4 out of 5.

Participant Number	UI Simplicity Score
1	2
2	4
3	4
4	3
5	4

Table 4.2: Participant responses to UI simplicity

Due to time constraints no white and black box testing could be carried out, however throughout development a focus was kept on ensuring the application was reliable in different use cases. For example when re-scaling the window size on the front-end, the code is able to handle re-sizing the components on the page to the correct proportional sizes.

# **Chapter 5**

## **Evaluation**

Within this report there were multiple aims which were set out to be achieved. These aims covered a variety of subjects such as computer vision for image analysis and following best practices for developing an API and Web App. The main of these requirements were developing the API, developing the Web front-end, developing a system for filtering images, and most importantly developing a recommender system for those filters. The system proposed and implemented in chapter 4 meets most of these goals well, and the evaluation of the product will consist of how well the original objectives set out in section 1.4 were met. The objectives and their evaluations are set out below.

**Objective 1 - Implement a simple online web application for filtering images using a framework or library designed for front-end development across multiple platforms.**

The front-end proposed in this report is very effective in its use as an interface on top of the API. It is very lightweight and simple however it is also very functional and robust. The component structure of React allows for very easy management of application flow, with variables and callback functions being passed to child components the application is easily able to deal with calling an external API and passing that information to components which automatically update upon receiving that information. However, the requirement for the web app working across multiple platforms was not met. During development the front-end was run purely on a few different desktop browsers such as Safari and Chrome which worked fine, however after finally deploying it to a public URL and accessing it from a mobile device it was found that no response came back to the device once an image was uploaded for filtering. Due to the nature of mobile devices there was no easy way to troubleshoot what was going on, where on a desktop device this could easily be done by debugging it in an IDE. Finally, there was not enough time after deploying the application to do an investigation into this issue therefore it can only be viewed as a partial failure to meet one of the objectives.

**Objective 2 - Implement a back-end application which is able to process images including applying filters and creating thumbnails.**

The python back-end proposed in this report is highly capable of processing images due to the libraries available and its ability to connect easily to the API code. The libraries NumPy and Pillow were both instrumental to the success of image processing, however there was a decent learning curve to both of these along with python itself. This slightly limited how much processing was able to be done on the images as much more time had to be spent researching what libraries are available and what the best practices are for

processing images in python.

**Objective 3 - Research various recommender systems to determine the common technologies/solutions for solving the problem of recommendations.**

In section 2.2 various examples of recommender systems were found and a general consensus was discovered that machine learning tools are a very useful tool in recommender systems. While there was not a single blanket solution, and the solutions available are often still not very effective - facing issues such as the cold-start problem which is still a popular research topic. Deep learning was discovered to be a very common tool in industry however this was not feasible due to performance constraints and that they did not seem to be fully applicable to the problem of recommending filters.

**Objective 4 - Implement a recommender system which is able to suggest photo filters to users based on image content.**

Within this objective the strict definition was met, as a recommender system was built which is able to recommend filters based on image content. However, the accuracy of the AI system which was implemented was not very high, and often came up with highly inconsistent values when producing classification accuracy reports. When using a trained model with a higher trained accuracy of around 37% on average per class, the model did however seem to have some predictability in how it would recommend filters such as when the photo included one or more faces. The low accuracy would seem to suggest that the method for training the model was not ideal in that users may have had differing tastes for which filters to apply. Furthermore the recommender would occasionally suggest different recommendations when the same photograph was uploaded twice. This is likely a combination of the K-Means analysis producing slightly different results each time and the classifiers not having a strong enough trained model. Finally the results from section 4.4 showed that users generally did not find the recommendation to be accurate, with

only 2 out of 5 saying they found the recommendation to be accurate.

**Objective 5 - Implement a RESTful API which exposes both the recommender system and the filtering system to the front-end web app and allows developers to connect them into their own applications.**

This objective was met easily with the python back-end using the Flask library. This library included helper methods for handling RESTful API's and allowed for doing things such as responding with an 'HTTP 201 CREATED' response when a resource was uploaded to the site via an 'HTTP POST' request. The URI structure was also within REST constraints, with images being uploaded to the '/images/' URL via an 'HTTP POST' request, then being accessed using an 'HTTP GET' request from that URL such as '/images/98119-black\_and\_white.jpg'. The URI structure was relatively simple due to the simplicity of the request being made therefore it was easier to develop. The API was then easily connected to the front-end and showed that it was possible to integrate the API into a client application.

# Chapter 6

## Conclusion

This report has outlined a system which can be used to filter photographs by accessing a REST API, or by connecting to an accompanying web app using a desktop web browser. The API and web app are able to give a recommendation of which filter should be chosen based on the content of the image, however this recommendation was not proven to be very accurate. The code solution is simple and includes many carefully chosen libraries to handle difficult tasks, rather than implementing them manually which creates unnecessary work and introduces too much complexity into the product. Due to time constraints some issues were left unsolved such as not having HTTPS on the front-end server and the front-end not working on mobile devices.

## 6.1 Further work

If offered more time to extend the functionality of the product, initially the existing problems mentioned in the preceding paragraph could be addressed. Further to solving those issues, as a result of the low accuracy from the recommendation algorithm a more advanced recommender algorithm could be implemented. This could include developing a user system which is able to track individual user history for applying photo filters and have that data be used to provide more personalised recommendations for users. Additionally, background processing could be introduced where uploaded images are not analysed and filtered synchronously but are done at the same time in order to reduce response times.

## 6.2 Author reflection

When starting this project I set out a personal goal to learn a few different technologies such as Python, React, and AWS. I do feel like I was able to do this and also create a product which serves its purpose relatively well. However, I think the biggest hindrance throughout the project was that every time I started work on a different part of the product implementation I would have to start from absolute scratch, spending more time learning the basics than implementing desired features. This lead to some gaps in quality of code design, and features which never made it into the final product.

Overall, I have enjoyed the process of completing this project and it feels like the true summation of all the work I have put in over my whole university career. I hope to utilise lessons learned from this project throughout my later life in both programming and real life.

