

Dublin City University

Technical Manual

Project Name: Clothes Similarity Search

Student Name: Dovydas Baranauskas

Student Number: 15336731

Supervisor Name: Owen Corrigan

Dovydas Baranauskas

5-17-2019

Contents

1. Introduction	2
2. General Description	5
3. Design	8
4. Implementation	12
5. Problems and Resolution	16
6. Testing and Validation.....	18
7. Results.....	35

Abstract

Clothes similarity search is a web application which aims to find fine grained similarity in images of clothing. Using deep neural networks, the model aims to find articles of clothing that are similar in design and colour and serve them to the user.

The web application allows a user to upload an image of an article of clothing, and using the neural network algorithm, the model hopes to get the most similar images to the query image from the database.

1. Introduction

Overview

The aim of this web application is to allow users to upload images of articles of clothing and based on the uploaded image, return similar images. This is especially useful when you are looking for a specific item of clothing i.e. a hoodie, but in a different colour or design pattern. This web application can benefit companies to help improve user satisfaction and customer retention as users can quickly browse for specific articles of clothing by simply uploading an image.

The web application uses deep neural networks to train a model in order to predict similar image based off of the query image. The model embeds the image as a vector and plots it on a plane. It then collect the closest images based on distance to the query image. The images will be gathered and displayed in the web application allowing the user to easily view the images.

Glossary

Neural Networks - Artificial neural networks are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

Machine Learning (ML) - Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead.

Artificial Intelligence (AI) - In computer science, artificial intelligence is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and animals.

Web application (web app) - A web application or web app is a client–server computer program which the client runs in a web browser.

Web service gateway (WSGI) – WSGI is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language.

NGINX - Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

Affiliate Marketing - Affiliate marketing is a type of performance-based marketing in which a business rewards affiliates for each visitor or customer brought by the affiliate's own marketing efforts.

Fastai - Fastai is a python library hoping to make AI simpler.

PyTorch - PyTorch is an open-source machine learning library for Python, based on Torch.

Triplet Loss – Triplet loss is an AI algorithm where an anchor input is compared to a positive input and a negative input. The network will be adjusted to prefer the positive input for the specific anchor over the negative input.

Amazon Web Services (AWS) - Amazon Web Services is a subsidiary of Amazon that provides on-demand cloud computing platforms to users.

Google Cloud Platform (GCP) – Google Cloud Platform cuts through complexity and offers solutions for your storage, analytics, big data, machine learning, and application development needs.

Digital Ocean - Digital Ocean provides developers cloud services that help to deploy and scale applications that run simultaneously on multiple computers.

Droplet – Digital Ocean Droplets are flexible Linux-based virtual machines (VMs) that run on top of virtualized hardware.

Graphics Processing Unit (GPU) - A graphics processing unit (GPU) is a computer chip that performs rapid mathematical calculations, primarily for the purpose of rendering images.

Domain Name System (DNS) - The Domain Name System is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network.

Stochastic Gradient Decent - Stochastic gradient descent is an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization.

Data loader - Combines a dataset and a sampler, and provides single- or multi-process iterators over the dataset.

Back propagation - Backpropagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network.

Convolutional Neural Network (CNN) - In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analysing visual imagery.

Recurrent Neural Network (RNN) - A recurrent neural network is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence.

NumPy - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

CUDA - CUDA is a parallel computing platform and programming model that enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

2. General Description

Motivation

I chose this AI project as I have a considerable interest in neural networks and Artificial intelligence. Neural network projects have been interested me and I wanted to understand how they worked. So I decided that I would love to attempt a deep neural network project to gain a better understanding into the topic.

The idea for this project stemmed from a real world use case. I was looking for a specific jumper but in different colour or design as the original jumper was too expensive. When I tried to reverse image search the image on google, the images returned were the same image but from different shops. I decided to search for are any programs that do image similarity search, but I only found humans offering it as a paid service. So this is why I decided to try implement a solution.

Business Context

This system could be adapted by any business or organisation that wishes to retain customers and improve user experience. This can bring in extra customer that enjoy the feature and wish to look for specific articles of clothing rather than having to browse through pages of clothing items searching for a specific piece of clothing. Companies could also train the model on their own databases so that the only clothes that the web application recommends would be from their store. This system could be used by any demographic as it would be very simple, the user would just need to choose a picture of clothes, upload and choose any image they like.

Another business strategy would be to use affiliate marketing. This would involve users that upload images to the web application. Once a picture is uploaded and the user sees the results, the user can click on an article of clothing that they would like to buy. They will get redirected to buy the clothes from an external website, if the user chooses to buy the article of clothing, based on the link we would get some commission for referring the user.

Research

For this project, almost every technology I used was new to me. This was my first web application, and my first time using flask so I needed to research how to create a web application with flask. It was also the first time I worked on server set up and configuration, so I needed to research into how to set up servers to host web applications. Lastly it was also my first time creating a neural network model and training it.

Web application

Since this was my first web application, I researched what would be the best technology and approach to create a web application with python. After reading through online forums and asking my supervisor, I concluded that flask would be the best approach for this task as it is a micro web framework and is quick to set up. I was recommended the “Flask mega tutorial” by Miguel Grinberg, to get me started with making a web app as a tutorial. After following this tutorial I also followed the Flask cookie cutter approach as a guide on which features and coding standards I should follow in order to correctly implement the web application.

Server configuration

This was also my first time setting up a server to serve a web application. I found out that I need to implement a web service gateway. A web server can serve html and CSS files directly from the file system, however it cannot talk directly to the flask application. So you need WSGI to feed requests to the application from web clients, and return responses. uWSGI is a WSGI implantation that flask uses. Nginx is open source software for web serving, reverse proxying, caching, load balancing, media streaming and more.

Artificial Intelligence

I had a limited knowledge of neural network fundamentals, so I decided to do an online course on Coursera by Andrew Ng, to gain a more theoretical deeper understanding. I also researched into potential projects using deep neural networks, when I came across the idea to for clothes similarity search. I researched ways to implement a similarity search when I came across the ‘Fine grained similarity search’ paper. This paper explained how a multi-tiered neural network architecture with triplet sampling is used to achieve highly accurate results. I decided to try implement this architecture to achieve a clothes similarity search. I brought this idea to my supervisor, and he also recommended that I look into the deep fashion paper which created a fashion dataset with over 800,000 images for classification tasks. So I decided to merge the ideas of the two papers into one project.

Once I had the theoretical knowledge and knew how to approach my project, I needed a more practical understanding of neural networks and how to apply them. I decided to follow the fast.ai online course using their Fastai library, I was quickly able to set up and train some neural network classifiers. I set up a data loader with the deep fashion dataset to predict the

category of clothing, as a sub task to get a better understanding of Fastai and neural networks I received an overall accuracy of 70%. Once I set up the classifiers, I learned that I could not do my project using the Fastai library as it did not incorporate many other features other than classification which I needed for the similarity search.

I decided to use PyTorch instead, so I needed to research how to set up and train neural networks using PyTorch. I learned that PyTorch offers a TripletMarginLoss function which allows to easily pass an anchor, positive and negative image to train the loss function. I also decided to use a more basic resnet model instead of a three tier architecture as this would be easier to implement within the time frame.

System Functions

The system is designed as a web application to allow users to search for similar clothes based on a query image. The web application is designed to be as simple and intuitive to use as possible to cater for all possible users.

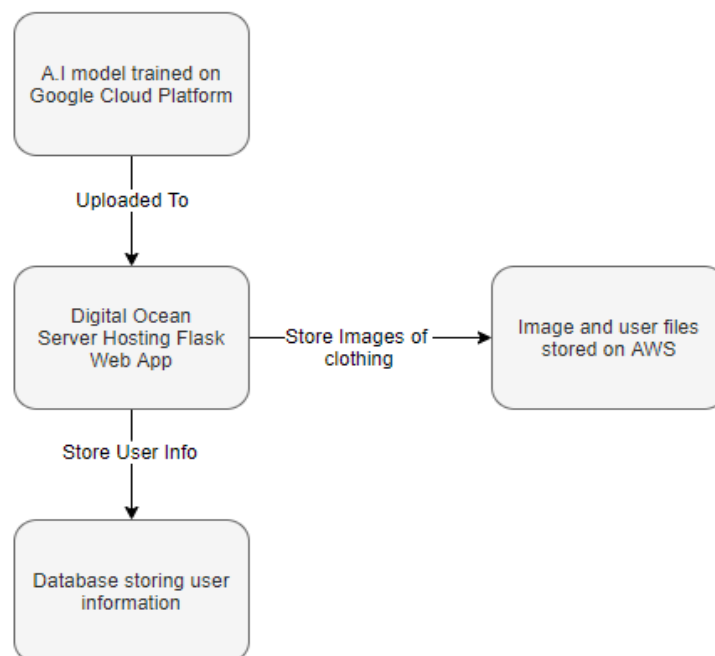
The user will need to create an account and log in, to access the file uploading page. From there with an easy click of a button, the user can choose any image file that they have downloaded on their PC to upload. Once the file is chosen, it is processed by the deep learning algorithm, and similar clothes are returned and displayed in a table. From there a user can choose to up vote or down vote an image if they wish to do so.

The user can then navigate to their profile page if they wish to look at their previously uploaded images. From the profile page they can quickly search for a previously searched image again, or remove the image from their saved. They can also choose to delete their account if they no longer want the application to have access to their data.

