INTELLIGENT IMAGE SEARCH ENGINE WITH AI-BASED SIMILARITY DETECTION FOR WEB APPLICATION

CHONG WAI SOON

UNIVERSITI TUNKU ABDUL RAHMAN

INTELLIGENT IMAGE SEARCH ENGINE WITH AI-BASED SIMILARITY DETECTION FOR WEB APPLICATION

CHONG WAI SOON

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

October 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : Chong

Name : Chong Wai Soon

ID No. : 2106577

Date : 30/9/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"INTELLIGENT IMAGE SEARCH ENGINE WITH AI-BASED SIMILARITY DETECTION FOR WEB APPLICATION"** was prepared by **CHONG WAI SOON** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Mohammad Babrdel Bonab

Date : 30/09/2024

Signature :

Co-Supervisor :

Date :

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, CHONG WAI SOON. All right reserved.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone for providing me with the resources and support necessary to complete this project. I would like to thank my supervisor, Dr Mohammad Babrdel Bonab, for his guidance and support throughout the research process. His expertise and feedback have been invaluable in shaping this project.

In addition, I would also like thank my friend, Chan Jia Jun, for his help in testing the UI flexibility across his laptop. His efforts have been essential to ensure that the web application is user-friendly and performs well on various devices.

Lastly, I am deeply grateful to my family and friends for their unwavering support and encouragement throughout this final year project journey. Their love and support have been a constant source of motivation.

ABSTRACT

With the growing reliance on visual data and the vast amount of information available on the internet, efficiently searching and retrieving relevant images has become increasingly important. Traditional image search engines which majorly depend on keyword-based approaches often give unsatisfactory results since they cannot accurately understand what users mean by their queries in just text. This project aims at developing an intelligent image search engine that uses AI-based similarity detection to have more precise and relevant image retrieval. The project involves developing a web application where users can upload images and search for images in each database that are visually like them. The system uses advanced AI techniques such as Convolutional Neural Networks (CNNs) and Siamese Networks to extract features from input images and compares them with pictures already in the database to pick out and retrieve the most similar ones. This content-based approach does away with the need for describing images using keywords by users thus improving accuracy and relevancy of search results. This project is a web application that uses different technologies such as HTML, CSS, JavaScript, Python, PyTorch and MongoDB. The front-end of the application allows for user interaction while the backend handles image processing, similarity detection and database management. To achieve high accuracy in similarity detection, this AI model is continuously refined and improved through iterative development methodology. The project has great impact on various fields like e-commerce sites, digital libraries or social networks where effective and efficient picture retrieval is highly needed. By creating an excellent image search engine that is user-friendly, it promotes AI powered technology progress in addition to laying a foundation for future research and developments within the field of image search as well as recognition.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
SDLC	Software Development Life Cycle
UI	User Interface
SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
CNNs	Convolutional Neural Networks

CHAPTER 1

INTRODUCTION

1.1 General Introduction

In the modern age, technology had become familiar to everyone which one of the important footprints of technology is Internet. Lots of information can be gained from Internet using the search engine such as Google and Bing by only given the keywords for the searching items such as blogs, images, videos, and news. However, it is difficult to decipher users' search intentions from only the query keywords which may lead to unsatisfactory search results due to there are many information and data nowadays on Internet. Image is one of the things that people like to search on Internet by using keyword and Google Images search engine is one of the famous tools that achieve the function for the people to search the images. Google Images search engine will return large number of images for a given keyword-based query, but it is still unsatisfactory due to poor accuracy rate and include large amounts of junk images. Figure 1.1 shows the top results of query "Tick" in google image search and it shows two different topics which are the symbol and the arachnids, this shows that keyword-based query for image search is not that high accurate.

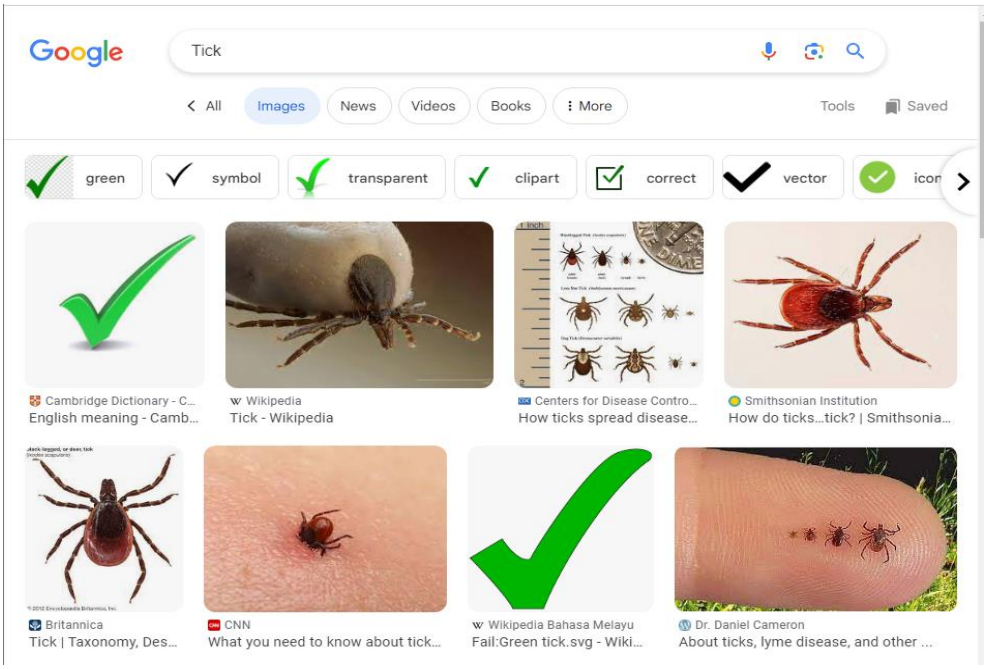

Figure 1.1 illustrates the top results retrieved by searching for "Tick" on Google Image Search

In order to increase the accuracy of getting the correct returned image results, Intelligent Image Search Engine with AI-Based Similarity Detection would be a suitable tool to solve the problem of keyword-based query for searching an image. Image search engine is a specialized search engine for retrieving images which show different to traditional search engines which are type-based search engine. Although type-based search engine can also use to type keywords and find the images, but the text information is limited and can be inaccurate and there is no way to promote the variation of the results without analyzing content of the images. Therefore, image search engine is implemented to allow users to input image query and find the image due to compare the features and patterns of image, retrieved the highest similarity image.

Nowadays, there are many image search engines had been implemented such as in Google it called Google lens and in Pinterest it called Pinterest lens. The similar characteristic of both Google lens and Pinterest lens is using AI for similarity detection of the image. AI has become so popular and trend in different areas and now it also uses in image search engine due to its accuracy and efficiency as it can analyze images quickly and returned the similar image results after searched from the database. By having AI to do the similarity detection in image search engine, this can ensure that the returned image results would be in high accuracy same with the input image query and also efficiency while response in similarity detection. This is because AI similarity detection will determine the similarity between the images by analyze the features of image so it can return the most accuracy result which is the highest similarity image as the result.

1.2 Problem Background

Image search engine had became familiar and popular nowadays due to it is the effective way for people to search for image that they want and there are many images search engines had been developed such as Yahoo Image Search engine, Google Image Search engine and Google lens. However, there are some limitations and weaknesses for each image search engine.

1.2.1 Yahoo Image Search: Keyword-Based Approach

Yahoo Image Search Engine is using keyword-based approach which the image retrieval will based on what text or keyword type by the user then based on the keyword, the related images will be show as results to user. This is the traditional way that many users had used to find the image that they want. However, this keyword-based approach using by Yahoo Image Search Engine had a problem that it may show the wrong results that not the user expected. This is because keyword-based image search engine will depend on the information of keyword and if any image related to the keyword, it will show as results and this will cause the irrelevant search results. (Vijayan Vijayarajan & Mohd Khalid & Chandra Mouli P.V.S.S.R, 2012)

1.2.2 Google Lens: Image Search Engine with Website Results

Google Lens is using content-based approach which needs user to input image query to find the similar image based on the image given. It is an image recognition technology to analyze the features and patterns of the image and retrieve the most similar images from the website. (Viktor Shapovalov & Zhanna Bilyk & Yevhenii Boris Shapovalov, 2020) Although Google Lens is using this image recognition technology and it is high accuracy on its return results, but the results are the images in other websites, and it is not directly return the image only. It is hard for users to find themselves the specific website that they want for the image as the results showed will be all the similar images from other websites in Internet.

1.2.3 Google Image Search: Textual Input

Google Image Search Engine is using textual input which is to input text to access image retrieval. It requires user to input text query and Google Image Search Engine will based on the text to return the image results which fit its description. However, it is hard for users to fully describe the image that they want to search by using text and return with irrelevant results (Bo Luo & Xiaogang Wang & Xiaoou Tang, 2003) Textual input is depend on all the text given by user that to search the image based on the description of text but sometimes user will confuse and wrongly use the word or even do not know which words need to type in order to find the image that they want and this may cause waste of time.

1.3 Problem Statement

An intelligent image search engine with AI-based similarity detection for web application is to give an effective way to user to find the image which has the high similarity with the input image query and return the result images which retrieve from a specific database. Nowadays there are many image search engines existed such as Yahoo Image Search Engine, Google Image Search Engine and Google lens. Although image search engine is that famous and familiar now, but there are still insufficient on each image search engine which the results search is from the Internet but not a specific place. The image search engine existed nowadays do not allow user to get only the results from a specific place but will list all the possible results from other websites. For example, user want to search the image in one specific shopping website, but the existed image search engine will return the results that may be similar to the input image query from other website but not only from the shopping website that the user wants. Therefore, this intelligent image search engine with AI-based similarity detection for web application enable user to only search the image which only return results from a specific database but not from Internet.

The image search engine existed nowadays are more on keyword-based approach which need user to type some text or keyword to find the image based on the keyword given. It is using the keyword to describe the content of the image and retrieve the images. However, this approach will bring problem that the results shown may not be accurate. This is because image retrieval is difficult to specific the keyword or text which use to describe the content of images

as some difference in annotation will change the retrieval result. Therefore, intelligent image search engine with AI-based similarity detection for web application can solve the problem since it will develop as a content-based image search engine.