# References

- Adomavicius, G. & Tuzhilin, A. (2005), ‘Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions’, *IEEE transactions on knowledge and data engineering* **17**(6), 734–749.
- Beaver, D., Kumar, S., Li, H. C., Sobel, J., Vajgel, P. et al. (2010), Finding a Needle in Haystack: Facebook’s Photo Storage., in ‘OSDI’, Vol. 10, pp. 1–8.
- Caicedo, J. C., Kapoor, A. & Kang, S. B. (2011), Collaborative personalization of image enhancement, in ‘Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on’, IEEE, pp. 249–256.
- Cattell, R. (2011), ‘Scalable SQL and NoSQL data stores’, *Acm Sigmod Record* **39**(4), 12–27.
- Ciregan, D., Meier, U. & Schmidhuber, J. (2012), Multi-column deep neural networks for image classification, in ‘Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on’, IEEE, pp. 3642–3649.
- Coates, A., Ng, A. & Lee, H. (2011), An analysis of single-layer networks in unsupervised feature learning, in ‘Proceedings of the fourteenth international conference on artificial intelligence and statistics’, pp. 215–223.
- Coleman, G. B. & Andrews, H. C. (1979), ‘Image segmentation by clustering’, *Proceedings of the IEEE* **67**(5), 773–785.
- Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. et al. (2010), The YouTube video recommendation system, in ‘Proceedings of the fourth ACM conference on Recommender systems’, ACM, pp. 293–296.
- Duval, E., Hodgins, W., Sutton, S. & Weibel, S. L. (2002), ‘Metadata principles and practicalities’, *D-lib Magazine* **8**(4), 1082–9873.
- Fielding, R. T. & Taylor, R. N. (2000), *Architectural styles and the design of network-based software architectures*, University of California, Irvine Doctoral dissertation.
- Gomez-Uribe, C. A. & Hunt, N. (2016), ‘The netflix recommender system: Algorithms, business value, and innovation’, *ACM Transactions on Management Information Systems (TMIS)* **6**(4), 13.

- Google (2015), ‘Google Cloud Vision API changes the way applications understand images’, <https://cloudplatform.googleblog.com/2015/12/Google-Cloud-Vision-API-changes-the-way-applications-understand-images.html>. Viewed on 10-01-2018.
- Google (2018), ‘Cloud Vision API | Google Cloud’, <https://cloud.google.com/vision/docs/reference/rest/>. Viewed on 17-01-2018.
- Graziotin, D. & Abrahamsson, P. (2013), Making sense out of a jungle of JavaScript frameworks, in ‘International Conference on Product Focused Software Process Improvement’, Springer, pp. 334–337.
- Guyon, I. & Elisseeff, A. (2006), ‘An introduction to feature extraction’, *Feature extraction* pp. 1–25.
- Han, J., Haihong, E., Le, G. & Du, J. (2011), Survey on NoSQL database, in ‘Pervasive computing and applications (ICPCA), 2011 6th international conference on’, IEEE, pp. 363–366.
- Hu, Y., Manikonda, L., Kambhampati, S. et al. (2014), What We Instagram: A First Analysis of Instagram Photo Content and User Types., in ‘Icws’.
- Instagram (2017), ‘700 million’, <https://instagram-press.com/blog/2017/04/26/700-million/>. Viewed on 9-11-2017.
- Jannach, D., Zanker, M., Felfernig, A. & Friedrich, G. (2010), *Recommender systems: an introduction*, Cambridge University Press.
- Johnson, C. (2015), ‘From Idea to Execution: Spotify’s Discover Weekly’, <https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly/>. Viewed on 10-11-2017.
- Kang, S. B., Kapoor, A. & Lischinski, D. (2010), Personalization of image enhancement, in ‘Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on’, IEEE, pp. 1799–1806.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in ‘Advances in neural information processing systems’, pp. 1097–1105.
- Lee, H., Pham, P., Largman, Y. & Ng, A. Y. (2009), Unsupervised feature learning for audio classification using convolutional deep belief networks, in ‘Advances in neural information processing systems’, pp. 1096–1104.
- Linden, G., Smith, B. & York, J. (2003), ‘Amazon. com recommendations: Item-to-item collaborative filtering’, *IEEE Internet computing* 7(1), 76–80.
- Lops, P., De Gemmis, M. & Semeraro, G. (2011), Content-based recommender systems: State of the art and trends, in ‘Recommender systems handbook’, Springer, pp. 73–105.

MacQueen, J. et al. (1967), Some methods for classification and analysis of multivariate observations, in ‘Proceedings of the fifth Berkeley symposium on mathematical statistics and probability’, Vol. 1, Oakland, CA, USA, pp. 281–297.

Minsky, M., Papert, S. A. & Bottou, L. (2017), *Perceptrons: An introduction to computational geometry*, MIT press.

Mobahi, H., Collobert, R. & Weston, J. (2009), Deep learning from temporal coherence in video, in ‘Proceedings of the 26th Annual International Conference on Machine Learning’, ACM, pp. 737–744.

Neuhaus, J. (2017), ‘Angular vs. React vs. Vue: A 2017 comparison’, <https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176>. Viewed on 29-01-2018.

Pixabay (2013), ‘Grey and white short fur cat · free stock photo’, <https://pixabay.com/en/cat-pet-animal-domestic-fur-1151519/>. Viewed on 05-02-2018.

Poppendieck, M. & Poppendieck, T. (2003), *Lean software development: an agile toolkit*, Addison-Wesley.

Popper, B. (2015), ‘Tastemaker: How Spotify’s Discover Weekly cracked human curation at internet scale’, <https://www.theverge.com/2015/9/30/9416579/spotify-discover-weekly-online-music-curation-interview>. Viewed on 11-11-2017.

Ricci, F., Rokach, L. & Shapira, B. (2011), Introduction to recommender systems handbook, in ‘Recommender systems handbook’, Springer, pp. 1–35.

Sanderson, M. & Croft, B. (1999), Deriving concept hierarchies from text, in ‘Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval’, ACM, pp. 206–213.