3. Design

The web application design has evolved from the initial functional requirements which is natural in most software development lifecycle. Some features such as the android application had to be excluded as due to time constraints, as it would not be feasible to implement on time. The overall architecture changed also, as I added more cloud based features such as deployment on Digital Ocean, and hosting the image data on AWS.

System Architecture



The clothes similarity search system architecture is a cloud based computing platform hosted on Digital Ocean, and Amazon Web Services. The web application contains

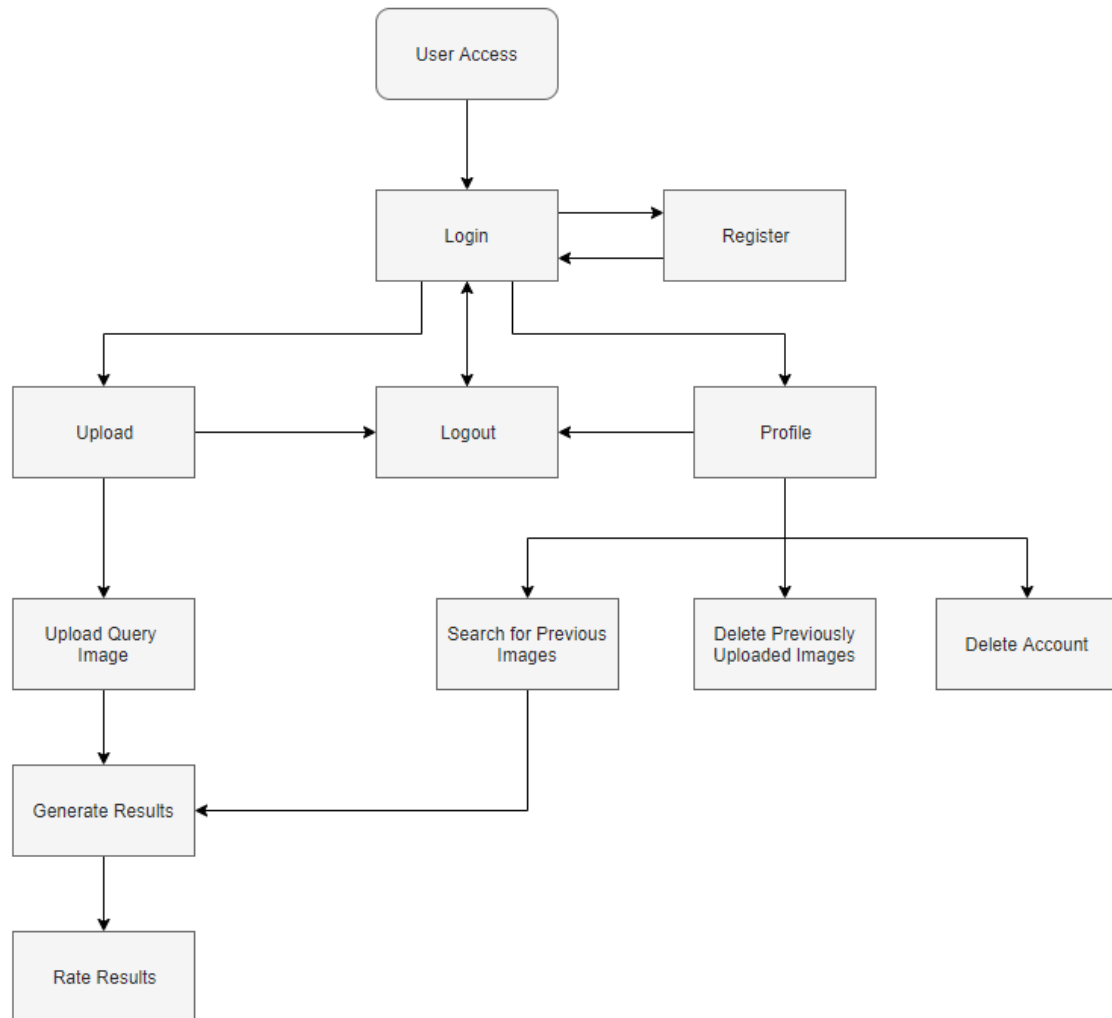
- Linux Operating system
- Nginx/Apache Web Server
- SQLite database
- Digital Ocean for hosting server
- AWS buckets for storing images of clothes
- GCP for training the neural network model

The Digital Ocean Linux server is responsible for hosting the flask web application and computing the similar images to the query image. I chose to host the server on digital ocean as their droplet feature are containerised versions of the application, so if I wanted to

implement some load balancing features, this could be easily done by simply requesting it. I also received fifty euro in student credit from the GitHub developer pack, so I was able to use the service for free. The graphical user interface is created using bootstrap, CSS and HTML to allow users to easily navigate through the website. The model to predict the similar images is trained on the google cloud platform with a GPU. I decided to use the google cloud platform as they offered three hundred euros in free credits if you were a new user to the platform, and I also received another fifty euro credit from the university. At the end I had used up about two hundred euro in credits, so choosing google cloud platform was a correct decision. Once the model was trained it was sent to the live server so that users can upload images and generate results.

Users can register and create accounts, which allows them to access the web application. The user data is stored in a SQLite database on the server. The image data that each user uploads and gets returned is all hosted on Amazon Web services. I chose AWS as they offered one hundred and fifty euro in free credit which I hoped to avail of, I made over fifty thousand requests and only paid thirty cent for them all. This creates a separation of the data and the system function which allows for easy load balancing, as we can easily spin up more droplets using Digital Ocean when traffic increases.

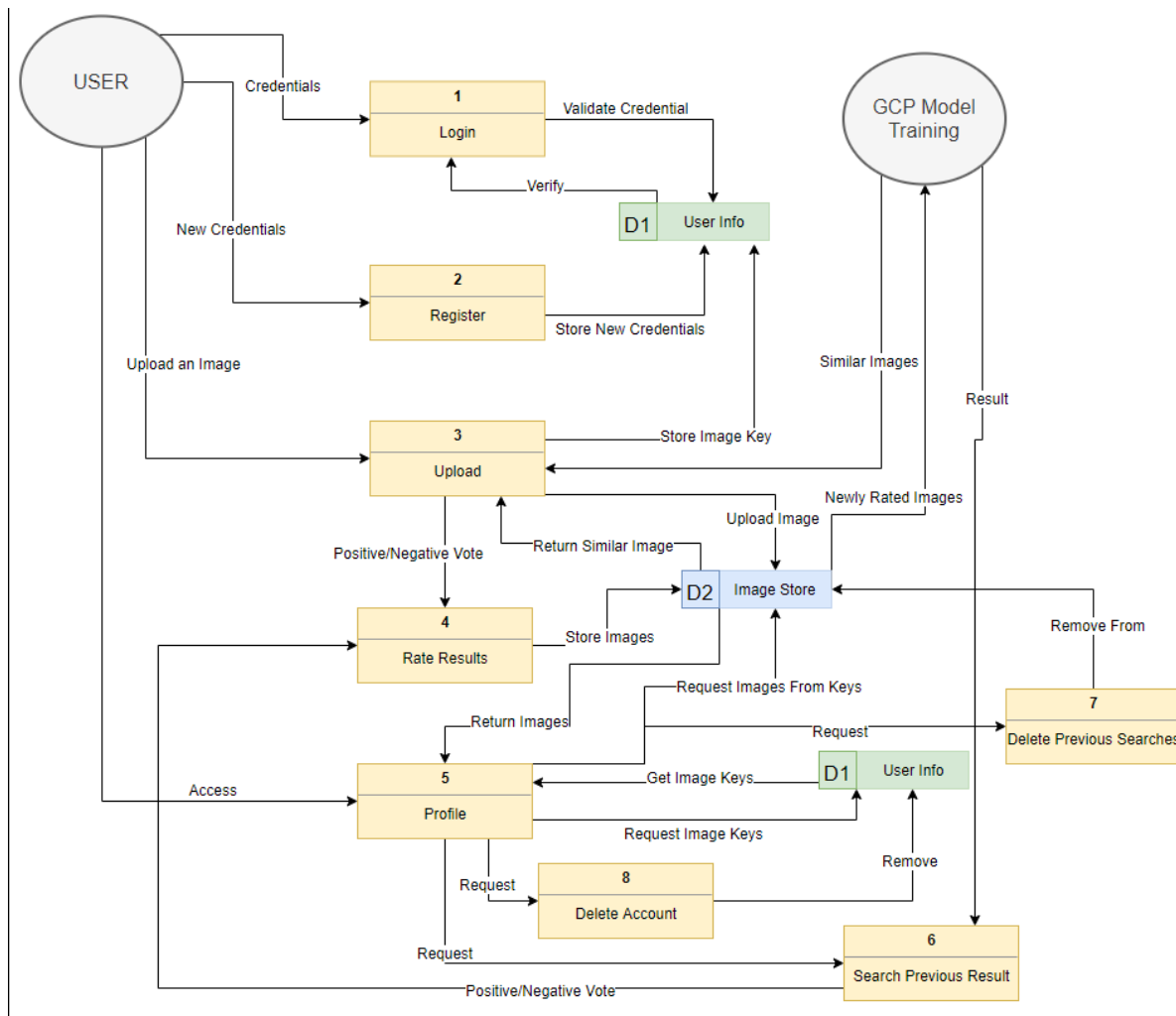
High Level Design



This is a high level design of how a user could interact with the system. The front end is designed to be simple to use, as the front end was developed following Schneiderman's 8 golden rules. When a user first accesses the web application, they will be asked to log in or register. If the user registers, they will be redirected to the login page, from there the user enters their credentials and can access the landing page.