The textual input for image search engine will also annoying user that sometimes user may not know what word need to describe on the image for input the text query. This is because if the text query is not specific, the search results could be a large number of irrelevant results as a few words cannot fully describe the image content that user wants to find. Thus, Intelligent Image Search Engine With AI-Based Similarity Detection for web application enable user to direct input image as it is content-based search engine so that user no need to use the specific keyword to find the image that they want but just let AI to detect the similarities and return the image results to them.

1.4 Aim and Objectives

The primary goal of this project is to design and implement an advanced image search engine that leverages AI techniques for similarity detection. Users can upload an image, and the system will search a database to find and present all visually similar images. The results will be displayed through a user-friendly web application. In order to achieve this goal, the following objectives had to be met:

1. To develop a web application to facilitate user interaction and integrate the AI-based image search engine into the web application, allowing users to upload images and view search results
2. To implement a deep learning model for extracting features from uploaded images and learning the features of image content
3. To utilize AI techniques to compare the extracted features and determine the similarity between the query image and images in the database and list the results

1.5 Scope and Limitation of Study

1.5.1 Scope

This project focuses on developing a comprehensive content-based image search engine that employs cutting-edge AI techniques for accurate similarity detection. The AI technique will be use to compare the similarity of query image and images in the database and then retrieve similar images from the database. This image search engine will be implemented in web application which is user-friendly to allow user input image query and show the results in descending order of similarity. There are several vital features which are the fully functional intelligent image search engine, web application featuring an intuitive interface for image uploads and result display and accurate AI-based similarity detection for query image and images in database.

1.5.2 Limitation

There are limitation of this study which the user only can input image query but no text and keyword since it is content-based and not keyword-based search engine. The return results will only show the images and do not include other information such as website link as the image results will be from database but not Internet and the return result images will based on the database and will not give result images from Internet. Since it retrieves images from the database, search results may not be displayed if the input image does not correspond to any images stored in the database due to there is fixed number of images in database.

1.5.3 Tools

The tools that will be use in develop an intelligent image search engine with AI-based similarity detection are HTML, CSS and JavaScript for develop frontend of web application, RESTful API as intermediary between the frontend and backend of web application, OpenCV as software library for image processing tasks, TensorFlow or PyTorch as deep learning framework and MySQL or MongoDB as database to store the datasets.

1.5.4 Target Users

The target user for this intelligent image search engine with AI-based similarity detection for web application is people who want to input image as query in order to search the image instead of using keyword-based approach and get the results from specific place but not from the Internet that show the results in different websites.

1.6 Proposed Solution

An intelligent image search engine with AI-based similarity detection for web application will be created in this project to enable users to upload an image, and present similar images through a web application. Figure 1.2 shows the overview of the intelligent image search engine with AI-based similarity detection for web application. First, user will be able to input their image query in user interface of web application and the image will be pass to NumPy and PIL for image processing through RESTful API as intermediary between frontend and backend. Then, the processed image data will be performing neural network inference by TensorFlow or PyTorch and query the MySql or MongoDB by using the output from the neural network to search for similar images via API. While the highest similarity of image with the input image query had been found, API retrieves the image from database and return as result to web application for display to the user.

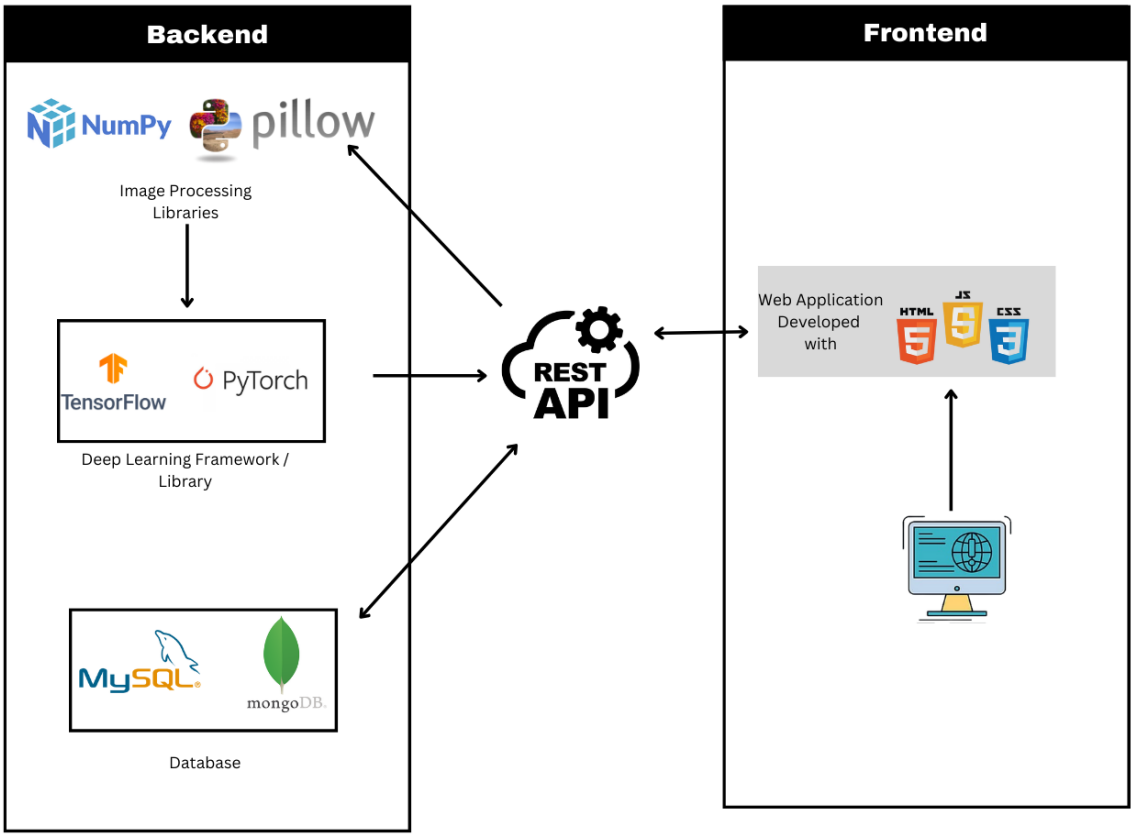

Figure 1.2 Overview of Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application

Chapter 2
Literature Review

2.1 Introduction

Several important pieces of knowledge need to be explained to achieve this project's objectives. In this chapter, several similar image search engines that operates as web application will be discussed. Next, AI algorithm and deep learning techniques will be explored to facilitate the selection of suitable methods to accomplish corresponding features in image search engine with AI-based similarity detection for web application. Furthermore, five base models that based on convolutional neural networks will be discussed, followed by Siamese Network review and lastly One-Shot Learning to have an understanding of how the backend works by looking at the models and techniques. By having a review on others' similar applications and techniques for backend, it is giving help to gain ideas on how to design an image search engine with AI-based similarity detection for web application.

2.2 Similar Applications and Related Works

In this section, three relevant image search engines that operates as web application were selected from browser, and their design and functionality were analyzed. The project can cover more usage scenarios and finds more functional and non-functional requirements by reviewing other similar image search engines that operates as web application.

2.2.1 TinEye

TinEye is a content-based image search engine that operates as web application developed by Idee Inc.. TinEye provides functionality that allows users to search for similar images by uploading image or providing its URL. Figure 2.1 presents the user interface of the TinEye image search engine that operates as web application.

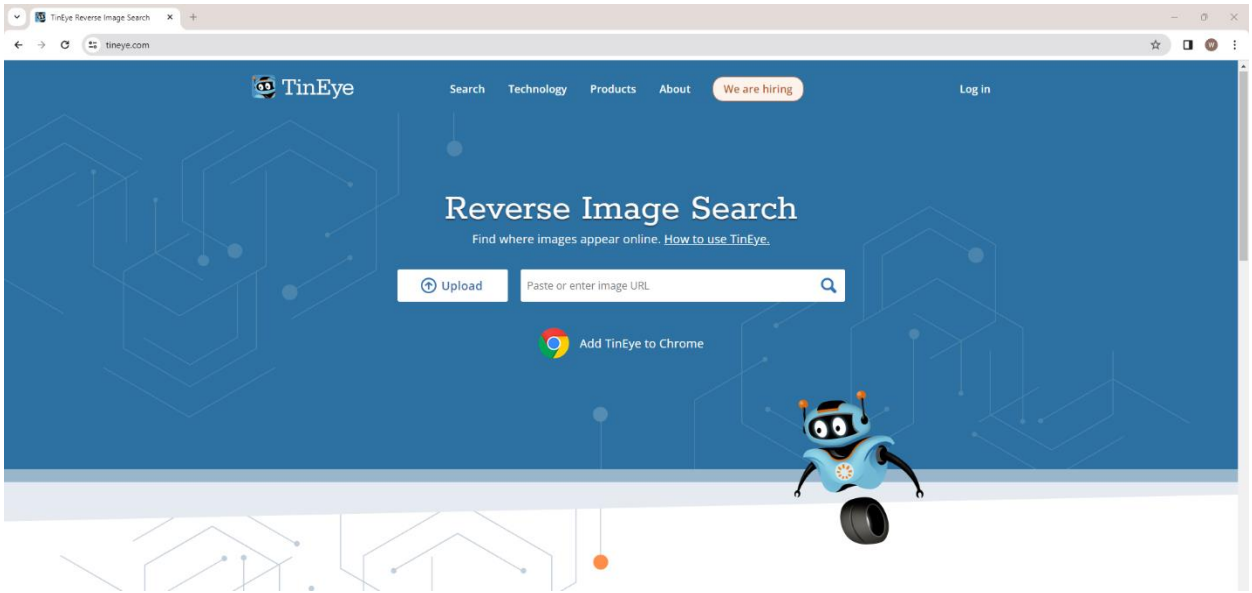

Figure 2.1 User Interface of TinEye

Main Features

- Upload Image for image searching purpose

Users can upload image for searching the similar image in TinEye. By receiving the image upload by user, TinEye image search engine will return the similar images as results to the users.

- Upload URL for image searching purpose

Users can upload URL which is the web address that specifies the location of image on the internet for searching the similar image in TinEye. By receiving the URL upload by user, TinEye image search engine will access the image from the URL and return the similar images as results to the users.

- Find Similar Images

TinEye image search engine will receive image or URL input from the users. Based on the image queries, TinEye image search engine will search the similar images in its database and return the most similar images as results to users.