Schein, A. I., Popescul, A., Ungar, L. H. & Pennock, D. M. (2002), Methods and metrics for cold-start recommendations, in ‘Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval’, ACM, pp. 253–260.

Shi, J. & Malik, J. (2000), ‘Normalized cuts and image segmentation’, *IEEE Transactions on pattern analysis and machine intelligence* **22**(8), 888–905.

VicMan LLC (2017), ‘askAI - Make your photo more likable’, <https://itunes.apple.com/gb/app/askai-for-instagram-more-likes-on-photo-on-instaid1253856702?mt=8&ign-mpt=u0%3D4>. Viewed on 10-02-2018.

Yee, K.-P., Swearingen, K., Li, K. & Hearst, M. (2003), Faceted metadata for image search and browsing, in ‘Proceedings of the SIGCHI conference on Human factors in computing systems’, ACM, pp. 401–408.

# Appendices

# **Appendix A**

## **Terms of Reference**

Please note that the project title, aims, and objectives have changed since the following terms of reference was submitted. The course-specific learning outcomes have not changed however. The updated aims and objectives can be seen in sections 1.3 and 1.4 respectively.

A web-based game: Humans vs Algorithms for image classification

# *A web-based game: Humans vs Algorithms for image classification*

*Terms of Reference*

**Student:** Joshua Bridge  
**Supervisor:** John Darby

## A web-based game: Humans vs Algorithms for image classification

1.1 Bsc (Hons) Software Engineering Learning Outcomes.....	2
1.2 Project Background.....	2
1.3 Aims.....	3
1.4 Objectives .....	3
1.5 Problems .....	4
1.6 Timetable and Deliverables .....	4
1.7 Required Resources: .....	4
1.8 References.....	5
1.9 Bibliography .....	5

### **1.1 Bsc (Hons) Software Engineering Learning Outcomes**

- Demonstrate an understanding of the principles of object orientation in the context of analysis and design.
- Apply project management techniques in order to plan, monitor and control a project.
- Explain and utilise fundamental object orientation concepts such as classes, encapsulation, inheritance and polymorphism and relate them to their practical situations including library and graphical user interface (GUI) development.
- Utilise in-depth, practical experience of the types of software tools that can support an object-oriented software lifecycle and develop this through practical experience.
- Utilise and understand methods and appropriate software tools for software development, including Software Testing Tools, Version Control and Project Management.
- Demonstrate an understanding of the fundamental, basic issues of software testing.
- Evaluate and apply design patterns for the development of high-quality, object-oriented software systems.
- Build robust, secure distributed systems using techniques such as messaging, persistent storage, remote methods and components.
- Have a range of programming skills to apply in the software engineering environment.
- Demonstrate that they can participate in and complete a substantial project, involving research, planning, specifying, designing, building and testing software, integrating knowledge gained from the core units on the award.

**(MMU, 2017)**

### **1.2 Project Background**

Image classification in static images is one of the most popular subjects of research within the computer vision field. It involves letting a computer program look at an image and let it try to guess what it contains. Its applications are far-reaching including areas such as filtering images via content (**Vailaya et al., 2001**), which could be used in web image search, or searching through local photo libraries. While much research has already been done into the area, it is still fast developing and only recently have major consumer applications started to make use of image classification. For example, ‘Quick, draw!’ is an online game which prompts a user to draw a certain object on the screen, then using image classification techniques it tries to guess what a player has drawn (**Quickdraw.withgoogle.com, 2017**). While the task of recognising a certain object within an image is generally trivial for a human

## A web-based game: Humans vs Algorithms for image classification

mind, a computer doing the same thing is most certainly non-trivial. Teaching a computer to recognise certain objects is notably similar to teaching a baby to apply names to certain objects, except in this case you must first tell the baby how to learn. To teach a computer how to recognise an object using machine learning, a huge dataset is initially needed to ‘train’ the algorithm about what represents a certain object.

This project will attempt to build a web-based game, where a player will race against one or more types of image classification methods to guess the content of an image. The game should implement some kind of progress tracking, where the player can judge their performance history. This could be done using some kind of database software, to keep a history of the users scores, possibly linked to a username in permeant storage on a server, or just in temporary storage on the local machine. Furthermore, some kind of difficulty system should be implemented, so as the player progresses through the game, it should get more challenging as time goes on. The challenge of difficulty is complicated as it will entirely depend on how good the image classification method is. If the algorithm can guess the images consistently more accurately than a human, which has been possible in a few types of algorithms (**He et al., 2015**) then difficulty could be introduced by implementing a handicap for the algorithm to begin with, and slowly removing it. On the other hand, if the player is consistently better than the algorithm, then a handicap would need to be slowly introduced for the player as they progress. The requirement of a handicap and what kind of handicap is needed depends on the image classification algorithm used, so research will be needed to find the best algorithm for the fairest gameplay. If there is time, more than one algorithm could be implemented as another type of difficulty tier. An assessment will need to be made of each algorithm’s ability to run on a local machine (more specifically, in a browser), and their viability for competing with a player. For algorithms that will run locally, a dataset may be needed to ‘train’ the algorithm of image categories. A widely available and non-copyrighted set such as Caltech 101 (**Li Fei-Fei, Fergus and Perona, 2006**) could be used as this set has been used as a benchmark for many classification algorithms, and has plenty of data available for comparisons of algorithm success rates etc.

### 1.3 Aims

1. Develop a web-based game with user progress tracking, and multiple difficulty tiers
2. Explore the differences between humans and algorithms when trying to recognise objects in images

### 1.4 Objectives

1. Review existing literature & assess viable options of image classification algorithms
2. Research languages best-suited for a web-based game
3. Research UI frameworks and back-end frameworks for use in a web-based game
4. Design GUI mock-up highlighting the main areas of the application and how it is controlled
5. Create prototype game with rudimentary UI, to test and develop gameplay mechanics (such as difficulty)
6. Implement mock-up GUI design
7. Perform analysis of win/loss rates and implement difficulty levels
8. Document progress, findings and procedure in report

## A web-based game: Humans vs Algorithms for image classification

### 1.5 Problems

One of the main problems I will face is that many modern methods of image classification often require significant processing power or time to run. Furthermore, if the processing is to happen locally, then the speed of recognition may differ from device to device. One way of negating this would be to use an online service to perform the processing, thereby eliminating the bias per-device. There also may also be a cost involved if high processing power is needed. If the algorithm was to run locally, then possibly some kind of minimum processor requirements could be implemented, or a limiter on speed of classification, which could allow all devices to run at the same rate of recognition.