From the landing page, the user can choose to upload an image, access the profile page, or logout. If the user chooses to upload an image, they will be prompted with a file selection screen where they can choose a file to upload. From there the user will be shown their results. If the user chooses to access their profile page, they will be able to see all their previous uploads. The user can either choose to delete the previously uploaded image, or search for the same results again if they wish. Lastly the user can log out of the application if they wish to do so.

Data Flow Diagram







The data flow diagram, shows how the data will flow through the web application. It shows where the data will leave and enter and where it will be stored. The diagram displays how the user can interact with the system and where there information will go when they register, and upload a photo. This also displays the two different data stores, one which stores user profile information, and the other one which stores images. The user can give feedback on the images, which will then be sent to the image store, which can be later used to retrain the neural network.

4. Implementation

Web Application

The web application was created using the Flask web framework. I chose Flask as my web framework as it is easy to implement, and also uses the python programming language which I am most comfortable with. It would also be easier to integrate the trained neural network model which I was also training using python. For the front end of the application, I decided on using HTML, and CSS. I started on the web application using the flask mega tutorial. Using this I implemented the user login system with SQLite.

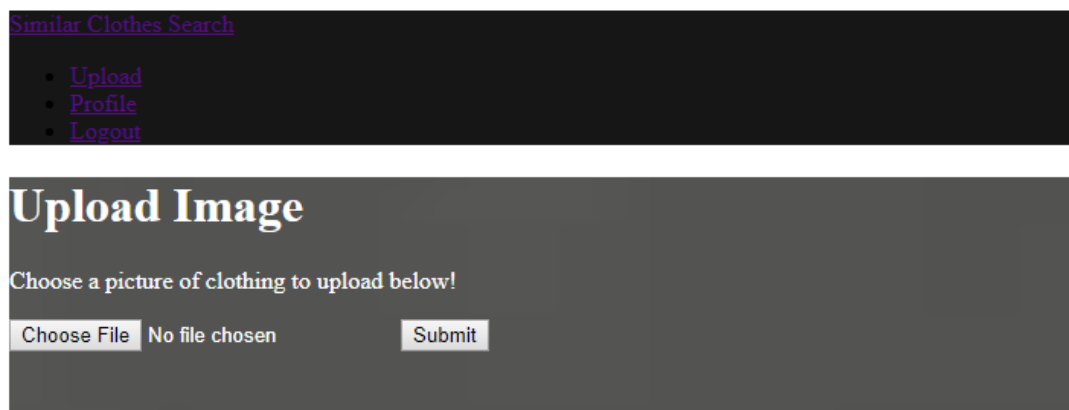
The next feature I added was the ability for users to upload files. This was done using the flask uploads module. This allowed users to upload a file, which was then stored on AWS making sure we also secure the filenames to prevent malicious filenames. I also make sure to secure the AWS links so that only the authorised users can access the images. After the image was uploaded, the web app displayed a set of images as placeholders to where the similar images would be displayed. The s3 console showing a user's uploaded images is shown below.

<input type="checkbox"/> Name ▾	Last modified ▾	Size ▾	Storage class ▾
<input type="checkbox"/>  01_1_front.jpg	Apr 17, 2019 6:35:48 PM GMT+0100	14.5 KB	Standard
<input type="checkbox"/>  01_1_front_2.jpg	Apr 17, 2019 5:57:40 PM GMT+0100	14.5 KB	Standard
<input type="checkbox"/>  03_1_front.jpg	Apr 17, 2019 10:12:12 PM GMT+0100	15.8 KB	Standard
<input type="checkbox"/>  testly.png	Apr 17, 2019 5:27:28 PM GMT+0100	34.5 KB	Standard

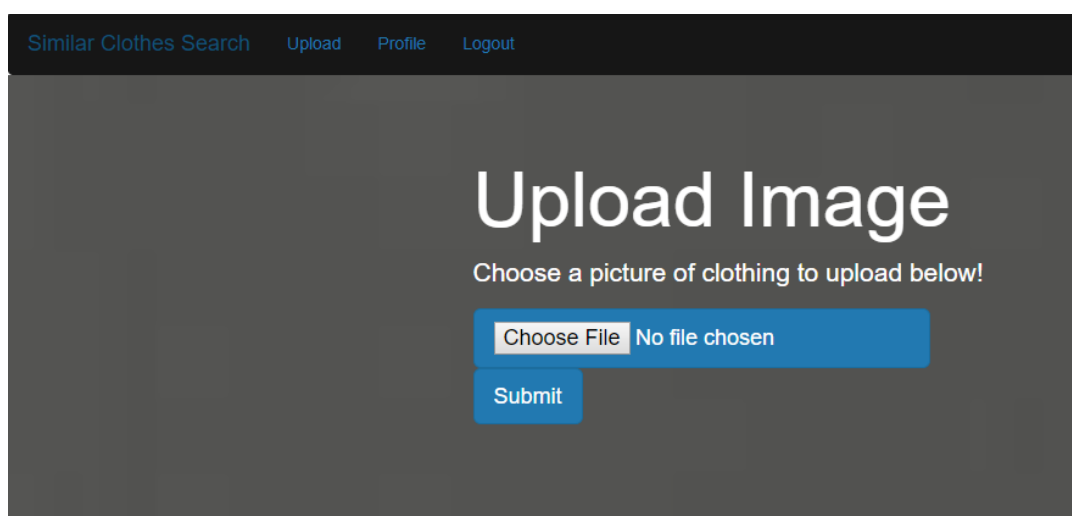
After the uploading functionality was implemented, I focused on creating the user profile page. Each user had a folder on AWS with all their previously uploaded images. These images were all displayed in the profile page, where the user can choose to search for previously searched images, or delete the image, which removes the image from the database and from AWS. I also added some extra logging information which informs me when a user uploads a photo successfully or unsuccessfully, or other actions such as deleting or creating accounts.

After the base functionality was implemented, I focused on improving the look of the application by adding some CSS with bootstrap. I added backgrounds to the login and landing pages, and changed the font sizes and colours. The before and after is shown below.

Before:



After:



I then began integrating the model with the web application. I had to change the mock pipeline to work with the actual working pipeline. I moved all of the image data to AWS for storage, so the model make a prediction of images, and returns the image location in the storage, the images are then fetched by an AWS library and displayed in the table that used to show mock data.

Finally I implemented a feature to allow users to delete their profiles. If a user wishes to delete their profile and remove all information, they can do so in the profile tab by clicking the delete profile option. This will delete the users profile from the database, along with any previously uploaded image names, and will also delete all the images stored on AWS. I also added a feature to allow users to up vote and down vote the results based on the quality. When a user up votes or down votes an image, the query image along with the voted on image get stored in an AWS bucket. If multiple images are voted on, they all get stored in the same folder. This will be useful for constantly retraining the model by having a “Human in the loop”.

Server

This was my first time setting up a server myself, I decided to host the server on Digital Ocean as students get 50 euro credit. I followed a tutorial to set up a droplet with Ubuntu 18.1 and began setting up uWSGI and NGINX configuration. uWSGI is needed to allow the server to communicate with flask and load balancing. The flask built in server should not be used in production as it would be a bottle neck that would slow down the server.

I also added an NGINX implementation, which would improve the server loads even further. Also using NGINX I was able to set up a DNS and got free domain from freenom.com. I was able to set up a simsearch.ml domain for free and host my web application.

Deep learning model

After I had finished researching ways to implement the neural network model, I began my implementation of the model. The first step was to set up all the data to correctly read in all the data annotations to build the data frame. I needed to load in the annotation data into a single data frame, so I needed to merge the information about the clothing such as the category type, the testing/training splits, and the colour of the clothing into a single data frame. An example of the data frame is shown below.

	image_name	clothes_color	eval_status	clothes_type	pose_type	x_1	y_1	x_2	y_2	category_name	t
0	img/WOMEN/Blouses_Shirts/id_00000001/02_1_fron...	Cream	val	1	1	50	49	208	235	Blouses_Shirts	
1	img/WOMEN/Blouses_Shirts/id_00000001/02_2_side...	Cream	test	1	2	119	48	136	234	Blouses_Shirts	
2	img/WOMEN/Blouses_Shirts/id_00000001/02_3_back...	Cream	val	1	3	50	42	213	240	Blouses_Shirts	
3	img/WOMEN/Blouses_Shirts/id_00000001/02_4_full...	Cream	test	1	4	82	30	162	129	Blouses_Shirts	
4	img/WOMEN/Dresses/id_00000002/02_1_front.jpg	Black-blush	train	3	1	65	45	233	252	Dresses	

After all the annotations were set up, I went on to create the data loader to load data into the model before training. I use a deep fashion dataset class which creates batches, and returns three images for the triplet sampling function. The three images are selected when the data loader is initialised by randomly choosing a positive image from the same class, and choosing a random image that is not from the same class.