- Identify Source of image

TinEye image search engine will return the similar images to users based on the image or URL input by users. The image results will also state the source of the results which is the website of the image located.

Evaluation of TinEye

TinEye, as an image search engine by content-based approach, is a good attempt with a successful outcome. By using TinEye image search engine web application, users can find the similar image results of their image queries. TinEye image search engine with content-based approach will not require users to consider which keyword or text need to input for search the image they want. Users can directly input the image or URL then can search for the similar image result which improve the convenience of users. Several features in TinEye are advantageous and suitable to implement in this project.

The first feature is the upload image for searching purpose. It can call as content-based approach which users can direct input the image as query. There will have a place to notice users that they can input the image there and can perform the image search function. When users drag the image into the place or upload the image, the searching function will automatically perform and then return the results of how many images match with the image query. This feature will improve the user experience and let users feel convenient due to users no need to think about the keyword or text need to type to describe the image so that it can return more accurate image results to them.

The second feature is find the similar images. TinEye image search engine will based on the image input by users and return the similar images that match with the image query. It will perform similarity detection on both the image query and images in its database and only output the results that have almost high accuracy similarity. This feature is useful in content-based approach as it can return the results based on similarity but different from keyword-based approach which may misunderstand the description of keyword and return irrelevant images.

However, there are some differences in TinEye with the intelligent image search engine with AI-based similarity detection for web application which this project intended to develop. Those differences are TinEye also allow users to input URL for image searching purpose but not only the image input. The results shown by TinEye will also give the information of images which are the website of the images located at but not just from one specific site. In conclusion, despite the slight difference in requirements, TinEye's user interface design and the features of image input and search similar images can be referenced and adapted to this project.

2.2.2 Google Lens

Google Lens is a content-based approach image search engine which implement image recognition technology. It allows users to search information of images and find the similar image by input image or provide URL. Figure 2.2 presents the user interface of the Google Lens in Google browser.

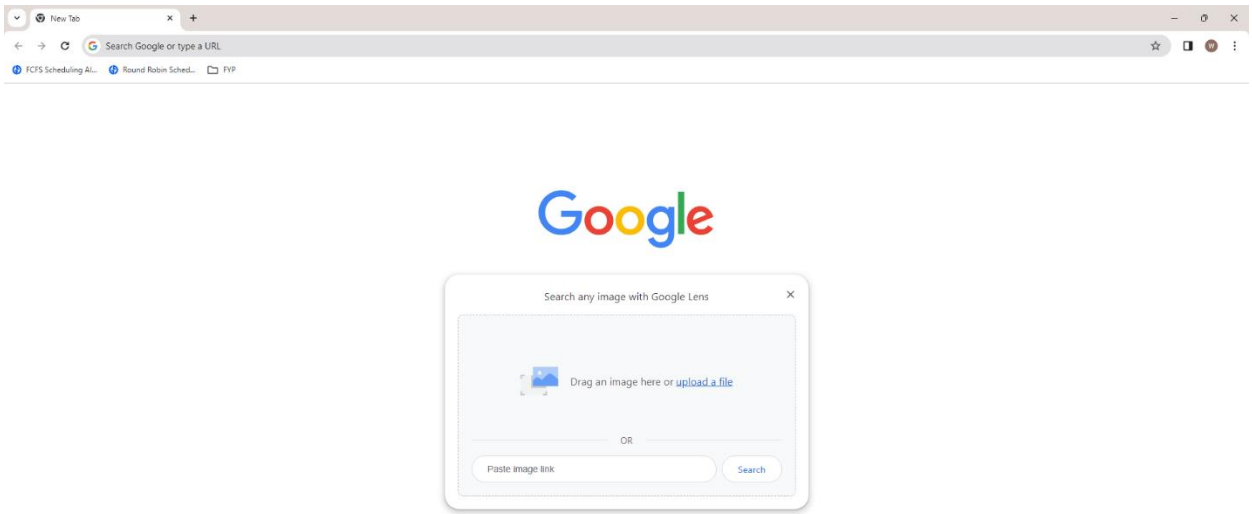

Figure 2.2 User Interface of Google Lens

Main Features

- Search using images

Users allow to drag or input image into specific place in Google Lens and then the search function will automatically perform. Google Lens will return the information of image and list

out similar images which match the image query. This feature will make users easy to get the information of the image and also find the similar image from other websites.

- Search using image link

Users can input image link which also known as URL in the provided place and then click the search button to perform search functionality. Google Lens will get the image from the provided link and then return the information of the image and similar images to the users.

- Image Recognition

Google lens has the functionality of image recognition which it can extract the feature and pattern of the input image query from users and obtain the information of the image. By using the extracted features to compare with other images, it will get the similar image from the comparation and able to return then most accurate similar image results to user.

- Translation

Google lens provide the feature of translation which it can recognize the text in the image and translate the text to other languages. Users can upload the image of document and input the image in Google Lens and it can direct translate the text of the document to other language that the user want.

Evaluation of Google Lens

Google Lens is an excellent content-based image search engine which use the image recognition technology. Google has a keyword-based approach image search engine which allow users to input keyword or text to search the image depend on the description of keyword. However, it has probability to return irrelevant image results to the users, thus Google comes out with a solution which is content-based image search engine, Google Lens. There are many users nowadays using Google as browser to search for information so this Google Lens also famous and familiar to the users and try to use it for image searching purpose.

One of the features of Google Lens is the ability to allow users to upload image in specific place which is to receive the query of users in form of image and return information to the users as the normal functionality of search engine but now in the form of image search

engine. Users need to drag or upload the image from their site and then put into Google Lens, when it receives any image input it will directly perform search function and return with the information of the image and similar images which match the input image. This can advance user experience that users can directly search the image they want by only input the images but no need use any word to describe on the image which sometimes may using the wrong word and text cause the irrelevant results.

Other than that, Google Lens has use the image recognition technology which can increase the accuracy on similarity detection between the image query and other images which intend to return as results output. It is a field of AI which allow Google Lens to extract the feature of the input image and then compare the information with other images. Therefore, with the comparison of features and patterns information it is efficient to find a high similarity images and return back to users.

In conclude, Google Lens is excellent in content-based image search engine as it has implemented AI technique in image similarity detection. The user interface design and features are well worth learning. However, there also difference in Google Lens with the intelligent image search engine with AI-based similarity detection for web application which intended to develop in this project which is the output results of the images are from whole source of Internet. It will not return image from specific site and need users to define themselves which site's image that they want. Google Lens also show too much information about the image which not just directly return the image as result.

2.2.3 Yahoo Image Search

Yahoo Image Search Engine is a keyword-based approach image search engine which require keyword input to perform search functionality. Figure 2.3 presents the user interface of Yahoo Image Search.

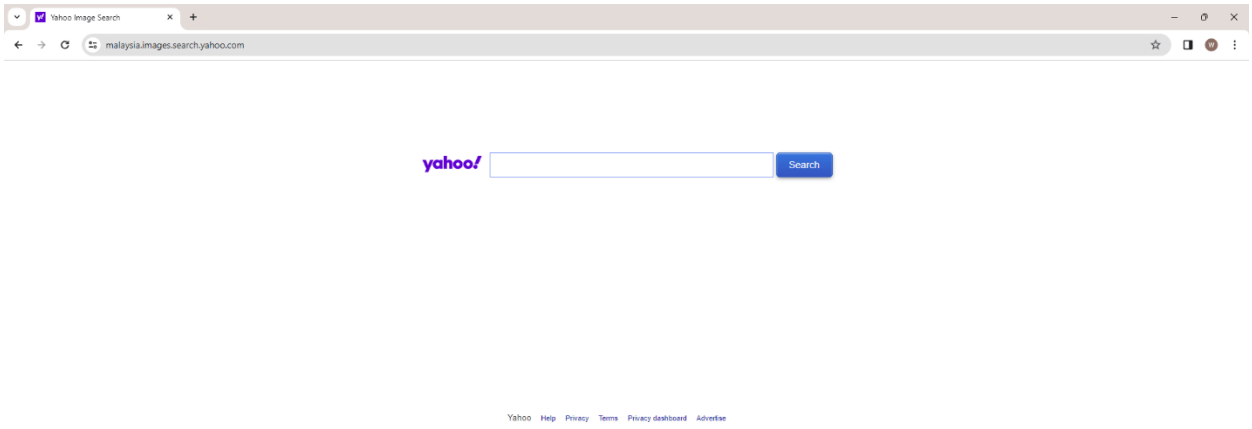

Figure 2.3 User Interface of Yahoo Image Search

Main Features

- Input keyword for image searching purpose

Users allow to input keywords or text in search box that is provided in Yahoo Image Search and click the search button to perform image search function. Yahoo Image Search will then show the image results to the users based on the keyword input by the users.

- Filtering Options

Yahoo Image Search will show the results after users input the keyword and click the search button. Users can apply filter functions such as size, colour, type and layout to the results show by Yahoo Image Search to get the image that meet their needs.

- Preview Thumbnails

Yahoo Image Search will return the image results that only display the image without showing other information. This will let users easy to find the image that they want and click on the image then only will show the website link that the image located.

Evaluation of Yahoo Image Search

Yahoo Image Search is a keyword-based approach image search engine and it shows excellent in this image search function. At first, users need to type in keyword in search box that is provided in Yahoo Image Search and click the search button to perform search function. Users

need to determine what keyword need to type so that it can search the image that what users are exactly want. From the description of the keyword, Yahoo Image Search will show the image results that are related to the keyword.

The main feature of Yahoo Image Search is to return image as searching results to the users. Any query input in search box will only get image results but not the result as the normal search engine which is the website. Yahoo Image Search will analyze the input of users and then based on the description to find the related images and return them. However, the approach uses in Yahoo Image Search is keyword-based and it does not provide content-based approach which mean that users need to think about the keyword themselves to find the image such as features and patterns but cannot directly input an image and let image search engine analyze the features of the image.

Furthermore, Yahoo Image Search will return the image results as thumbnails which is a good preview result method to users. By showing the images only to the users, users can use less time to search for the image that they want from the numerous of image results. There are no information such as website to get the image but users can show this information by click on the image to show details of the image.