Another problem that could arise is the automation of the difficulty tiers. If the difficulty system is implemented using some kind of handicap, then that could require doing something like applying an image filter to some of the images to make them more difficult to recognise. Applying an image filter is something that can be easily done using many coding languages, and can generally be done in real-time, depending on the filter. However, automating this in a way that predictably affects difficulty is not an easy task, so research will be needed to find the most predictable handicap that can be automated and put into the code.

### 1.6 Timetable and Deliverables

#### 1.6.1 Timetable

Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
<b>Research</b>																										
Review existing literature on image classification methods	■	■	■																							
Look for frameworks and languages best-suited for a web-based game		■	■	■																						
Compare viability of classification methods for use in a web-based game				■	■	■																				
<b>Develop</b>										■	■	■	■	■												
Design GUI mock-up										■	■	■	■	■												
Create rudimentary prototype game including basic difficulty tiers										■	■	■	■	■												
Implement GUI design into prototype										■	■	■	■	■												
Test and tweak difficulty levels										■	■	■	■	■												
<b>Evaluate</b>																										
Perform analysis of win/loss rate for player	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
Document progress, findings and procedure in report	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	

#### 1.6.2 Deliverables

1. A prototype game
2. An interim report
3. A final project report
4. A final game
5. A demo presentation

### 1.7 Required Resources:

Text Editor  
Web Browser  
Development Computers  
GitHub  
Git  
MySQL

A web-based game: Humans vs Algorithms for image classification

### 1.8 References

- Fei-Fei, L., Fergus, R. and Perona, P. (2006) ‘One-shot learning of object categories’. *IEEE transactions on pattern analysis and machine intelligence*, 28(4), pp.594-611.
- He, K., Zhang, X., Ren, S. and Sun, J. (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).
- Photos.google.com. (2017) *Google Photos - All your photos organized and easy to find*. [Online] [Accessed on 10 Oct. 2017] <https://photos.google.com>.
- Quickdraw.withgoogle.com. (2017) ‘*Quick, Draw!*’. [Online] [Accessed on 21 Jun. 2017]. <https://quickdraw.withgoogle.com/>.
- Vailaya, A., Figueiredo, M.A., Jain, A.K. and Zhang, H.J. (2001) ‘Image classification for content-based indexing’. *IEEE transactions on image processing*, 10(1), pp.117-130.

### 1.9 Bibliography

- Ciregan, D., Meier, U. and Schmidhuber, J. (2012, June) ‘Multi-column deep neural networks for image classification’. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 3642-3649). IEEE.
- Lu, D. and Weng, Q. (2007) ‘A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5), pp.823-870.
- Johnson, R. (2015) *Microsoft, Google Beat Humans at Image Recognition | EE Times*. EETimes. [Online] [Accessed on 20 October 2017] [https://www.eetimes.com/document.asp?doc\\_id=1325712](https://www.eetimes.com/document.asp?doc_id=1325712).
- Karpathy, A. (2014) *What I learned from competing against a ConvNet on ImageNet*. Karpathy.github.io. [Online] [Accessed on 20 October 2017] <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>.
- Nowak, E., Jurie, F. and Triggs, B. (2006) ‘Sampling strategies for bag-of-features image classification’. *Computer Vision-ECCV 2006*, pp.490-503.



# **Appendix B**

## **Ethics documents**

Again please note that while the exact title and project activities have changed since the original submission of the ethics documents, none of the ethical requirements have changed. The updated project title can be found on the cover of this report, and the abstract near the start of the report contains an updated description of the project activities.



Manchester  
Metropolitan  
University

## ETHICS CHECKLIST

This checklist must be completed **before** commencement of **any** research project. This includes projects undertaken by **staff and by students as part of a UG, PGT or PGR programme**. Please attach a Risk Assessment.

Please also refer to the [University's Academic Ethics Procedures; Standard Operating Procedures](#) and the [University's Guidelines on Good Research Practice](#)

Full name and title of applicant:	Mr Joshua Michael Ephraim Bridge	
University Telephone Number:	07957585934	
University Email address:	14032908@stu.mmu.ac.uk	
Status:	<input checked="" type="checkbox"/> Undergraduate Student <input type="checkbox"/> Postgraduate Student: Taught <input type="checkbox"/> Postgraduate Student: Research <input type="checkbox"/> Staff	
Department/School/Other Unit:	Computing and Digital Technology	
Programme of study (if applicable):	Software Engineering (Hons)	
Name of DoS/Supervisor/Line manager:	Dr. John Darby	
Project Title:	A web-based game: Humans vs Algorithms for image classification	
Start & End date (cannot be retrospective):	22/09/17 - 28/04/18	
Number of participants (if applicable):	10	
Funding Source:		
<b>Brief description of research project activities (300 words max):</b>  This project will aim to create a web-based game where a player can compete against an algorithm to guess the content of an image. The game should feature multiple difficulty levels, and a way to track player progress throughout the game.		
	YES	NO
Does the project involve NHS patients or resources?  If 'yes' please note that your project may need NHS National Research Ethics Service (NRES) approval. Be aware that research carried out in a NHS trust also requires governance approval.  Click <a href="#">here</a> to find out if your research requires NRES approval Click <a href="#">here</a> to visit the National Research Ethics Service website To find out more about Governance Approval in the NHS click <a href="#">here</a>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Does the project require NRES approval?  If yes, has approval been granted by NRES? Attach copy of letter of approval. Approval cannot be granted without a copy of the letter.	<input type="checkbox"/>	<input checked="" type="checkbox"/>