Once the data loaders were set up, it was time to implement the model. Following a similar GitHub implementation as a reference, I used the resnet50 model to replace the three tier architecture as it would be a much easier and feasible implementation with the limited time for the project. I used the TripletMarginLoss function from the torch.nn library and stochastic gradient decent to train the model. Once the model starts training, the data loader loads the batch of size 16 images and samples. It passes the images into the triplet loss function where it calculates the loss for each batch, and passes it to the optimiser

function. The `optimizer.zero_grad()` function needs to be called before back propagation because PyTorch accumulates the gradients on subsequent backward passes and this is a feature used to train RNN's and not CNN's. I then use the `backward()` function to compute the gradient, and then call the `optimizer.step()` function which updates the parameters. Finally if we are on an epoch that's divisible by 3, we decay the learning rate by dividing it by 1.5 for the next epoch. I trained the model for 25 epochs and made some incremental changes throughout which will be documented in the next section. After the 25 epochs, the loss went down to .35. The code to train the model is shown below.

```
for epoch in range(num_epochs):
    for i, (anchor,pos,neg) in enumerate(dataloader):
        print(i,end='\r')
        #forward pass
        P = forward(pos[0])
        Q = forward(anchor[0])
        R = forward(neg[0])

        triplet_loss = nn.TripletMarginLoss(margin=1.0, p=2)
        loss = triplet_loss(P,Q,R)
        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        Loss_tr.append(loss.item())
        optimizer.step()
        if (i+1) % 500 == 0:
            temp = sum(Loss_tr)/len(Loss_tr)
            print ("Epoch [{}/{}], Step [{}/{}] Loss: {:.4f}".format(epoch+1, num_epochs, i+1, total_step, temp))
            Big_L = Big_L + Loss_tr
            Loss_tr = []

    # Decay Learning rate
    if (epoch+1) % 3 == 0:
        curr_lr /= 1.5
        update_lr(optimizer, curr_lr)
```

The next part of the implementation was to create the embedding's for the training images, in order to make our predictions. This was done by loading the model, and passing the batches of training images to the model. We then concatenate all the results into a list. When we make a prediction, we embed the query image and get the 10 images deemed closest to the query by our model. The accuracy received for the model predicting similar images was 15%. The accuracy was calculated by comparing the original image, and the similar images returned based on if the colour or type of clothes was the same.

5. Problems and Resolution

Throughout the project, there have been many problems. This could have been due to having to use so many new technologies and being unfamiliar with them.

A.I Model Accuracy

Once the model was set up I began observing the results. I noticed that the results were very far off, I was getting results that made no sense to me. For example for female dresses I was getting images of male jackets, I tested the accuracy, which was only 7%. This was very disappointing, so I looked at the best ways to improve the accuracy for the A.I model. The first step was to train the model for more epochs. I started with 6 epochs initially, so I decided to increase the number of epochs to train for 20 more. This slightly increased the accuracy to 9%. I trained the model for 20 more epochs and it increased again to 10%, but decided that I would need to make some more changes to the model in order to increase the accuracy further.

For the next iteration of the model, I was testing out if the problems came from the triplet sampling method. So I decided to remove the in class positive image and only focus on having a class positive image and a negative image. This was to focus on predicting more images from the same class rather than design. I retrained the mode for 10 more epochs, but the accuracy did not change. This turned out to not be the problem so I decided to look for different approaches to fix the problem.

After discussing this problem with my supervisor, we decided to implement bounding boxes to crop the images as we thought the algorithm was learning on the poses of the models and not the clothes they were wearing. I trained this model for 25 epochs. The accuracy slightly increased to 15%, so we were correct in assuming the uncropped images were causing issues. This is the final accuracy achieved as there was not enough time for more iterations. With more iterations, it would be possible to replace the resnet model with the full three tier architecture as described in the deep fashion paper for a higher accuracy. Collecting more images of clothing without models would hopefully stop the model from focusing on recognising model poses, and focus on the actual clothes.

Setting up Machine Learning Environment

There were many problems in setting up the machine learning environment. I encountered problems installing the correct CUDA drivers to work with my GPU. I kept receiving import errors, and after following many tutorials nothing was resolving the issue, so I decided to

install Ubuntu and attempt to install them on a Linux distribution. This worked to fix the issue with the CUDA drivers but the performance was very slow, so I decided to try reinstall Windows, as I believed that the issue was with having an old version of the Windows OS in the first place. Once I reinstalled windows and the new CUDA packages, all the machine learning packages began working. So I was correct in assuming that the original problem arose from an outdated OS.

6. Testing and Validation

Unit and Integration Testing

Unit and integration testing are a vital part of the software development lifecycle to make sure the application system is bug free, and reliable. To make sure the web application continuously performed as expected after any newly added feature. I wrote a series of Pytest unit tests to test each of the web application functions, to make sure they act as expected, and when a new feature is added, the functions still worked. I also added Selenium integration testing, which made sure the entire web application performed as expected after a new feature is added. I used the coverage report library, to calculate my testing coverage for the web application. In total my line by line coverage was 92% so I was fairly happy with that result. The coverage report is displayed below.

```
(flask) C:\Users\dovy\2019-ca400-baranad2\src\simsearch>coverage report
Name                               Stmts  Miss  Cover
-----
app\__init__.py                     24      0   100%
app\confest.py                      43      0   100%
app\dataload.py                     36      9    75%
app\errors.py                        7      3    57%
app\filters.py                       4      0   100%
app\forms.py                         23      0   100%
app\get_query_images.py             54      0   100%
app\models.py                       25      2    92%
app\routes.py                       169     28    83%
app\testing\test_app.py             106      0   100%
app\testing\test_front_end.py        32      0   100%
config.py                           9      0   100%
-----
TOTAL                               532     42    92%
(flask) C:\Users\dovy\2019-ca400-baranad2\src\simsearch>_
```

For the flask web application, I wrote unit tests using a framework called PYTEST. I had to configure a test script, which simulates the entire system using a client. This configuration gets called in a test file that has to be named “test_” followed by a file name, so that PYTEST can automatically know which test files to run. Inside the test file is where I wrote all the test files and helper functions for the tests. I started writing test to check if the basic login, and register pages are up and running. This is done by simply checking the response code returns the “OK” 200 response. An example is displayed on the right.

```
def test_main_page(client):
    """Test if landing page is up"""
    response = client.get('/login', follow_redirects=True)
    assert response.status_code == 200

def test_register_page(client):
    """Test if users can access register page"""
    response = client.get('/register', follow_redirects=True)
    assert response.status_code == 200
```

After this I created tests for the user registration and login. This required a helper function which would populate the registration and login forms that are used by flask to send data across the web application. These forms were filled in with the correct arguments and the login, registration functions were tested. I also made sure to test that the users were receiving informative and correct error messages by logging in or trying to .

Helper function displayed.

```
def register(client, username, email, password, confirm):
    return client.post(
        '/register',
        data=dict(username=username, email=email, password=password, password2=confirm),
        follow_redirects=True
    )

def login(client, username, password, redirect):
    return client.post(
        '/',
        data=dict(username=username, password=password),
        follow_redirects=redirect
    )
```

Unit tests for register and login displayed below.

```
def test_unregistered_user_access(client):
    """Test if users can access page they should not be able to access
    without logging in"""

    response = client.get('/upload/test', follow_redirects=True)
    assert b'Please log in to access this page.' in response.data

def test_valid_user_registration(client):
    """Test a user can register for the app"""

    response = register(client, 'testy', 'testyy@test.com', 'testing', 'testing')
    assert response.status_code == 200
    assert b'Congratulations, you are now a registered user!' in response.data

def test_valid_user_duplicate_user(client):
    """Test a user cannot create an account with duplicate username or email"""

    response = register(client, 'testy', 'testyy@test.com', 'testing', 'testing')
    assert response.status_code == 200
    response = register(client, 'testy', 'testyy@test.com', 'testing', 'testing')
    assert b'Please use a different username.' in response.data
    assert b'Please use a different email address.' in response.data

def test_invalid_user_registration_different_passwords(client):
    """Test if user cannot register with different passwords"""

    response = register(client, 'testy', 'test@test.com', 'test', 'testy')
    assert b'Field must be equal to password.' in response.data

def test_invalid_login(client):
    """Make sure user cannot login with false credentials"""

    response = login(client, "xxx", "xxx", redirect=True)
    assert b'Invalid username or password' in response.data

def test_login(client):
    """Test if user can login and get redirected to upload page"""

    response = login(client, "test", "test", redirect=False)
    assert response.status_code == 302
```

After the login and register functions were tested, I moved on to testing the file uploads. I made sure the user can upload files by creating a test image and attempting to upload it. We again create a helper function to allow us to send the form data from the front end to our back end. This attempts to send the file through the upload and prediction system. If it fails at any point we will get an error that will cause the test to fail. I also test to make sure the user gets the correct error message if they upload the incorrect file. These are displayed below.

```

def test_file_upload(client):
    """Test if user can upload photo."""

    response = login(client, "test", "test", redirect=False)
    assert response.status_code == 302
    files = 'app/testing/test.jpeg'
    response = upload_image(client, files, redirect=True)
    assert response.status_code == 200

def test_incorrect_file_upload(client):
    """Test if users are not allowed to upload incorrect file types"""

    response = login(client, "test", "test", redirect=False)
    assert response.status_code == 302
    file = (io.BytesIO(b'my file contents'), "test.test")
    response = upload_image(client, file, redirect=True)
    assert b"Incompatible File" in response.data