In conclude, Yahoo Image Search can be defined as traditional image search engine because it still using keyword-based approach which require user to input keyword in the purpose of searching image but this may make user inconvenience that users need to identify themselves which keyword is correct to find the correct image that they want. Users sometimes may annoying that the image results are wrong when they use the keyword that they may think it is right. Even users had input the correct keyword, it still has the possibility to return irrelevant results due to it is difficult to know the features and patterns of the image need by users by only given the keyword.

2.3 Comparison of Image Search Engine with Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application

	TinEye	Google Lens	Yahoo Image Search	Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application
Image Input	✓	✓	×	✓
Keyword Input	×	×	✓	×
URL input	✓	✓	×	×
Return Image from database	✓	×	×	✓
Return Image from Internet	×	✓	✓	×
Similarity Detection	✓	✓	×	✓
Original source of image	✓	✓	✓	×
Return Image Only	×	×	×	✓

Table 2.1 Comparison of Image Search Engine with Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application

2.4 AI Algorithm and Deep Learning Technique

Algorithm is the mathematical instructions which is the step-by-step procedure for calculations. It is used for calculation, data processing, and automated reasoning. This algorithm is apply to machine or computer as an instruction manual to let them know what to do and when to do it. In the other words, without this algorithm given the information to the machine, there would be no way to start the programming process. Algorithm always being our sides when we are using a machine which carry out specific tasks. (Rock, 2021) However, algorithm is fixed that it will complete the task based on the fixed instructions until the end.

AI algorithm is almost the same with algorithm that give instruction and work on specific task but the difference is this AI algorithm will learn on the task and gain experience on each task, then improve its performance such as decision-making processes or recognize patterns in data. (Rock, 2021) The more times of using the AI algorithm, the greater it able to notice user individual preferences. There are different AI algorithms nowadays such as Clustering algorithms, KNN and Neural Networks. One of the most famous and often used AI algorithms in conjunction with deep learning techniques is Convolutional Neural Networks (CNNs).

2.4.1 Convolutional Neural Networks(CNNs)

Neural networks are a subset of machine learning and they are at the heart of deep learning algorithms. They have different node layers which are input layer, some hidden layers and output layers. Each layer has the nodes which has associated weight and threshold and the node will only activated when its output is above specified threshold value then the node will send data to the next layer of the network. Convolutional Neural Network is one of the neural network which more focus on image, speech, or audio signal inputs. The layers in CNNs are convolutional layer, pooling layer and fully-connected(FC) layer. With these layers in CNNs, it increases in its complexity, identifying greater portions of the image. CNNs performs its task as in the earlier layers, it work on simple features and follow by going to the next layer, the simple features will combine and finally identifies the intended object. (IBM, 2023) Figure 2.4 shows the overview of Convolutional Neural Networks(CNNs).

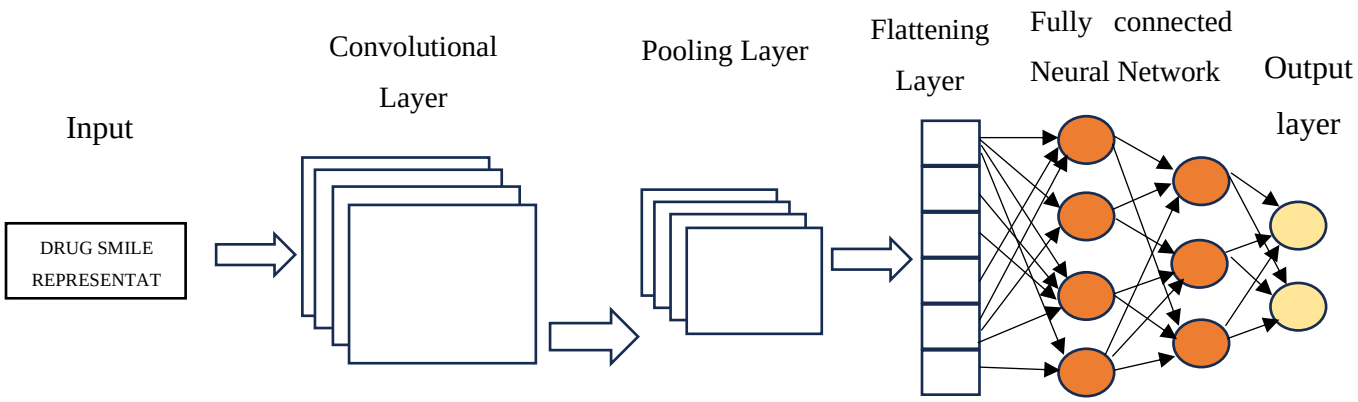

Figure 2.4 Overview of Convolutional Neural Networks(CNNs)

2.4.2 Base Model of Convolutional Neural Networks (CNNs)

2.4.2.1 LeNet

LeNet is one of the neural networks which using convolutional neural networks architectures and it contains the basic modules of deep learning which are convolutional layer, pooling layer, and full link layer. LeNet has a total of eight layer which includes the input layer but normally the input layer does not count as the network structure. The other seven layers are C1 layer, S2 layer, C3 layer, S4 layer, C5 layer, F6 layer and Output layer. It will have three convolution operation in C1 layer, C3 layer and C5 layer to extract features from the input image. Two pooling operations in S2 layer and S4 layer to reduce the spatial dimensions of the feature maps while preserving important features. Two specific operations in F6 layer and Output layer to perform classification based on the extracted features and also combine the features learned from the previous layers and map them to the output classes to enable the network to make predictions. The flow of the layers is input layer, C1 layer, S2 layer, C3 layer, S4 layer, C5 layer, F6 layer and Output layer. LeNet is very efficient for handwritten character recognition and its convolutional layer has fewer parameters which is determined by the main characteristics of the convolutional layers such as local connection and shared weights. (Varshney, P., 2020) Table 2.2 shows the parameter of each layer of LeNet and Figure 2.5 shows the overview of LeNet.

Layer		**Feature Map**	**Size**	**Kernel Size**
Input	Image	1	32 * 32	-
1	Convolution	6	28 * 28	5 * 5
2	Average Pooling	6	14 * 14	2 *2
3	Convolution	16	10 * 10	5 * 5
4	Average Pooling	16	5 * 5	2 * 2

5	Convolution	120	1 * 1	5 * 5
6	FC	-	84	-
Output	FC	-	10	-

Table 2.2 Parameter of Each Layer of LeNet

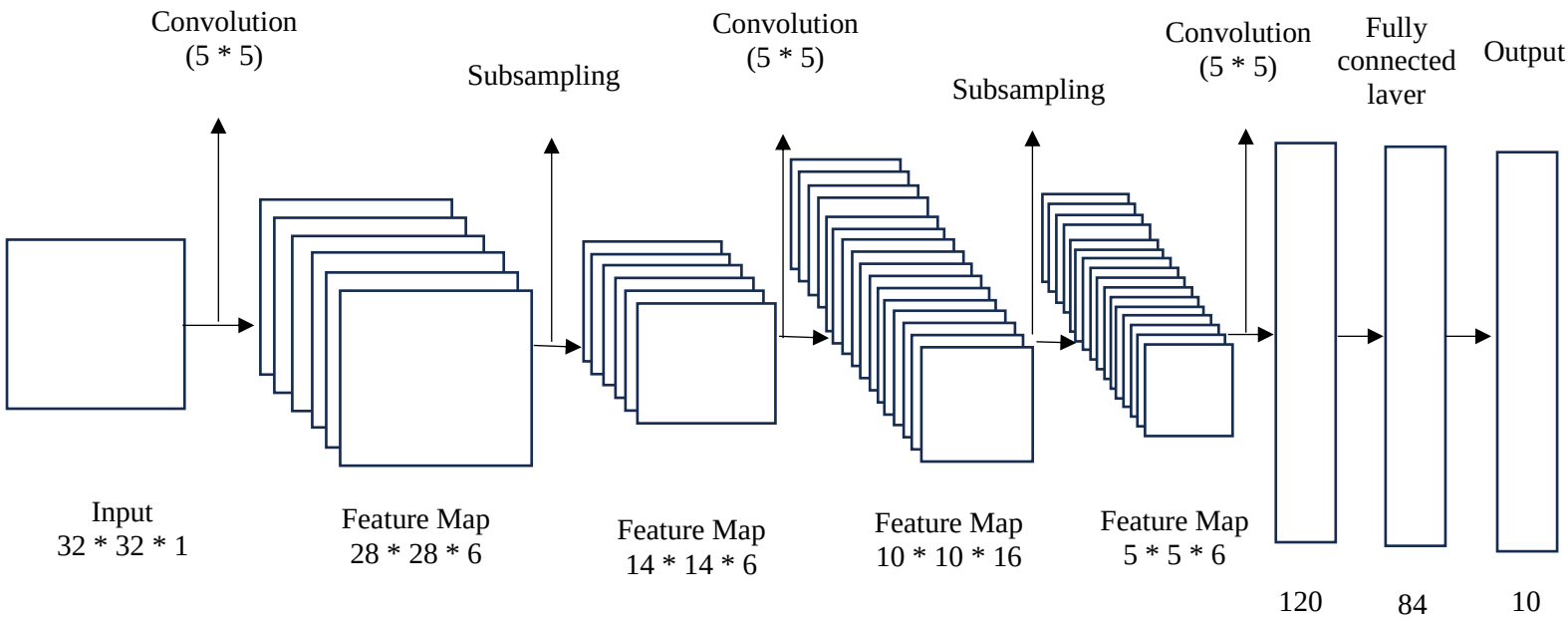

Figure 2.5 Overview of LeNet

2.4.2.2 AlexNet

AlexNet is one of landmark convolutional neural networks architecture that have give advancements in image classification and also is the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet also comprises fundamental components of deep learning which are convolutional layers, pooling layers and fully connected layers. AlexNet has a total of eight layers included the input layers and the other seven layers are Conv1, Conv2, Conv3, Conv4, Conv5, FC6, FC7, and FC 8layers. It also consists of Pool1, Pool2 and Pool3 layers but they are not count in the total layers because they are as overlapping

max pooling applied within certain convolutional layers. In AlexNet, Conv1, Conv2, Conv3, Conv4, Conv5 will perform convolutional operation to extract hierarchical features from the input images. Then, Pool1, Pool2 and Pool3 will perform pooling operation for spatial dimension reduction for preserving important features. After that, FC6, FC7 and FC8 which are the fully connected layers are to combine the features learned from the previous layers and then perform classification tasks to enable the network to make predictions. AlexNet had introduced new innovations which is the use of rectified linear units (ReLU) for activation functions, overlapping pooling, and dropout regularization. These techniques help to increase the performance and efficiency in image classification tasks and also give big impact on the field of deep learning and computer vision. (Varshney, P., 2020) Table 2.3 shows the parameter of each layer of AlexNet and Figure 2.6 shows the overview of AlexNet.