<b>NB Question 2 should only be answered if you have answered YES to Question 1. All other questions are mandatory.</b>	<b>YES</b>	<b>NO</b>
1. Are you gathering data from people?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
For information on why you need informed consent from your participants please click <a href="#">here</a>		
2. If you are gathering data from people, have you:	<input type="checkbox"/>	<input type="checkbox"/>
a. attached a participant information sheet explaining your approach to their involvement in your research and maintaining confidentiality of their data?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
b. attached a consent form? (not required for questionnaires)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to see an example of a <a href="#">participant information sheet</a> and <a href="#">consent form</a>		
3. Are you gathering data from secondary sources such as websites, archive material, and research datasets?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Click <a href="#">here</a> to find out what ethical issues may exist with secondary data		
4. Have you read the <a href="#">guidance</a> on data protection issues?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
a. Have you considered and addressed data protection issues – relating to storing and disposing of data?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
b. Is this in an auditable form? (can you trace use of the data from collection to disposal)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5. Have you read the <a href="#">guidance</a> on appropriate research and consent procedures for participants who may be perceived to be vulnerable?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
a. Does your study involve participants who are particularly vulnerable or unable to give informed consent (e.g. children, people with learning disabilities, your own students)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6. Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited (e.g. students at school, members of self-help group, nursing home residents)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click for an example of a PIS and <a href="#">information about gatekeepers</a>		
7. Will the study involve the use of participants' images or sensitive data (e.g. participants personal details stored electronically, image capture techniques)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click <a href="#">here</a> for guidance on images and sensitive data		
8. Will the study involve discussion of sensitive topics (e.g. sexual activity, drug use)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here for an advisory distress protocol		
9. Could the study induce psychological stress or anxiety in participants or those associated with the research, however unlikely you think that risk is?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read about how to deal with stress and anxiety caused by research procedures		
10. Will blood or tissue samples be obtained from participants?	<input type="checkbox"/>	<input type="checkbox"/>
Click here to read how the Human Tissue Act might affect your work		
11. Is your research governed by the Ionising Radiation (Medical Exposure) Regulations (IRMER) 2000?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click <a href="#">here</a> to learn more about IRMER		
12. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read about how participants need to be warned of potential risks in this kind of research		
13. Is pain or more than mild discomfort likely to result from the study? Please attach the pain assessment tool you will be using.	<input type="checkbox"/>	<input checked="" type="checkbox"/>

<a href="#">Click here to read how participants need to be warned of pain or mild discomfort resulting from the study and what do about it.</a>		
14. Will the study involve prolonged or repetitive testing or does it include a physical intervention?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">Click here to discover what constitutes a physical intervention</a> and <a href="#">here to read how any prolonged or repetitive testing needs to managed for participant wellbeing and safety</a>		
15. Will participants to take part in the study without their knowledge and informed consent? If yes, please include a justification.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">Click here to read about situations where research may be carried out without informed consent</a>		
16. Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">Click here to read guidance on payment for participants</a>		
17. Is there an existing relationship between the researcher(s) and the participant(s) that needs to be considered? For instance, a lecturer researching his/her students, or a manager interviewing her/his staff?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">Click here to read guidance on how existing power relationships need to be dealt with in research procedures</a>		
18. Have you undertaken Risk Assessments for each of the procedures that you are undertaking?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
19. Is any of the research activity taking place outside of the UK?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
20. Does your research fit into any of the following security sensitive categories: <ul style="list-style-type: none"> <li>• commissioned by the military</li> <li>• commissioned under an EU security call</li> <li>• involve the acquisition of security clearances</li> <li>• concerns terrorist or extreme groups</li> </ul>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
If Yes, please complete a <a href="#">Security Sensitive Information Form</a>		

I understand that if granted, this approval will apply to the current project protocol and timeframe stated. If there are any changes I will be required to review the ethical consideration(s) and this will include completion of a 'Request for Amendment' form.

- have attached a Risk Assessment  
 have attached an Insurance Checklist

If the applicant has answered **YES** to **ANY** of the questions **5a – 17** then they must complete the [MMU Application for Ethical Approval](#).

Signature of Applicant: Joshua Bridge Digitally signed by Joshua Bridge  
Date: 2017.10.27 15:48:32  
+01'00' Date: 27/10/17 (DD/MM/YY)

**Independent Approval for the above project is (please check the appropriate box):**  
**Granted**

- I confirm that there are no ethical issues requiring further consideration and the project can commence.

**Not Granted**

- I confirm that there are ethical issues requiring further consideration and will refer the project protocol to the Faculty Research Group Officer.

Signature: \_\_\_\_\_ Date: \_\_\_\_\_ (DD/MM/YY)

Print Name: \_\_\_\_\_ Position: \_\_\_\_\_  
**Approver:** Independent Scrutiniser for UG and PG Taught/ PGRs RD1 Scrutiniser/  
 Faculty Head of Ethics for staff.

## Participant Information Sheet

### Recommending photo filters via image content in a Web App & REST API

I am inviting you to take part in a research study. In order to do this, you need to know why this research is being carried out and what your involvement will be. Please take your time to read over this to ensure everything you have read is clear. If something is not clear then please ask for more details. Please carefully consider whether or not you would like to take part.

#### Study Purpose

This educational research is being conducted as part of a Software Engineering degree, and aims to build a web application where users can upload images in order to apply a photo filter to them. The application will aim to recommend the best filter for use on each photograph.

#### Why have I been invited?

Anyone who is able to use a laptop computer is able to participate in this study. There will only be up to 10 people participating in this study.

#### Do I have to take part?

Whether or not you take part is completely up to you. The following sections will explain your involvement in the study, so please read them carefully. You may withdraw from the study at any time, and you do not have to give a reason.

#### What will happen to me if I take part?

This study will run from 22/09/17 to 28/04/18, your involvement may not last the whole duration, but it is possible for it to be required at any point within this period. Your involvement will be limited to (up to) 3 sessions lasting from 10 minutes - 1 hour. All that is required of you will be to use a version of the web app.

#### Expenses and payments

There are no expenses or payments available for this study. The web app will be accessible from the researcher's laptop and will require the researcher's attendance to guide you through the steps required to use the web app.