```

The final set of test cases were for testing the profile page available for users. I tested if the file that the user uploaded is in the database, and to see if the user can search for the previously uploaded image. After this I also made sure that the users could also delete the image and then if they wish to, they can delete their accounts. I also implemented a test to see if users can rate the returned images correctly. The test cases are shown below.

```

def test_search_previous(client):
    """Test if searching for previously uploaded image is working"""

    response = login(client, "test", "test", redirect=False)
    assert response.status_code == 302
    file = "test/test.jpeg"
    response = search_image(client, file)
    assert response.status_code == 200

def test_delete_file(client):
    """Test if file is deleted """
    response = login(client, "test", "test", redirect=False)
    assert response.status_code == 302
    bucket = routes.get_bucket()
    key = "test/temp.png"
    response = delete_image(client, key)
    bucket.Object(key).delete()
    assert b"Successfully removed file" in response.data

def test_rating_system(client):
    """Test if rating system is working"""

    response = login(client, "test", "test", redirect=True)
    assert response.status_code == 200
    query = "test.jpeg, img/MEN/Denim/id_00000080/01_1_front.jpg, positive"
    response = rating_check(client, query)
    assert b'Thank you for your feedback!' in response.data

def test_delete_account(client):
    """Test if user can delete accounts"""
    logout(client)
    response = login(client, "testvy", "testvy", redirect=True)
    assert response.status_code == 200
    response = delete_account(client)
    assert response.status_code == 200
    response = login(client, 'testvy', 'testvy', redirect=True)
    assert b'Invalid username or password' in response.data

```

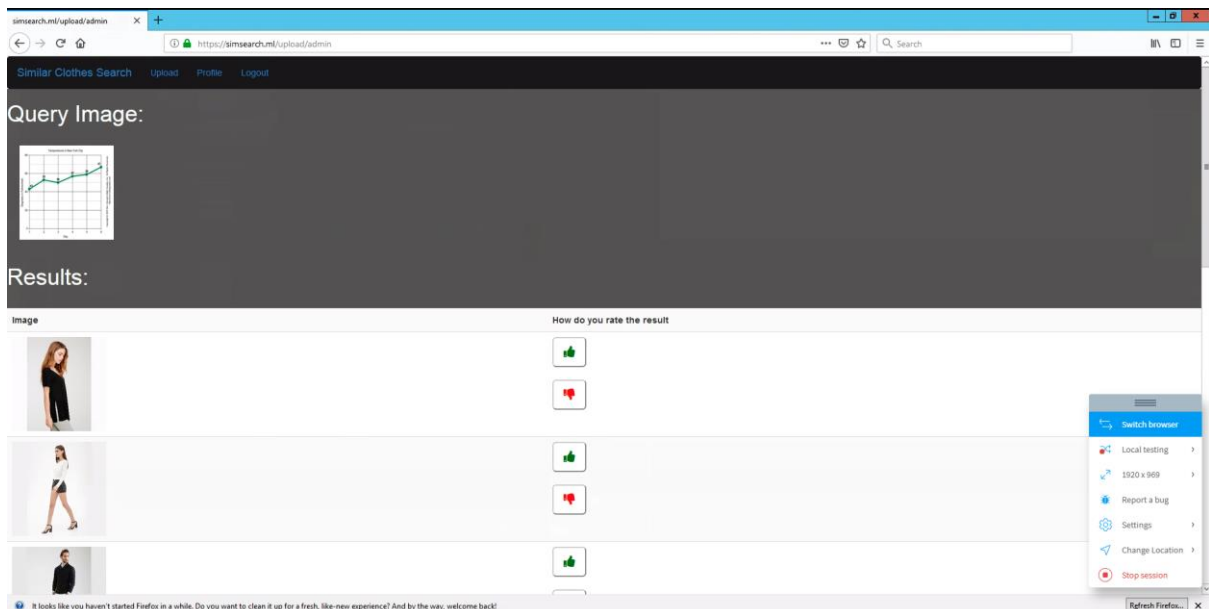
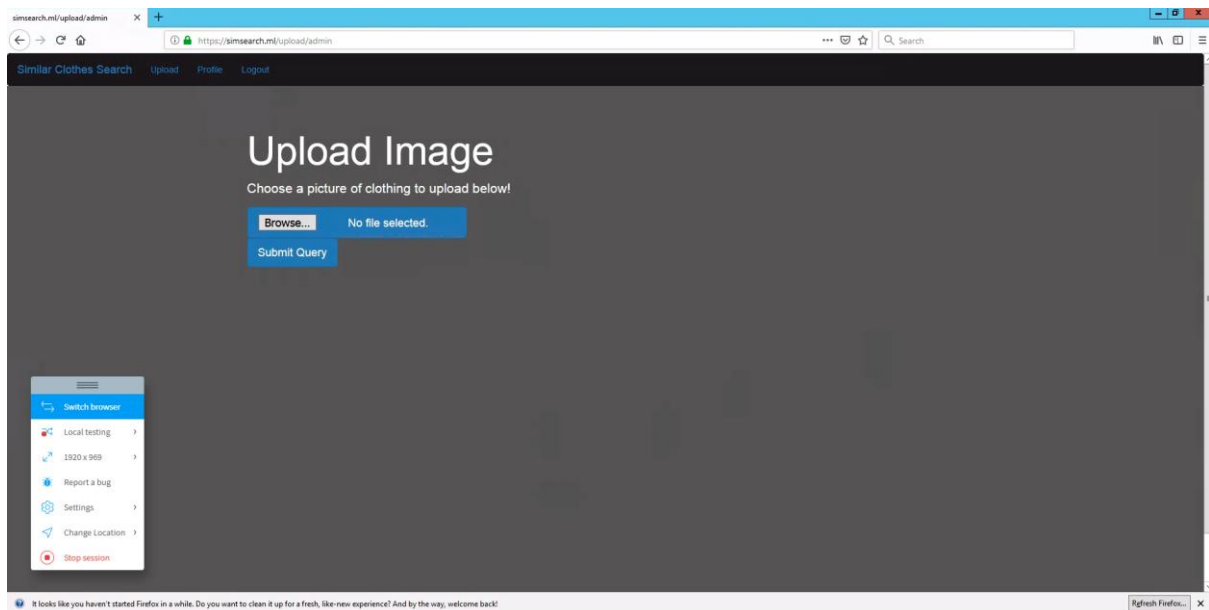
Finally I wrote a selenium test that would run through all of the actions in my web application to make sure, all buttons and all features are fully working together as expected. Selenium uses an automatic google chrome driver to simulate how a user would interact with the web application automatically. All of these tests together gave me a coverage score of 92%.

Browser Testing

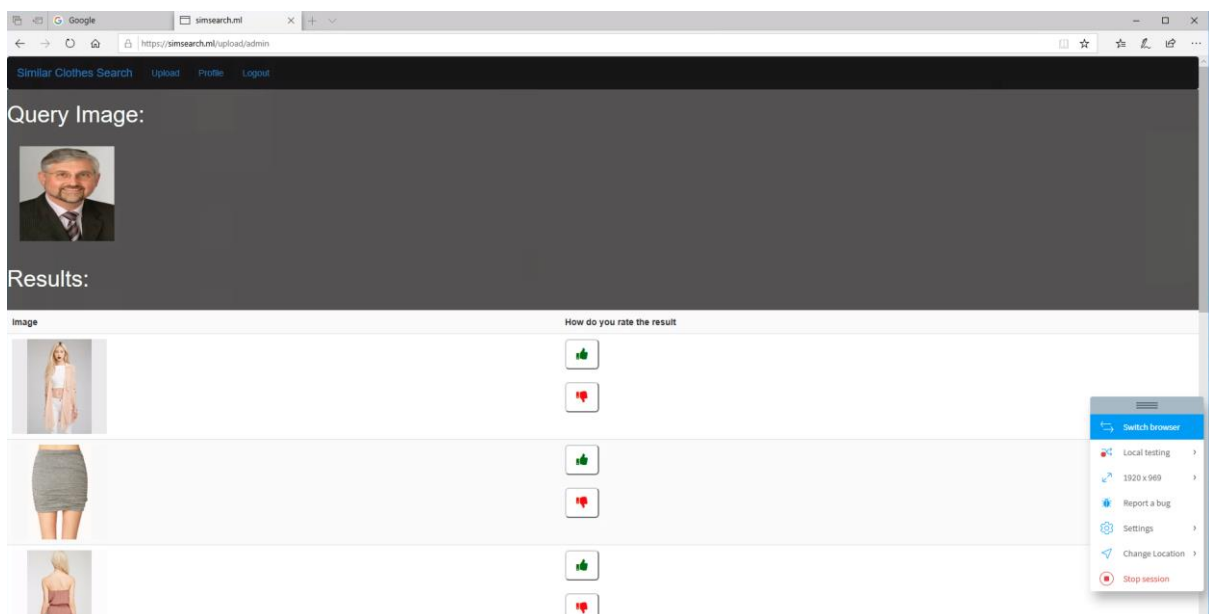
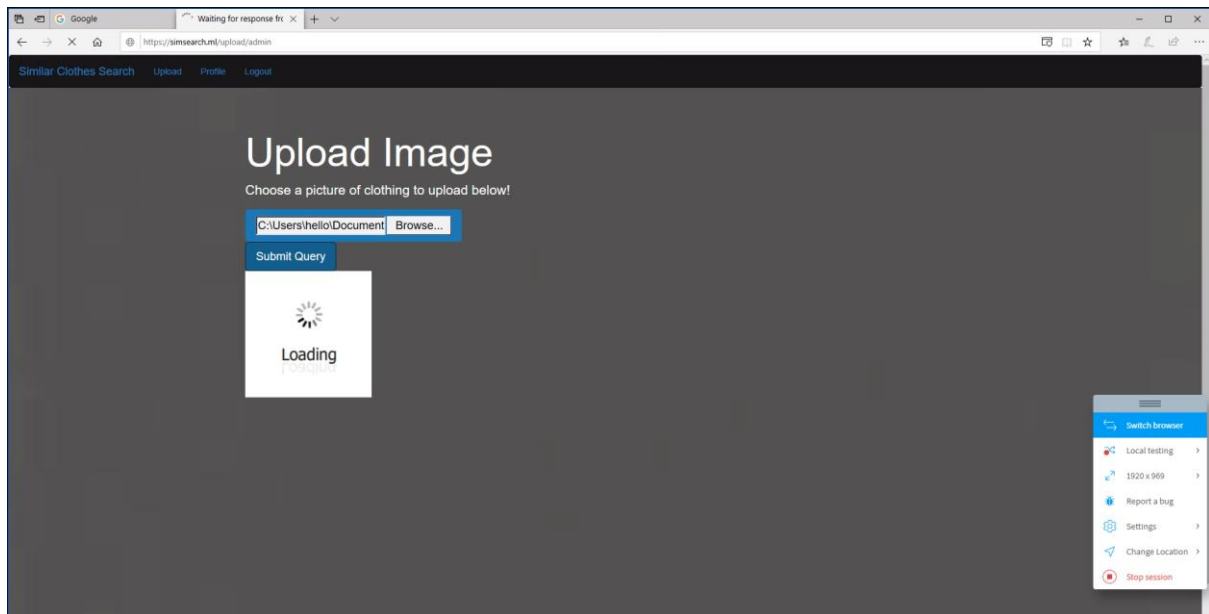
Browser testing refers to testing to make sure the web application works with all different types of browsers. I used a program called browser stack which is a cloud based browser