Layer	Neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 * 227 * 3	-
Conv 1	96	11 * 11	4	-	55 * 55 * 96	ReLU
Pool 1	-	3 * 3	2	-	27 * 27 *96	-
Conv 2	256	5 * 5	1	2	27 * 27 * 256	ReLU
Pool 2	-	3 * 3	2	-	13 * 13 * 256	-
Conv 3	384	3 * 3	1	1	13 * 13 * 384	ReLU
Conv 4	384	3 * 3	1	1	13 * 13 * 384	ReLU
Conv 5	256	3 * 3	1	1	13 * 13 * 256	ReLU
Pool 3	-	3 * 3	2	-	6 * 6 *256	-
Dropout 1	Rate = 0.5	-	-	-	6 * 6 * 256	-

Table 2.3 Parameter of Each Layer of AlexNet

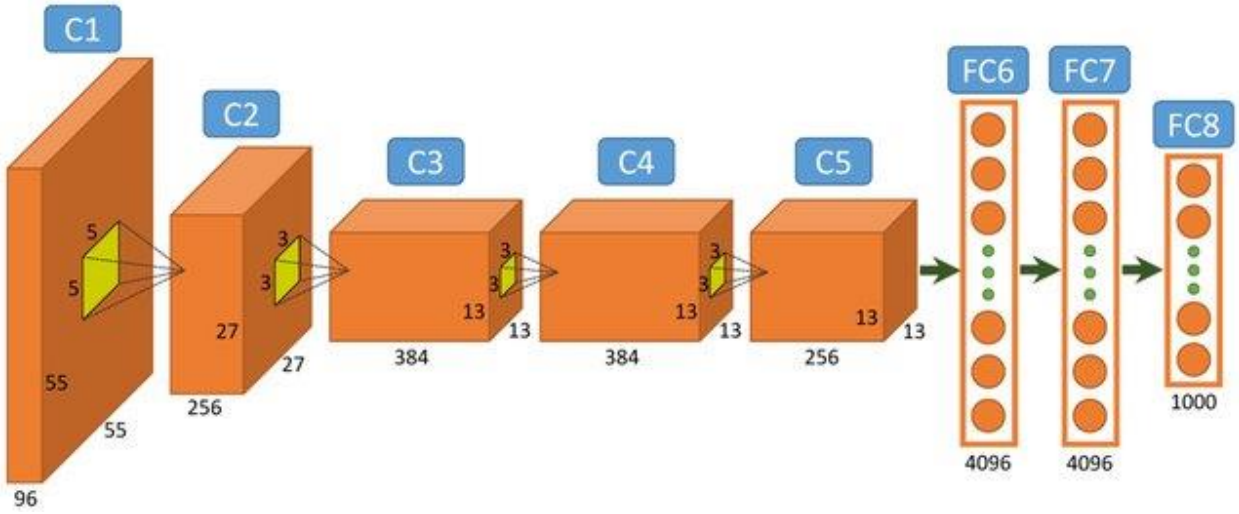

Figure 2.6 Overview of AlexNet (Robert Mash & Nicholas Becherer & Brian Woolley & John Pecarina, 2016)

2.4.2.3 GoogleNet

GoogleNet is using convolutional neural network architecture, and it is the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. GoogleNet also give new innovative concepts to improve the performance of convolutional neural networks. GoogleNet has multiple inception modules, each comprising parallel convolutional layers of different filter sizes which are 1 * 1, 3 * 3 and 5 * 5 and pooling layers. These parallel layers give advantage to the network to capture features at different spatial scales efficiently and the 1 * 1 convolutions also good to reduce the dimensionality of feature maps before applying larger convolutions which help to decrease computational complexity. GoogleNet has a deeper and wider architecture compared to the other CNNs base model with its total of 22 layers. Although it has more layers, but it still maintains the efficiency by using inception modules and global average pooling. The global average pooling is replaced the fully connected layers at the end of the network, and it uses to compute the average value of each feature map that across its entire spatial extend and give the result in a fixed-length vector representation of the features. Global average pooling had reduced the number of parameters and mitigates overfitting. The flow of the layers in GoogleNet is multiple inception modules followed by global average pooling and a softmax output layer for classification. GoogleNet has give improvements in image classification accuracy and also maintain the computational efficiency, paving the way

for subsequent advancements in deep learning architectures. Table 2.4 shows the parameter of each layer of GoogleNet and Figure 2.7 shows the overview of GoogleNet.

Type	Stride	Output size	depth	1 * 1	3 * 3 reduce	3 * 3	5 * 5 reduce	5 * 5	Pool proj	params	ops
Convolution	7 * 7 / 2	112 * 112 * 64	1							2.7K	34M
Max pool	3 * 3 / 2	56 * 56 * 64	0								
Convolution	3 * 3 / 1	56 * 56 *192	2		64	192				112K	360M
Max pool	3 * 3 / 2	28 * 28 *192	0								
Inception (3a)		28 * 28 * 256	2	64	96	128	16	32	32	159K	128M
Inception (3b)		28 * 28* 480	2	128	128	192	32	96	64	380K	304M
Max pool	3 * 3 / 2	14 * 14 *480	0								
Inception (4a)		14 * 14 * 512	2	192	96	208	16	48	64	364K	73M
Inception (4b)		14 * 14 * 512	2	160	112	224	24	64	64	437K	88M
Inception (4c)		14 * 14 * 512	2	128	128	256	24	64	64	463K	100M
Inception (4d)		14 * 14 * 528	2	112	144	288	32	64	64	580K	119M
Inception (4e)		14 * 14 * 832	2	256	160	320	32	128	128	840K	170M
Max pool	3 * 3 / 2	7 * 7 * 832	0								
Inception (5a)		7 * 7 * 832	2	256	160	320	32	128	128	1072K	54M
Inception (5b)		7 * 7 * 1024	2	384	192	384	48	128	128	1388K	71M
Avg pool	7 * 7 / 1	1 * 1 * 1024	0								
Dropout (40%)		1 * 1 * 1024	0								
Linear		1 * 1 * 1000	1							1000K	1M
Softmax		1 * 1 * 1000	0								

Table 2.4 Paramter of Each Layer of GoogleNet

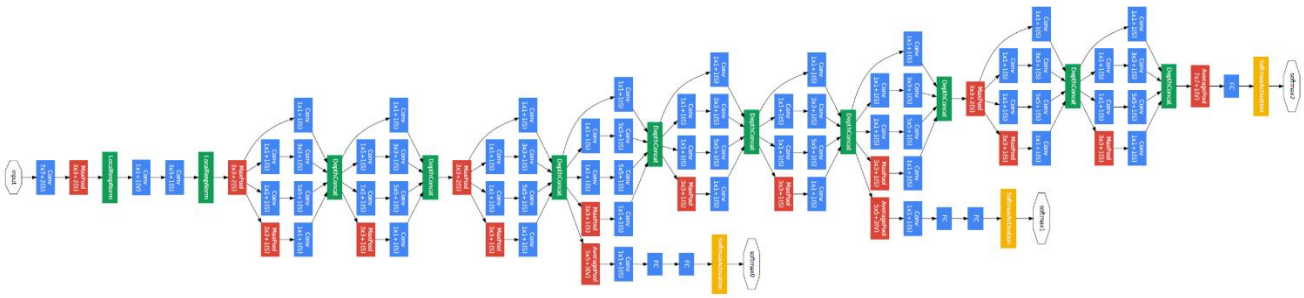

Figure 2.7 Overview of GoogleNet (GeeksforGeeks, 2020)

2.4.2.4 VGGNet (Visual Geometry Group Network)

VGGNet, also known as Visual Geometry Group Network, uses CNN architecture and it was the second-best performing network in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. Even though it fails to win ILSVRC in 2014, it is still one of the most influential models for deep learning. This VGGNet has many convolutional layers followed by max-pooling layers with small 3 * 3 filters which are used to capture spatial hierarchies of features in input image. This is a normal deep convnet structure with the “deep” meaning the number of layers in VGG-16 and VGG-19 which have 16 and 19 layers respectively. VGG-16 and VGG-19 differ in terms of their depth or simply put, how many levels of convolutional and fully connected they are. There are thirteen convolutional layers for VGG-16 with three fully connected layers while the VGG-19 contains sixteen convolutional as well as three fully connected layers. In this model, some convolutional features followed by max-pooling ones plus classification layers based on fully connected networks are used for the design. The output layer usually consists of softmax activation for multi-class classification tasks. VGGNet had achieved great performance on image classification tasks based on its straightforward architecture and uniform structure to make it easy for understanding and implement. It is a powerful tool for image recognition tasks and contributing to advancements in deep learning research and applications. (Boesch, G., 2021) Below are the table and figure to show one of the VGGNet architecture which is VGG-16. Table 2.5 shows the parameter of each layer of VGG-16 and Figure 2.8 shows the overview of VGG-16.