#### What will I have to do?

You will be asked to use a version of the web app and apply your preferred filters to various images of your choosing. No personal information will be taken from you. You may be asked to do this up to 3 times, for a period of 10 minutes – 1 hour. At the end, you may be asked to answer a very short questionnaire about your experience using the web app.

**What are the possible disadvantages and risks of taking part?**

There should be no disadvantages or discomfort caused by participating in this study. The risk caused from participating is no different to the risks of daily life when using a computer.

**What are the possible benefits of taking part?**

There are no potential benefits of taking part in this study, however your participation should provide data which will be used to further refine the web app's features, upon which you may be asked to use the web app again to check whether the work done has improved the overall usability.

**What if there is a problem?**

If you have any complaints or questions then please contact the researcher at 07957585934.

**Will my taking part in this study be kept confidential?**

Any data collected during this study will be kept securely in accordance with the Data Protection Act 1998. The data provided will be anonymous, hosted on secure platforms, and you will not be identifiable in any publication as a result of this study. The data itself will not be identifiable, as it is only a measure of your statistics in the web app.

**What will happen if I don't carry on with the study?**

If you withdraw from the study we will destroy any identifying information about you in research notes, however the anonymous data collected from you up until your withdrawal will need to be kept for later use in the study.

**What will happen to the results of the research study?**

The results of the study will be collected into a report. You will not be identified in this report.

**Who is organising or sponsoring the research?**

MMU

**Contact Details**

Researcher – 14032908@stu.mmu.ac.uk



## Research Insurance Checklist



### Overview

Manchester Metropolitan University holds insurance policies to cover claims for negligence arising from the conduct of the institution's normal business. This includes research undertaken by undergraduate and postgraduate students as part of their academic qualification as well as research carried out by staff.

If you are an undergraduate student, postgraduate student or staff researcher at the institution, you must complete all relevant sections of the checklist on the following pages to identify whether your application requires referral to the university's Insurance Officer.

Completing and submitting the checklist will ensure that your research study has appropriate insurance cover in place **before** it begins. Please submit your completed Research Insurance Checklist along with your Ethics Checklist and/or Application for Ethical Approval to your Faculty Research Officer.

### Referral to the Insurance Officer

If your research falls into any of the categories listed in Section 2 and/or Section 3 of the checklist, the Faculty Research Officer will send the following information to the Insurance Officer at [insurance1@mmu.ac.uk](mailto:insurance1@mmu.ac.uk):

- Insurance Checklist
- Ethics Checklist and/or Application for Ethical Approval Form
- Participant Information Sheet(s) (if applicable)
- Participant Consent Form(s) (if applicable)
- Risk Assessment

The Insurance Officer will liaise with the insurers to gain approval. Please note some types of research may require additional insurance, which may incur an additional cost to the Faculty.

**Research studies must not commence until insurance and all other relevant authorisations and/or approvals are given.**

### Travel Insurance

Manchester Metropolitan University has a policy to provide worldwide travel insurance for members of staff and students travelling in connection with their course or on an approved University trip. This includes travel undertaken in connection with undertaking a research study. You must complete the online travel insurance form to register for travel insurance and should do this at least two weeks before your departure date.

Please visit the [Financial and Legal webpage](#) for details.

### High Risk Countries

Please visit the [AIG Travel Guard website](#) to identify whether the overall rating for the country you are travelling to is 'High Risk' or more severe. Please contact your Faculty Research Officer for guidance on accessing the relevant information on the website.



## Research Insurance Checklist



### ADMINISTRATIVE DETAILS

Lead Investigator Name **Mr. Joshua Bridge**  
(Title/Forename/Surname)

Contact Email Address **14032908@stu.mmu.ac.uk**

Full Title of the Research **A web-based game: Humans vs Algorithms for image classification**

### SECTION 1 – TECHNIQUES, TESTING AND INTERVENTIONS

Does your research study involve:

**Physically invasive techniques?**

This refers to any test in which the skin of the participant is broken or an implement is inserted into any opening of the human body (e.g. eyes, ears, nose, mouth, lungs, stomach, rectum, vagina and urethra) or involves the taking of body samples such as saliva, hair, urine, faeces, sputum, skin, nails, or taking biopsies of any form for any purpose, or any form of scanning such as DEXA scans, Ultrasound scans, MRI, fMRI, CT, or PET scanning.

**Ingestion of food stuffs or drugs?**

This refers to the consumption of any substance which may impact on psychological or physical state. Substances may include but are not limited to food, beverages or drugs.

**Physical testing?**

This refers to any test in which a participant must perform an action resulting in the use of any muscle of the body and/or involves the use of scanning procedures, eye-trackers, mounted body cameras, sensors or electrodes, or the taking of swabs from any cavity of the body, respiratory challenge testing or recording of peak flows, EEG, ECG, Exercise ECG, Treadmill work.

**Psychological intervention?**

This refers to any test which purposely alters the mood of the participant or involves administering personality inventories, or any other form of psychological test.

**OR**

**I confirm that my research does not fall into any of the above categories (*please go straight to Section 3*)**



## Research Insurance Checklist



### SECTION 2 – CLINICAL TRIALS INSURANCE

Please complete this section only if you ticked one of the boxes in Section 1.

Does your research study involve:

- Pregnant persons as participants with procedures other than blood samples being taken from them?
- Children aged five or under with procedures other than blood samples being taken from them?
- Activities being undertaken by the lead investigator or any other member of the study team in a country outside of the UK? *If 'Yes', please refer to the 'Travel Insurance' guidance on Page 1 of this form.*

OR

- I confirm that my research does not fall into any of the above categories

### SECTION 3 – OTHER HAZARDS

Does your research study involve:

- Working with Hepatitis, Human T-Cell Lymphotropic Virus Type iii (HTLV iii), or Lymphadenopathy Associated Virus (LAV) or the mutants, derivatives or variations thereof or Acquired Immune Deficiency Syndrome (AIDS) or any syndrome or condition of a similar kind?
- Working with Transmissible Spongiform Encephalopathy (TSE), Creutzfeldt-Jakob Disease (CJD), variant Creutzfeldt-Jakob Disease (vCJD) or new variant Creutzfeldt-Jakob Disease (nvCJD)?
- Working in hazardous areas or high risk countries? *Please refer to the 'High Risk Countries' guidance on Page 1 of this form.*
- Working with hazardous substances outside of a controlled environment?
- Working with persons with a history of violence, substance abuse or a criminal record?