testing tool which emulates a wide range of browsers. I manually ran through the basic functionality of the web application like uploading an image, and being able to see all your results, and looking at your profile page for your previous searches. The web application behaved as expected throughout, some examples of how the web application looked in different browsers are shown below.

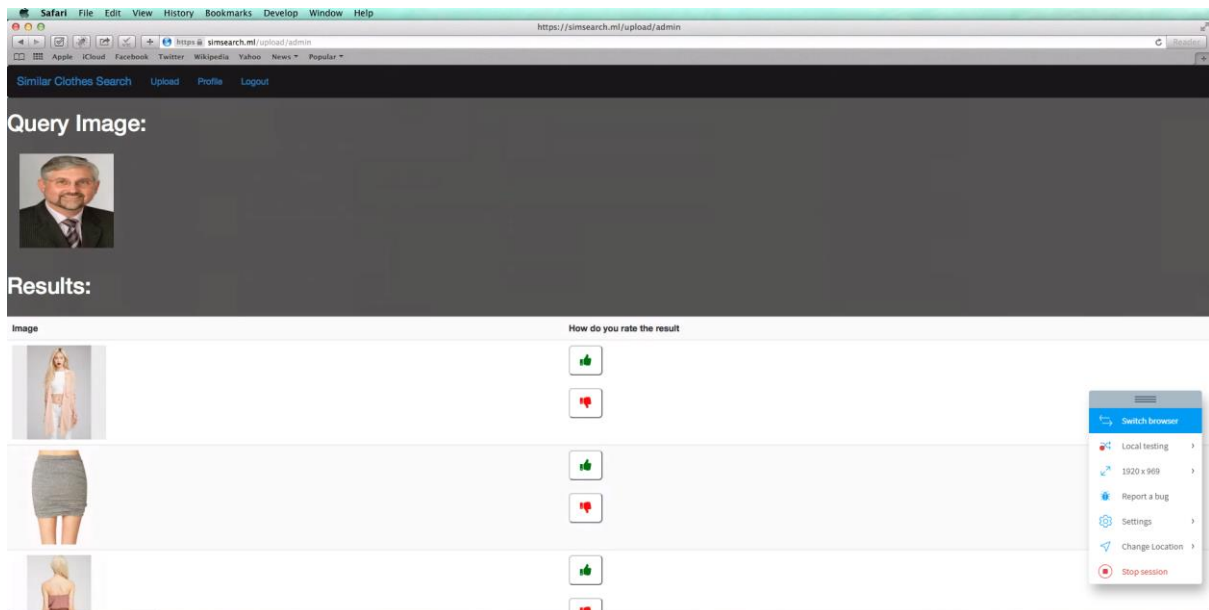
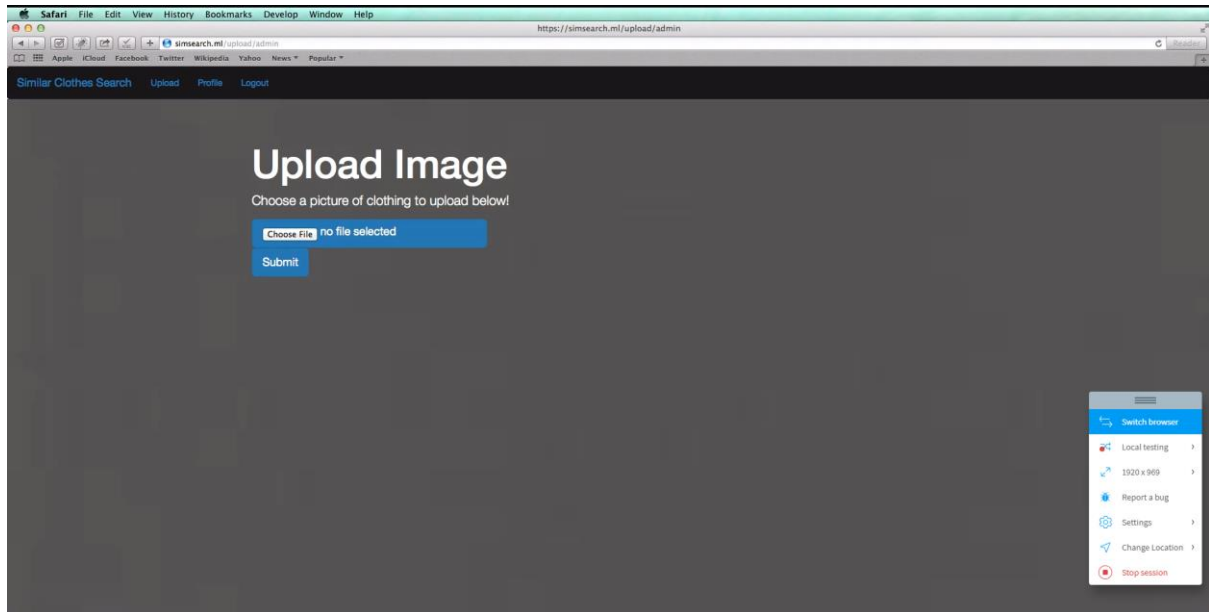
Firefox:



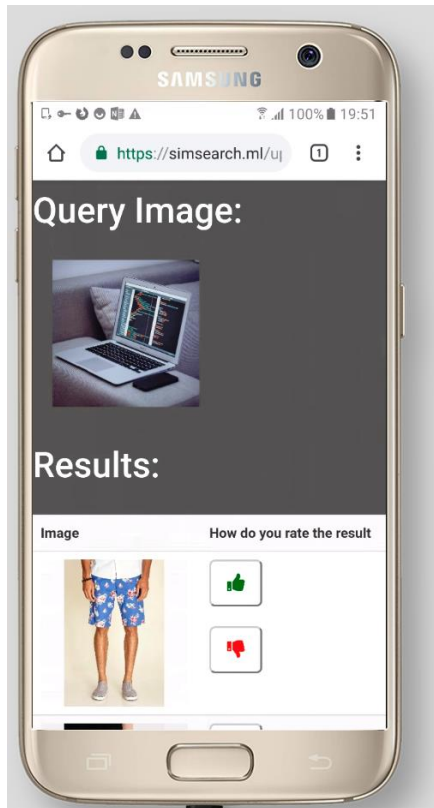
Edge:



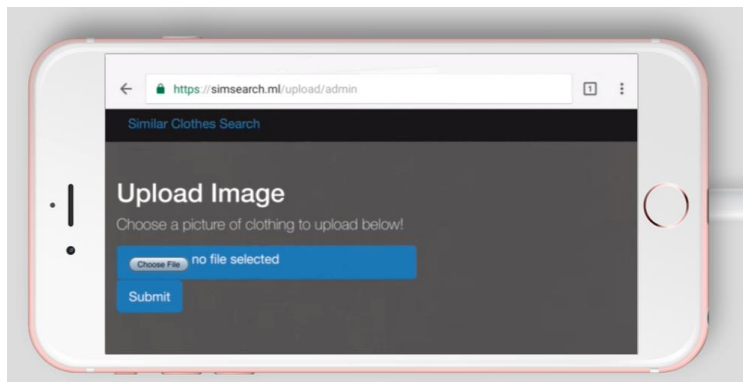
Safari:

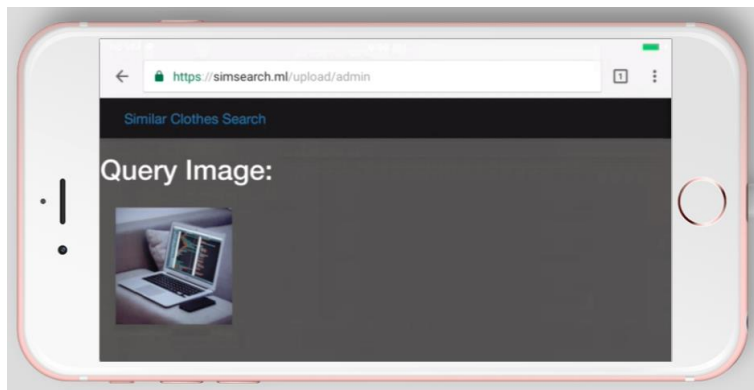


Android Phone:



iPhone:





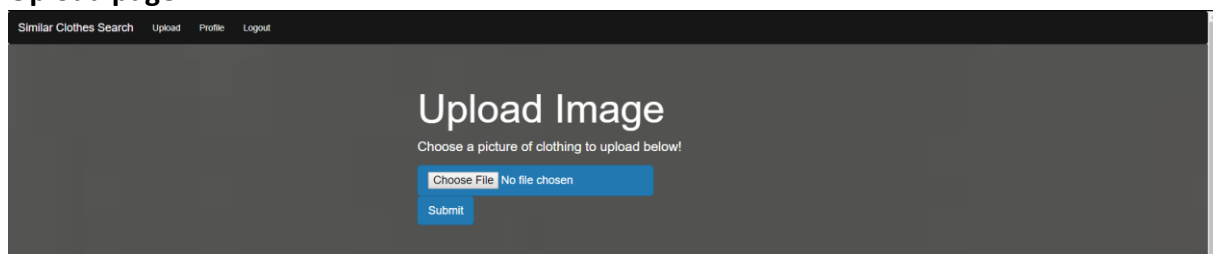
The application performed as expected on all the desktop browsers as it was expected to. There were no major differences between browsers, and the layout stayed consistent which I was happy about as the web application was designed to work mainly on desktop. There were a few errors with the mobile application with the menus displaying incorrectly. With more time there could be a mobile adaptation of the application, but there was just not enough time to add this extra feature.

Usability Testing

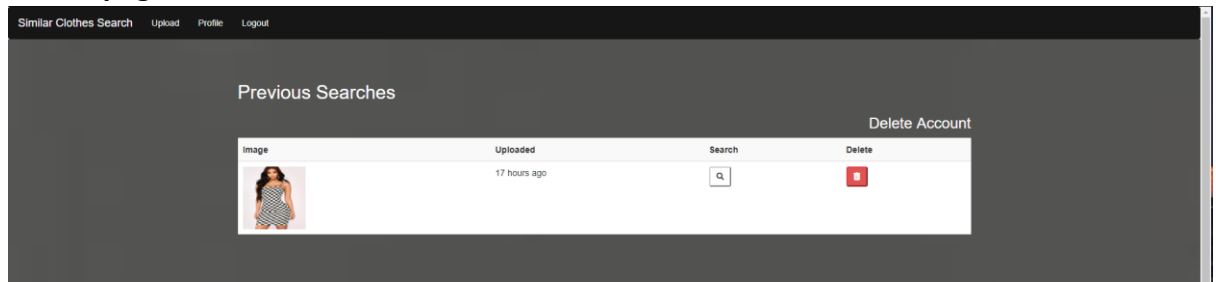
For my web application, I looked into what design approach would be most suited for the front end. I decided to choose Schneiderman's 8 golden rules of interface design as they are generally regarded as a good metric and I studied them in the Human Computer Interactions module.

1. The first rule refers to striving for consistency throughout the application. By utilizing familiar icons, colours, menu hierarchy. This allows the user to easily flow through the web application without needing to think extensively about each action. An example of this is the consistent menu bar throughout the web application

Upload page:

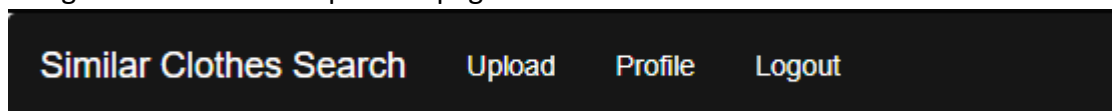


Profile page:



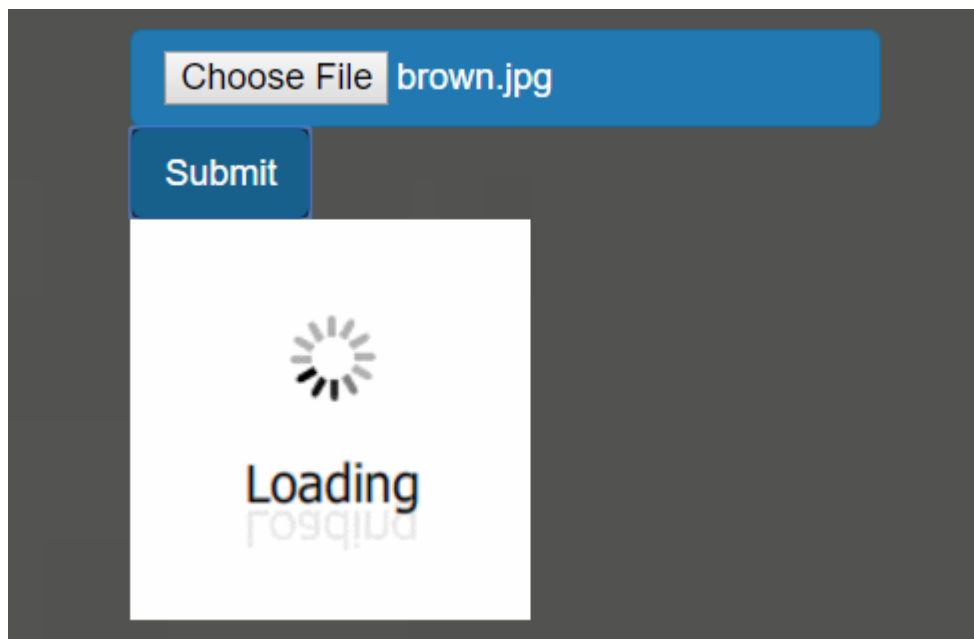
We see even though we are on different pages, the general layouts are very similar on both pages the main content resides in the centre of the page, the menu also stays consistently in the same place throughout the application. Also in the profile page we have a search magnifying icon and a red bin delete icon, which are known icons for searching and deleting actions. This will instantly inform users what actions they are about to do without prior knowledge of the application.

2. The second rule is to enable frequent user shortcuts. When users use applications frequently, they wish to be able to complete their required task as quickly as possible. In the web application, you can quickly access any of the pages by simply using the menu at the top of the page.



This allows users to quickly find the upload or the profile page of the web application. Also when the user first accesses the page they will be instantly shown the file upload screen, which will allow the user to instantly upload an image.

3. The third rule is to offer the users informative feedback. Once the user uploads a file, a loading gif will appear, as sometimes it can take a few seconds to get a result, and to avoid user frustration, they will be shown the loading gif. Once a user rates an image, they will also be informed that the action has been completed correctly. Examples are shown below:







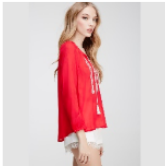




Thank you for your feedback!

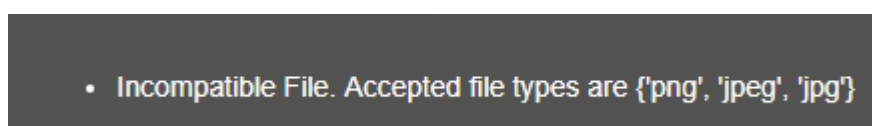
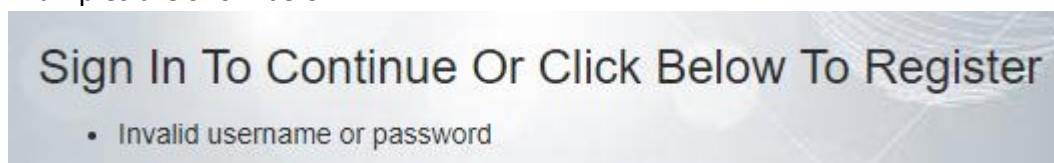
4. The next rule refers to design dialogue to yield closure.
Once you upload an image, you will be brought to the display page where you will be shown the results and your query image. Both of these are clearly labelled and displayed.



Results:

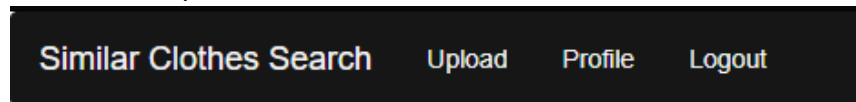
Image	How do you rate the result
	<div> </div>
	<div> </div>
	<div> </div>

5. The fifth rule states that the web application should offer simple error handling. I kept in mind throughout the entire design of the web application, to offer easy error handling to assist the user if something does not go as planned. When a user does not enter correct, input they will always be notified. Incorrect actions such as typing in your username or password incorrectly, or uploading a file with the correct extension, the user will be notified both times why their actions did not work. Examples are show below.