Layer	Activation shape	Activation size	Parameters
Input	224 * 224 * 3	150528	0
Conv 1	224 * 224 * 64	3211264	1792
Conv 2	224 * 224 * 64	3211264	36928
Pool	112 * 112 * 64	802816	0
Conv 3	112* 112 * 128	1605632	73856
Conv 4	112 * 112 * 128	1605632	147584
Pool	56 * 56 * 128	401408	0
Conv 5	56 * 56 * 256	802816	295168
Conv 6	56 * 56 * 256	802816	590080
Conv 7	56 * 56 * 256	802816	590080
Pool	28 * 28 * 256	200704	0
Conv 8	28 * 28 * 512	401408	1180160
Conv 9	28 * 28 * 512	401408	2359808
Conv 10	28 * 28 * 512	401408	2359808
Pool	14 * 14 * 512	100352	0
Conv 11	14 * 14 * 512	100352	2359808
Conv 12	14 * 14 * 512	100352	2359808
Conv 13	14 * 14 * 512	100352	2359808
Pool	7 * 7 * 512	25088	0
FC 14	4096 * 1	4096	102760449
FC 15	4096 * 1	4096	16777217
Softmax	1000 * 1	1000	4096001

Table 2.5 Parameter of Each Layer of VGG-16

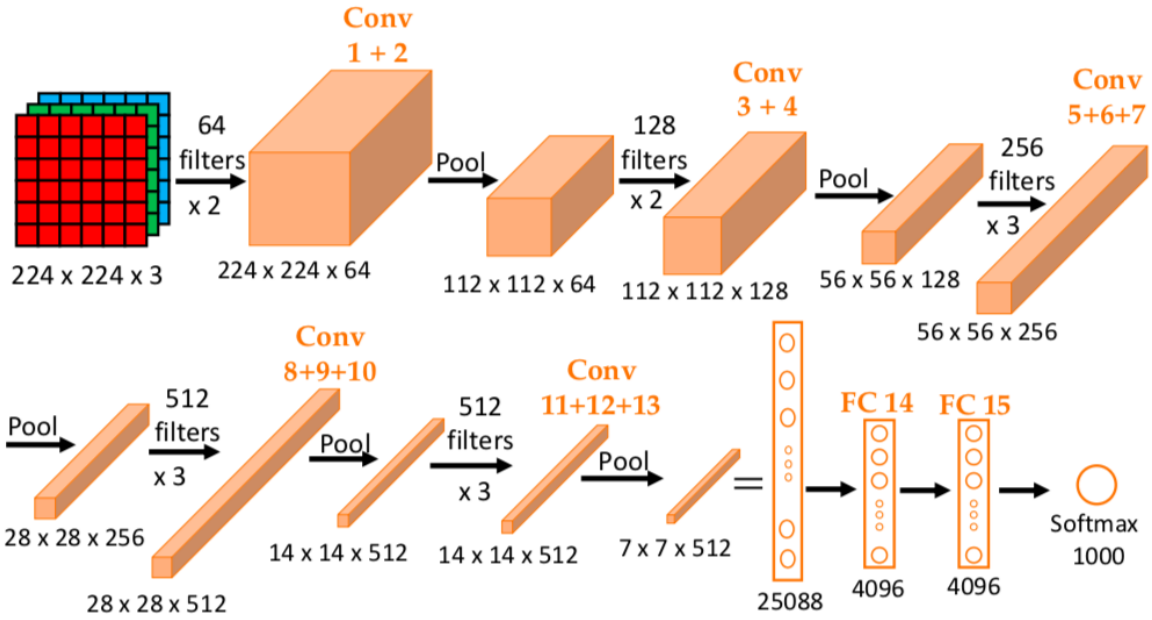

Figure 2.8 Overview of VGG-16 (Reynolds, A.h., n.d.)

2.4.2.5 ResNet (Residual Network)

ResNet which is Residual Network is using CNNs architecture and it also give advancement in deep learning by solve the problem of training very deep neural networks. ResNet introduced new concept of residual learning which is to get use of residual blocks to learn residual functions but not just learn the desired underlying mapping directly. This is done by adding shortcut connections or skip connections that bypass more layers to allow the network to learn residual mapping. ResNet has deep structure with networks ranging from tens to hundreds of layers and its models are easier to train compared to traditional deep neural networks. The skip connections mitigates the vanishing gradient problem by enable gradient flow throughout the network and allow for the successful training of very deep networks. ResNet has many variants such as ResNet-18, ResNet-34 and ResNet-50. Their difference is the number of layers which can show their depth. These models have been trained on ImageNet and widely used as feature extractors as the basis for transfer learning in computer vision tasks. The flow of layers in ResNet is multiple residual blocks then global average pooling and lastly softmax output layer for classification. ResNet has provided solution to the challenge of training very deep neural networks and give way for advancement in this field. (Bangar, S., 2022) Below are the table and figure to show one of the ResNet architecture which is ResNet-18. Table 2.6 shows the parameter of each layer of ResNet-18 and Figure 2.9 shows the overview of ResNet-18.

Layer	Output Size	Parameter
Conv1	112 * 112 * 64	7 * 7, 64, stride 2
Conv2_x	56 * 56 * 64	3 * 3maxpool, stride 2 $\begin{bmatrix} 3 * 3, 64 \\ 3 * 3, 64 \end{bmatrix}$ X 2
Conv3_x	28 * 28 * 128	$\begin{bmatrix} 3 * 3, 128 \\ 3 * 3, 128 \end{bmatrix}$ X 2
Conv4_x	14 * 14 * 256	$\begin{bmatrix} 3 * 3, 256 \\ 3 * 3, 256 \end{bmatrix}$ X 2
Conv5_x	7 * 7 * 512	$\begin{bmatrix} 3 * 3, 512 \\ 3 * 3, 512 \end{bmatrix}$ X 2
Average pool	1 * 1 * 512	7 * 7 average pool
Fully connected	2	512 * 2 fully connection
Softmax	2	Classification results

Table 2.6 Parameter of each layer of ResNet-18

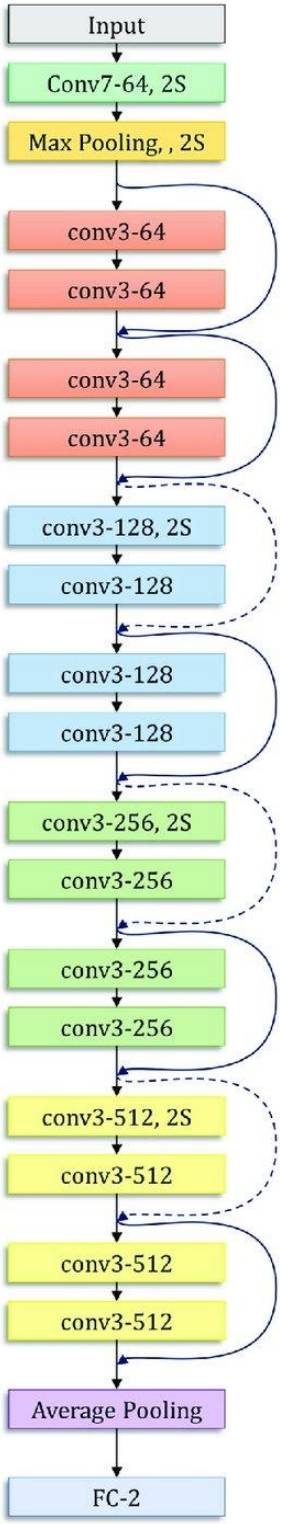

Figure 2.9 Overview of ResNet (Rohit Kundu & Ritacheta Das & Zong Woo Geem & Ram Sarkar., 2021)

2.4.3 Siamese Network

The Siamese Network architecture is a vital tool in the field of neural networks, especially for measuring similarity tasks like image comparison and face recognition. The Siamese Network basically has a unique ability to learn representations of input data within one shared feature space. This is done by using two similar sub-networks also known as Siamese twins that share the same parameters and architectural structure. In this case, those separate inputs are processed by each sub-network but undergo the same sequence of operations to produce the embedding. These embeddings represent the input data concisely and can be compared with each other using any type of distance metric to measure their similarity. The key strength of the Siamese Network is that it can tell between similarities and differences in inputs, by bringing together such inputs in the feature space as those which are close to one another and pushing apart those which are different from each other. Its usefulness includes distinguishing between similar and dissimilar examples, for example in face verification systems or finding similar images. When they are trained, most usually Siamese Networks get optimized using contrastive loss functions or triplet loss functions. These loss functions are important in promoting embeddings that reflect the true relationship between input pairs on the network. That is to say that they encourage embeddings of objects belonging to the same class within a short distance of one another while making sure that embeddings of dissimilar classes fall far apart in embedding space.

The ability of Siamese networks to learn efficiently from little labeled data stands out as one of their notable advantages. This happens because they use positive and negative input pairs in training, which give useful similarity information to the network. Consequently, Siamese Networks are good at generalizing across different input pairs making them highly adaptable and efficient in different domains and tasks. Their versatility has led to their wide usage in machine learning research and applications. Siamese Network is a powerful method for learning similarity metrics that can effectively perform accurate comparisons between similarities across various areas. The importance of this work in enhancing machine learning capabilities towards development in areas such as image analysis, recommendation systems among others cannot be underestimated also judging by the success that Siamese Networks have recorded in a task requiring fine-grained similarity assessments. (J, S.B., 2021) Figure 2.10 shows the overview of Siamese Network.