OR

- I confirm that my research does not fall into any of the above categories

**The MANCHESTER METROPOLITAN UNIVERSITY**  
**Faculty of Science and Engineering**  
**RISK ASSESSMENT COVER SHEET**

<b>REFERENCE NUMBER:</b> NPC / 090517 / JDE1.49			
<b>SCHOOL:</b> Computing, Mathematics & Digital Technology			
<b>TITLE OF WORK:</b> CMT Projects involving software development only			
<b>LOCATION OF WORK:</b> John Dalton Building computing facilities, computers at student's own home etc.			
<b>INTENDED ACTIVITIES</b> (attach methods sheets (e.g. standard operating practices) and work schedules to this form): <p style="margin-left: 20px;">General use of computers to develop and test software. Method sheets and work schedules not applicable.</p>			
<b>PERSONS AT RISK</b> (list names of all individuals (including status e.g. staff/student), and/or unit(s) / course(s) undertaking the activity. For students please indicate course and level, for staff give contact email / phone number): <p style="margin-left: 20px;">Undergraduate students.</p>			
<b>HAZARDS</b> (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form): <p style="margin-left: 20px;">Repetitive Strain Injury – work related upper limb disorder            Back injury resulting from improper posture            Eye strain            Fatigue Stress            Possible risk from 240v electrical mains supply</p>			
<i>Are these hazards necessary in order to achieve the objectives of the activity?</i> <p style="margin-left: 20px;"><b>Yes</b></p>			
<b>Hazard Rating</b> (delete as appropriate): <b>Low</b>			
<b>HAZARDOUS SUBSTANCES/MATERIALS USED AND HAZARD CLASSIFICATION</b> (appropriate COSHH data sheets / risk assessments must be attached to this form):			
<b>ALL CONTAINERS OF HAZARDOUS SUBSTANCES SHOULD BEAR CORRECT HAZARD WARNING LABELS.</b>			
<b>NAME OF MATERIAL</b> <i>Please provide also approximate quantity and concentration if applicable.</i>	<b>HAZARD CLASS</b>	<b>HAZARD LABEL</b>	<b>DISPOSAL</b> <i>Hazardous materials must not be removed from laboratories. List disposal arrangements for all materials listed below in the location where the work will be carried out:</i>

**RISK CONTROL METHODS** (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form):

The hazards identified above are controlled by:

Facilities review when laboratories are commissioned  
 Induction session on H&S given to students by Technical Services Manager  
 School H&S information given in Student handbook  
 Posters in laboratories  
 PAT testing of equipment after three years  
 Annual H&S inspections

*The laboratory workstations, whilst not legally required to be DSE compliant, (the continuous usage is too low to present risk) are fully compliant with current legislation. Monitors and keyboards are adjustable, chairs are adjustable and the lighting designed for both computer usage and associated reading activity. In each laboratory, there is an adjustable desk, suitable for wheelchair users, usually located in the next to the door.*

**Hazard Rating with Control Methods** (delete as appropriate): **Low**

**Will any specific training be required (if YES give details)?** N/A

**Are there any specific first aid issues (if YES give details)?** N/A

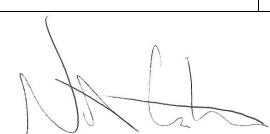
**PROCEDURE FOR EMERGENCY SHUT-DOWN** (if applicable):

In the event of fire, flood or other emergency, evacuation of the laboratory would take place and the technical staff would subsequently make an assessment of the necessity of switch-off. As overall system control is vested in a separate server room, there would be little physical harm to any device in directly cutting the power to the mains for each individual lab.

Re-start of the lab may present problems of a technical nature but would not affect the personal safety or health of any individual.

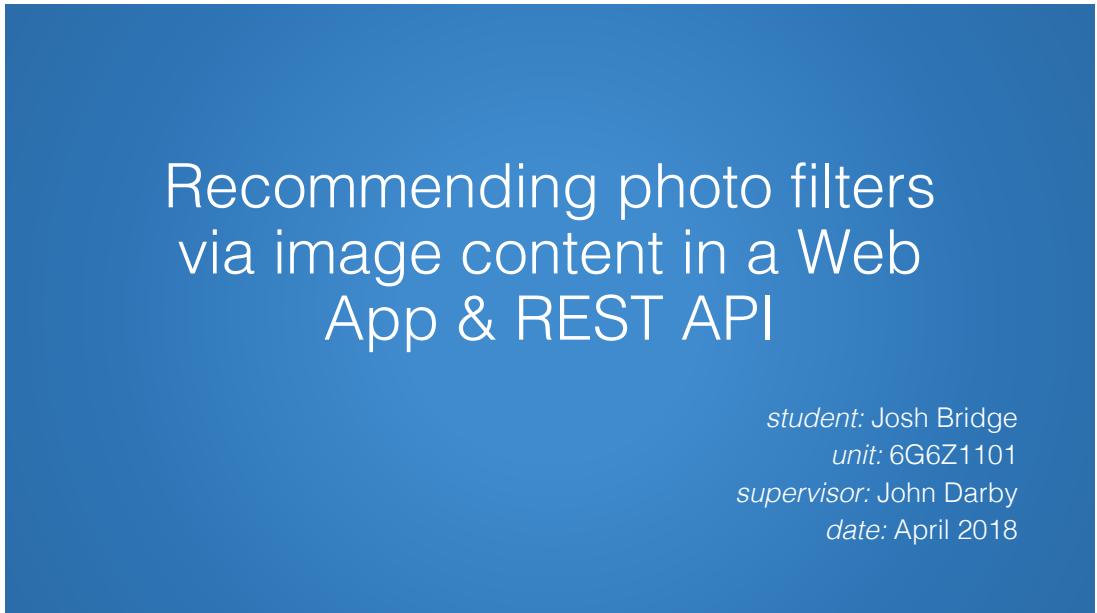
**IF OFF-SITE INDICATE ANY OTHER ISSUES** (e.g. associated with: individual's health and dietary requirements (obtain off-site health forms for all participating individuals and indicate where this information will be located); social activities, transportation, ID requirements; permissions for access and sampling).