6. The sixth rules refers to easy reversal of action. if there are any internal server errors, the web application will be able to fail well enough to allow the user to click the “go back” which will bring them back to the index page or go to a different tab on the menu, as that will remain on top of the

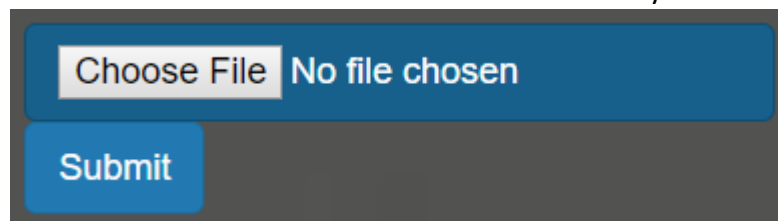
screen. Example of this are shown below



File Not Found

[Back](#)

7. The seventh rule refers to supporting the internal locus of control. Users are in control of which file to upload, and if they wish to longer have an account they can easily delete it by clicking on the delete account link located in the users profile page.
8. The eighth and final rule refers to reducing the short term memory load. Recognising something is always easier than recall, so I made sure to use as many recognisable icons as possible. Also if an action does not have a button, I made sure to label it as accurately as possible. Examples of this are shown below where, the choose file and submit buttons are labelled clearly.



User Testing

Since this web application has a front end which allows users to upload files and access previously uploaded files I wanted to conduct some users test to find out how a user would interact with the web application. I conducted tests where I timed the users interacting with the web application and asked them to fill out a survey afterwards. I used users from different backgrounds and computer experiences to see how well they would interact with the web application. I asked each of the users to do the full run through of the entire web application. The average time for all operations is displayed below.

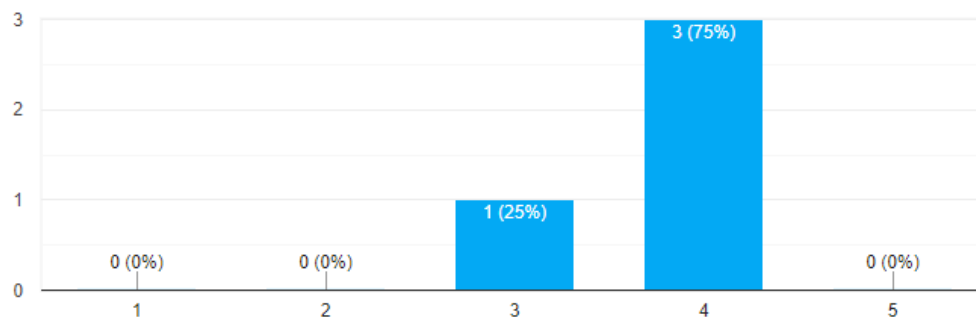
Action	Average Time Taken(seconds)
Registering and logging in.	64.7
Logging in and uploading photo	31.79
Logging in, accessing profile page, search previous images and then deleting them.	22.75

I was fairly happy with the average times as it shows how quickly it is to log in, upload a photo and get a result. And by the last test case you can see how the user can do more of

the actions in less time than it took for the previous test case. This told me that once the users used the application once, they got familiar with it and had no issues with it. I also gathered feedback from the users after where I asked them to rate the look of the overall website.

Rate the overall look at the website?

4 responses

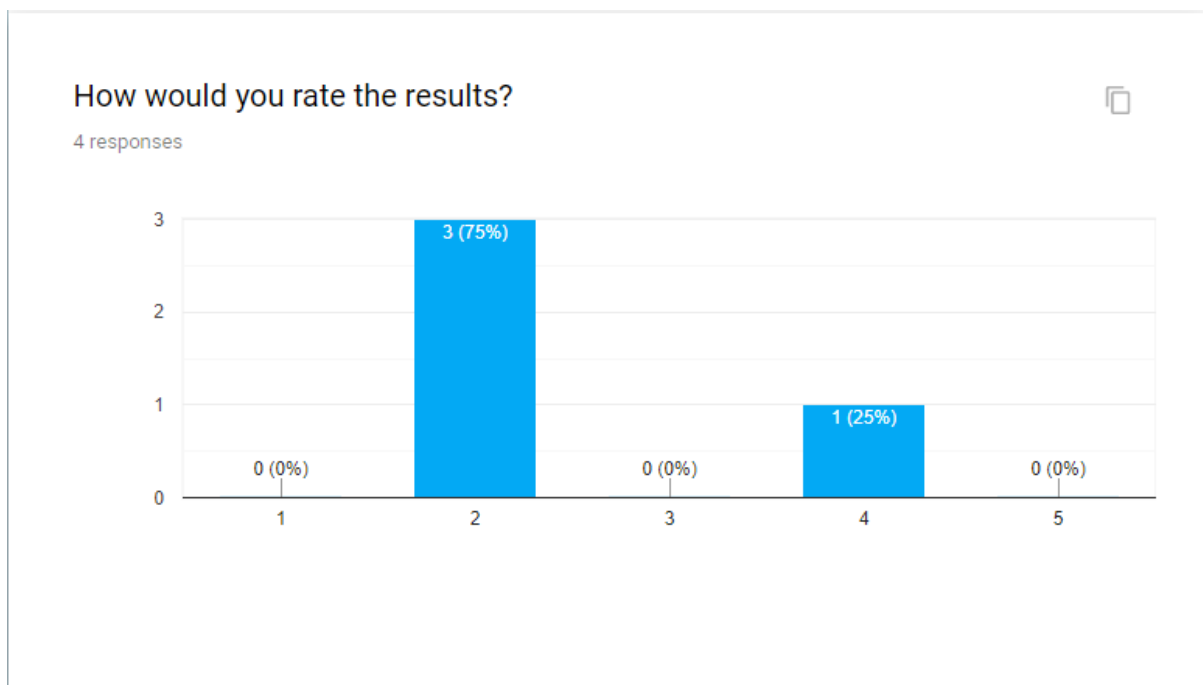


Most of the users were generally happy with the overall look of the website, 75% rating it a 4 out of 5 and 1 user rating it a 3 out of 5. If I did have more time I would go back and add a bit more flair in the overall design as right now it is very simple, but I was aiming for a simple front end design so I am happy with that feedback.

After this I asked the user how easy they found using the website to upload an image, the scale ranging from 1 being very difficult to 5 being very easy. 100% of the user rated that the uploading feature 5 out of 5, being very easy to use. This was great news as my main design focus was the ease of use and simplicity of the web application. The results are shown below.



Then I finally asked how they would rate the results of the web application. I knew that the results were substandard as the accuracy was quite low. I asked the user how they would rate the results 1 being poor and 5 being perfect. 75% of the users rated it a 2 out of 5 and 25% of the users rating a 4 out of 5. These results were expected as the accuracy does need to improve to keep users satisfied. The results are shown below.



Finally I asked if the user had any more comments. One user asked the login and upload form to the middle of the page, which I implemented. They also asked for an option to delete user profiles, I also implemented this feature. The response is shown below.

Move login form to the middle of the page.
Color scheme looks good.
An option to delete user profile.

They also did say that they enjoyed the colour scheme. Another user informed me that they in fact did not like the colour scheme. But they did mention how easy it was to use and that the results were inaccurate.

Don't like grey colour
Easy to use.
Inaccurate results

Performance Testing

To try and test my web applications performance, I used a tool called web page test which is a free online testing tool to test your web application. It allows me to quickly monitor my web applications performance. The result from the most recent iteration of the web application is displayed below.

Performance Results (Median Run)

	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Document Complete			Fully Loaded			
							Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	1.040s	0.315s	0.700s	0.985s	1.100s	> 0.662s	1.040s	6	168 KB	1.200s	7	168 KB	\$----

Web Page Performance Test for

simsearch.ml

From: Zurich, Switzerland - Chrome - Cable

5/17/2019, 9:13:05 PM

Need help improving?

A	A	A	B	A	✓
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

I got great ratings throughout the web application. I got a lower grade in the Compress images category as it informs me that the use of images is most impacting the performance.

7. Results

Future Work

There is a wide use for an application like this in the current market. With more time, and more data I believe I would be able to train the model to a higher accuracy and better performance. There was enough data to build a classifier, but to train a triple loss algorithm I needed more than just the 50,000 images. The web application with consistent use would allow our model to improve even more with the ratings feature.

If there was a bigger data collection effort for in shop images, along with a team of developers more used to the neural networks algorithms, I believe this project could still achieve a higher accuracy. There is also room for improvement in the model if the simple resnet architecture was replaced by the original multi-tier architecture from the deep fashion paper.

Conclusion

Overall, I am quite happy with the final state of the finished application. I set the bar very high for myself at the start of the year, and although I could not complete everything that I wished, the only thing that was stopping me was more time.

With the help of my supervisor I was able to consistently adapt to new requirements and reach my goals throughout this project. Although the A.I accuracy was not perfect, the main aim of the project was to work on my first A.I project and understand the power of neural networks. I now have a greater understanding of the concepts of neural networks and the different techniques used for different tasks. I am much more confident going into the industry knowing about advanced machine learning techniques such as neural networks.