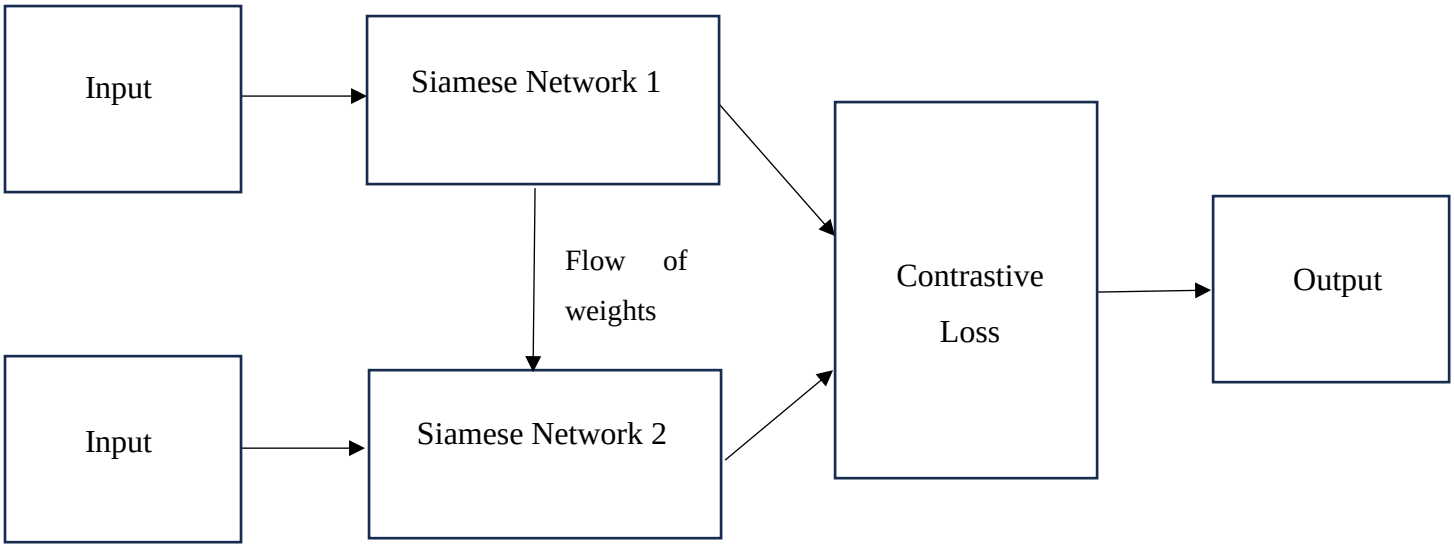

Figure 2.10 Overview of Siamese Network

2.4.4 One Shot Learning

One Shot Learning is a revolutionary concept in machine learning that helps address the major problem of teaching models effectively with limited datasets. This modifies the traditional ways of doing machine learning which require lots of data to understand complex patterns and relationships properly. The primary focus behind One Shot Learning is the ability of models to gain essential knowledge and make generalizations from as few training examples per class as possible, thus avoiding data paucity constraints. In certain instances, there may be a limitation of having enough data due to the limited supply of resources, costs linked to data collection or even time constraints; this is when One Shot Learning comes into play. In such situations, traditional learning methods falter whereas one shot learning serves as a beacon for effectiveness and efficiency. This allows models to easily adapt and learn from few sparse datasets making it an indispensable tool in various fields. Specialized architectures and algorithms specifically designed for efficient handling of sparse data are at the core of One-Shot Learning. For example, Siamese Network is a good illustration that performs well in measuring similarity during comparison task. By sharing parameterization across the Siamese Networks and having a dual subnetwork structure, these networks outperform others in learning discriminative embeddings that generalize the input data. These embeddings serve as the basis

for generalization and predicting on few-shot data by One Shot Learning which captures What is remarkable about this method is how it can capture both similarities between examples as well as differences between them.” similarities intrinsic in contrast with extrinsic ones.

The role of Siamese Networks is to act as a bridge for transferring knowledge in One Shot Learning, and to enable model comparisons in cases of novel examples. The learned compasses produced by Siamese Networks help guide unexplored territories, enabling them to infer similarity even when they haven’t seen this before. This reuse of learned information not only improves predictive powers but also enhances adaptability and robustness across different environments or task domains. The applications that One Shot Learning can be put to are wide-ranging, crossing fields such as computer vision, natural language processing, robotics and more. When it comes to using less data in an effective way, few things can match its adaptability and efficiency as a basic technique for modern machine learning which helps these models absorb knowledge quickly, generalize well and bear up in situations of scarce availability of data. The implications of this research cannot be underestimated because it has the capacity of opening up new vistas for deep learning techniques like one-shot learning. (J, S.B., 2021) Figure 2.11 shows the overview of One-Shot Learning.

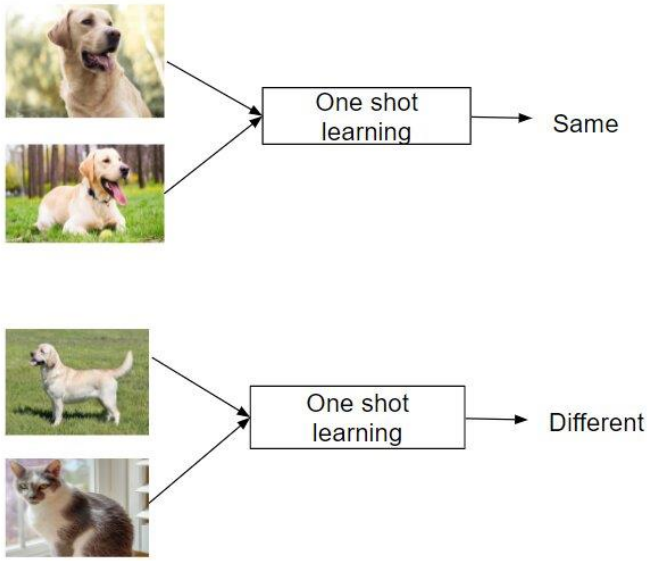

Figure 2.11 Overview of One Shot Learning (Shivaank Agarwal & Ravindra Gudi, 2022)

Chapter 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The most appropriate SDLC methodology for this project will be selected and the reasons for the methodology selection will be discussed, and the detailed implementation flow of the method will be listed in the Work Breakdown Structure. Gantt Chart was provided to state each task's start and end date clearly. Finally, the development tools required for software development will be covered in this chapter.

3.2 System Development Methodology

Iterative methodology was selected for the development of intelligent image search engine with AI-based similarity detection for web application after review on various software development lifecycle methodologies. Figure 3.1 shows the lifecycle of iterative methodology. This is due to its ability to adapt and evolve in dynamic project environments. This image search engine project will implement AI model for similarity detection, thus the challenges that may face in this project primarily related to the development and refinement of AI model. It is difficult to ensure the high accuracy of AI model by just training few times. This is because in order to achieve accuracy in AI, it depends on the quality of data, advanced model architectures and iterative refinement. Therefore, iterative methodology is suitable in this project as its iterative feature allow continuous refinement and improvement of AI model, allow to respond quickly to changes in requirements and receive feedback from user to evaluate the performance of the AI model.

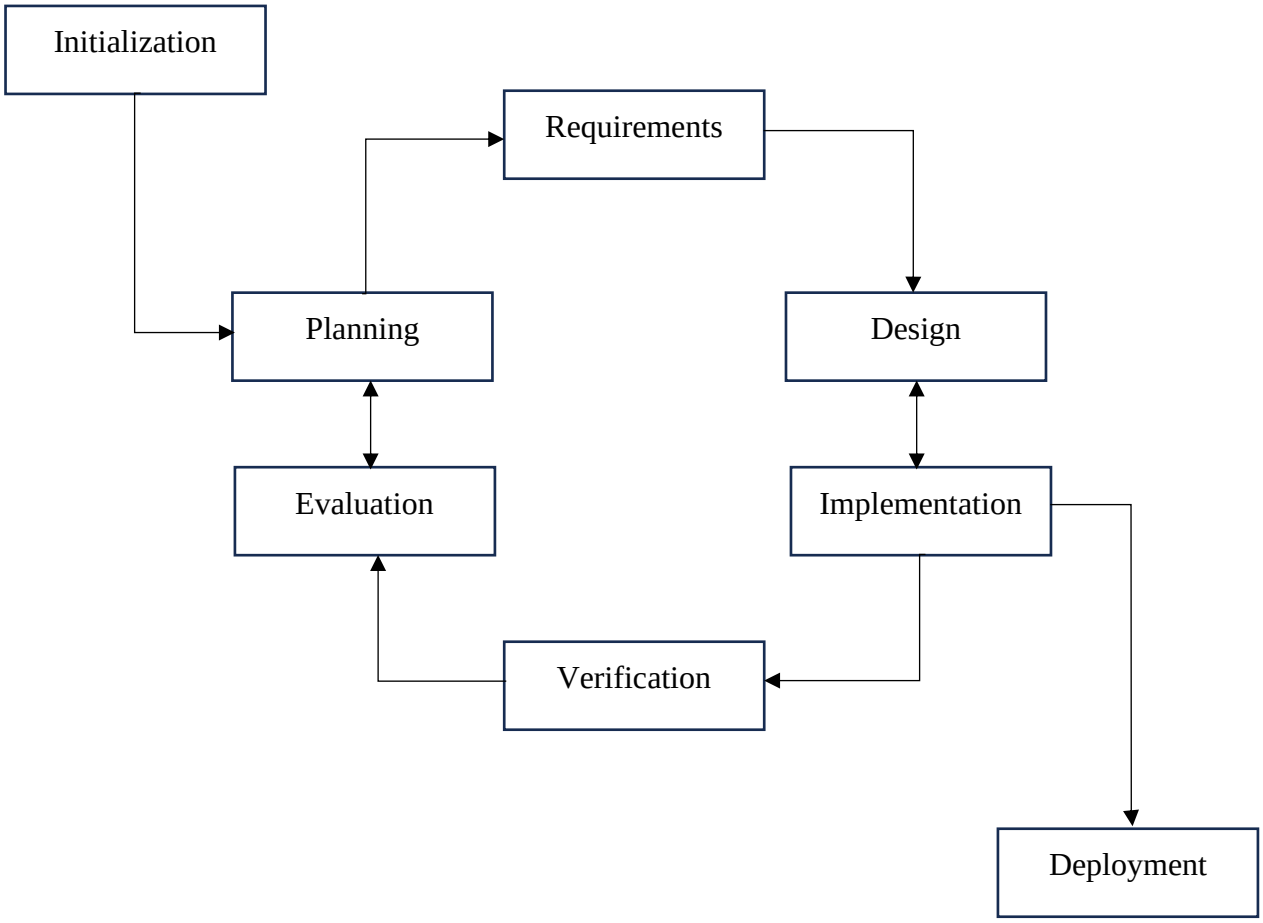

Figure 3.1 Iterative Methodology

After the initial planning phase is completed, this project will conduct several iteration to ensure iterative development and responsiveness to fulfil the requirements. The iteration will be performed five times and each iteration will be approximately 45 to 50 days. Although each iteration has the same SDLC process but different processes will be emphasized.

In the first iteration, the main task is more on gathering requirements and completing the analysis and design of the project. The use case diagram, UI design, dataset for training AI model and AI algorithm exploration will be completed during this sprint. The UI design is then

implemented in real software to be tested and made available to users for feedback. The UI design will be completed using Figma during this iteration. It needs to ensure the selection of colors and fonts for the web application and complete the design of web pages. The design of web page will show the location of where user can input image query and the output of image results. Additionally, the quality of dataset and AI algorithm also need to determine to ensure that the dataset gives high impact on AI model training and the chosen AI algorithm will give high accuracy and efficiency in AI model training.

Since the feedback from users is received in the first iteration, it may not meet the needs and expectations of the users. Therefore, the project still can change the requirements, find new dataset and AI algorithms and design in the second iteration. As for the implementation, web application will be developed to allow the functionality of input image query and output image results. These features will rely on the connection on frontend and backend to send request to backend to retrieve the similar image from database and receive the results by frontend. Therefore, it is important to have good connection between frontend and backend with the database. This phase must complete the development of user interface of web application and enable requests to be sent to the backend and retrieve image from database and return to web application. Testing will also be done to find errors in early state and reduce the risks. The main module of backend will be the implementation of AI model for similarity detection. Before the implementation of AI model, it needs to train the AI model to make it done specific task which is the similarity detection for image. Dataset is required to train the AI and algorithm to ensure the efficiency and accuracy of AI.