**Not applicable – this form applies only to the laboratories listed**

	NAME	STAFF/STUDENT No.	DATE
<b>Originator</b>	Nicholas Costen	01900261	171016
<b>Supervisor</b>	N/A		
<b>Technical Manager</b>			
<b>Divisional / School Health and Safety Coordinator (p.p. HoS)</b>		Digitally signed by Nicholas Costen DN: cn=Nicholas Costen, o=MMU, ou=SCMDT, email=n.costen@mmu.ac.uk, c=GB Date: 2017.05.24 16:33:58 +01'00'	
<b>DATE TO BE REVIEWED BY:</b> September 2018			

# Appendix C

## Presentation slides



Recommending photo filters  
via image content in a Web  
App & REST API

*student:* Josh Bridge

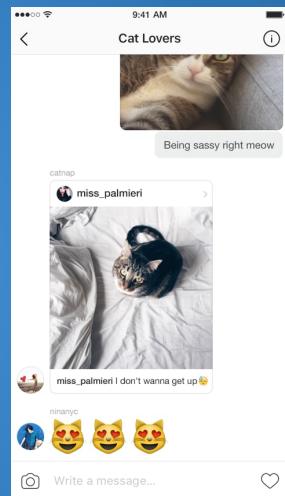
*unit:* 6G6Z1101

*supervisor:* John Darby

*date:* April 2018

## Problem?

- Sharing photos on social media
- Lazy people
- No time
- Too many choices

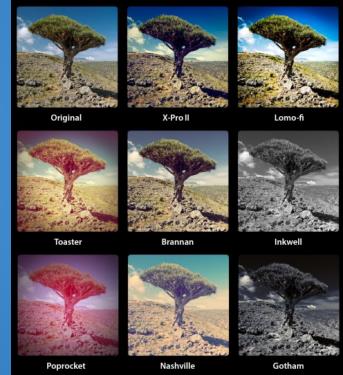


## Aims

- Reduce complexity & information overload
- Secure API for other applications
- Simple, effective front-end

## What are photo filters?

- Colour effects
- Contrast effects
- Brightness effects



Instagram photo filters

## Other apps



VSCO

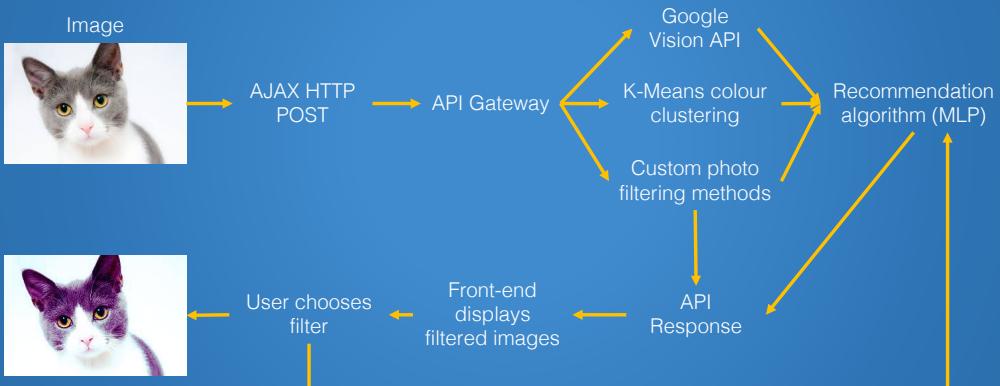


Afterlight



Snapseed

## Design



## Implementation

### API

- Python
  - Flask
  - NumPy
  - Pillow
  - Scikit-Learn
- MongoDB
- K-Means
- Google Vision API
- Multi-layer Perceptron

### Front-end

- React

## Effectiveness

- Low Multi-Layer Perceptron Accuracy
- 2 out of 5 people said recommendation was accurate
- Front-end works well
- API works well
- Filters work well

## Conclusion

- Product reached most objectives
- Recommendation could have been better
- Difficulty learning new technologies slowed progress

# Appendix D

## Product Link

OneDrive Link

<http://inflex.co>

<https://api.inflex.co>

# Appendix E

## Questionnaires

1. Was the recommendation given to you by the web app accurate (i.e. would you have chosen that filter even if it was not shown as recommended)? (**yes/no**)

No

2. How easy to use did you find the user interface? (**score of 1 to 5, 1 being very difficult, 5 being very easy**)

2

Figure E.1: Participant 1 questionnaire

1. Was the recommendation given to you by the web app accurate (i.e. would you have chosen that filter even if it was not shown as recommended)? (**yes/no**)

Yes

2. How easy to use did you find the user interface? (**score of 1 to 5, 1 being very difficult, 5 being very easy**)

4

Figure E.2: Participant 2 questionnaire

1. Was the recommendation given to you by the web app accurate (i.e. would you have chosen that filter even if it was not shown as recommended)? (**yes/no**)

No

2. How easy to use did you find the user interface? (**score of 1 to 5, 1 being very difficult, 5 being very easy**)

4

Figure E.3: Participant 3 questionnaire

1. Was the recommendation given to you by the web app accurate (i.e. would you have chosen that filter even if it was not shown as recommended)? (**yes/no**)

No

2. How easy to use did you find the user interface? (**score of 1 to 5, 1 being very difficult, 5 being very easy**)

3

Figure E.4: Participant 4 questionnaire

1. Was the recommendation given to you by the web app accurate (i.e. would you have chosen that filter even if it was not shown as recommended)? (**yes/no**)

Yes

2. How easy to use did you find the user interface? (**score of 1 to 5, 1 being very difficult, 5 being very easy**)

4

Figure E.5: Participant 5 questionnaire