In the third and fourth iteration, the requirements and design can still be changed based on user feedback from the previous iteration. However, these iterations will focus on completing other task such as refinement on AI model. It is difficult to use only few times to train a high accuracy AI, it needs to iteratively refine the AI model to achieve high performance and high accuracy on the return results. The AI model can be refined iteratively to improve its capabilities, overcome any shortcomings or deficiencies, and eventually achieve the expected levels of accuracy and performance in similitude detection feature of image search engine. Through this iterative scheme

there is a possibility for continuous perfection and adaptation grounded on user feedback and changing demands thus resulting in a more powerful and efficient solution. Different test cases will be listed and tested one by one to meet as many scenarios as possible.

In the last iteration, it will more focus on testing such as unit tests, integration tests, functional tests and model tests. It is to ensure that the modules can run with each other, and no exceptions are generated, and user acceptance testing will also allow users to use the web application to provide feedback before the official release. The final product will be presented to users. At this point, the image search engine web application development is complete and will be deployed to a production environment.

3.3 Work Plan

Work Breakdown Structure is used to simplify the complex tasks into smaller tasks for easy execution in this project. Figure 3.2, figure 3.3 and figure 3.4 shows the work breakdown structure of the project. The work plan shows how the agile methodology will be conducted. The specific work content and duration of work in each sprint are planned in Gantt Chart. Figure 3.4 and figure 3.5 display the Gantt Chart of the project.

3.3.1 Work Breakdown Structure

Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application

1. Initial Planning

    1.1. Project Planning

        1.1.1. Introduction

        1.1.2. Problem Background Research

        1.1.3. Define Problem Statement

        1.1.4. Define Project Objectives

1.1.5. Define Project Scope

1.1.6. Define Project Solution

1.2. Literature Review

1.2.1. Review On Similar Application

1.2.2. Comparison of Image Search Engine

1.2.3. Review On AI Algorithm and Deep Learning Technique

1.2.4. Review On Siamese Network

1.2.5. Review On One Shot Learning

1.3. Methodology and Workplan

1.3.1. Select SDLC Methodology

1.3.2. Develop Project Workplan

1.3.3. Select Development Tools

2. Iterative Process

2.1. First Iteration

2.1.1. Major Requirements Gathering and Documentation

2.1.2. Analysis & Design

2.1.3. Implementation

2.1.4. Testing

2.1.5. Evaluation

2.2. Second Iteration

2.2.1. New Requirements Gathering and Documentation

2.2.2. Analysis & Design

2.2.3. Implementation

4.1. Complete Report Writing

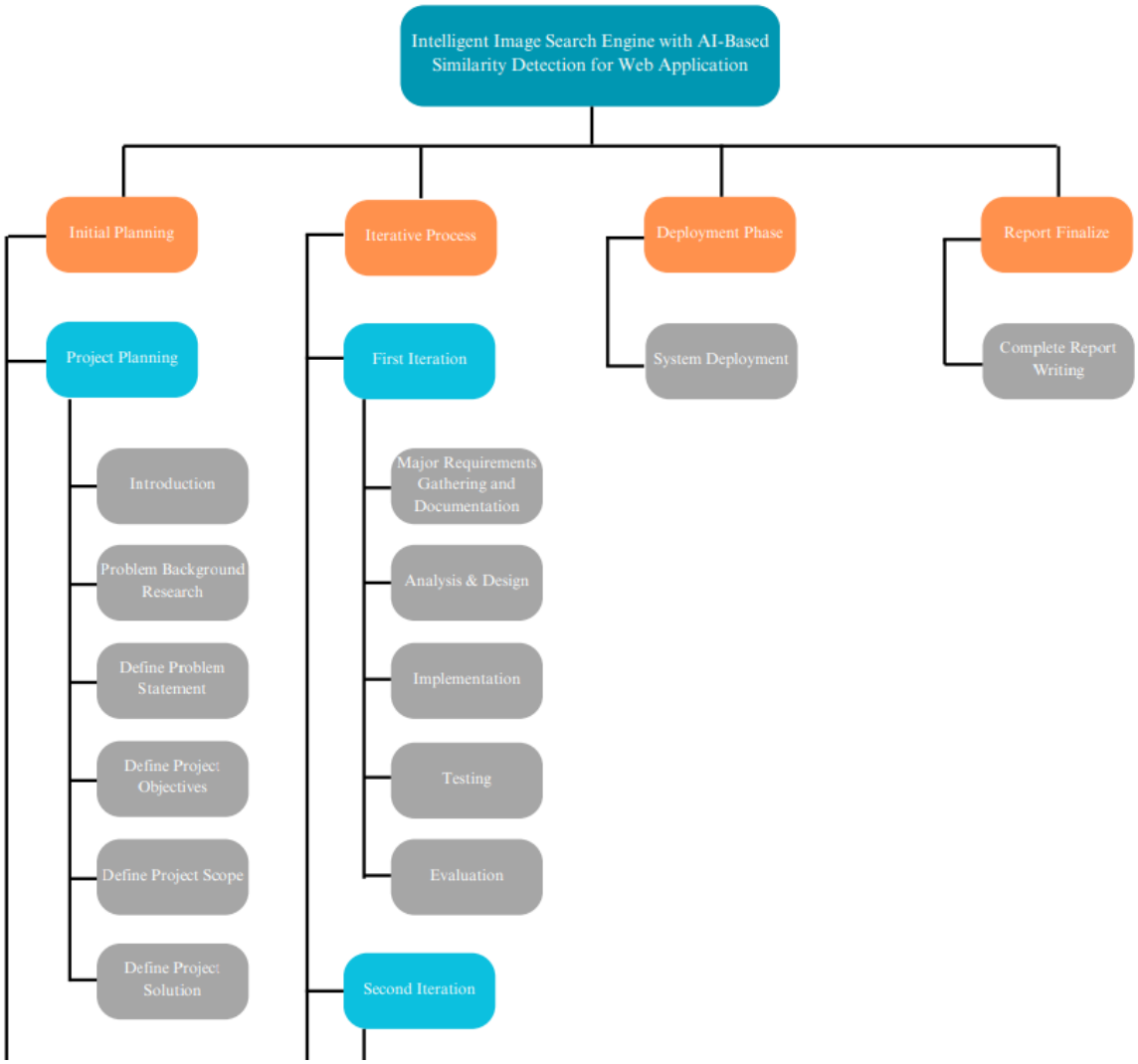

Figure 3.2 Work Breakdown Structure

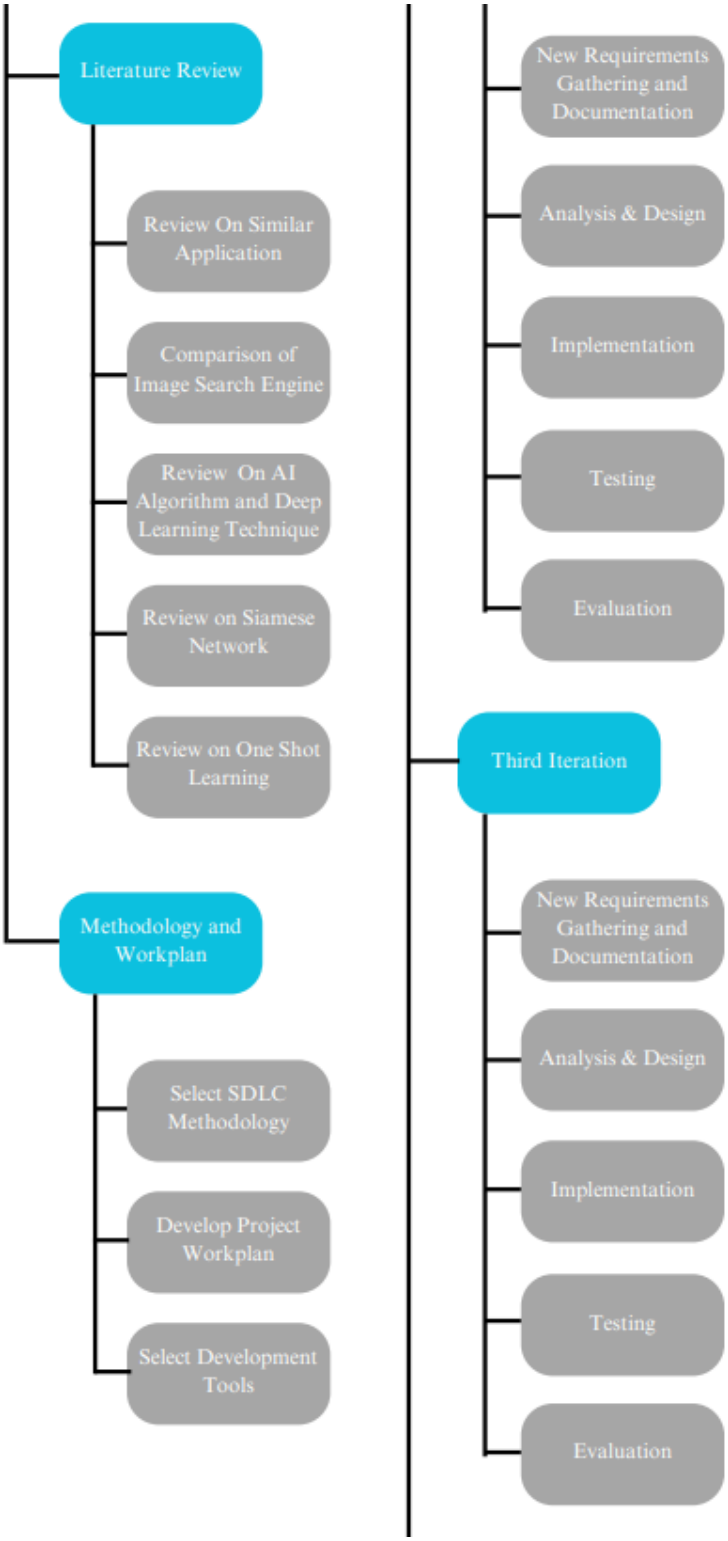

Figure 3.3 Work Breakdown Structure (Continued)

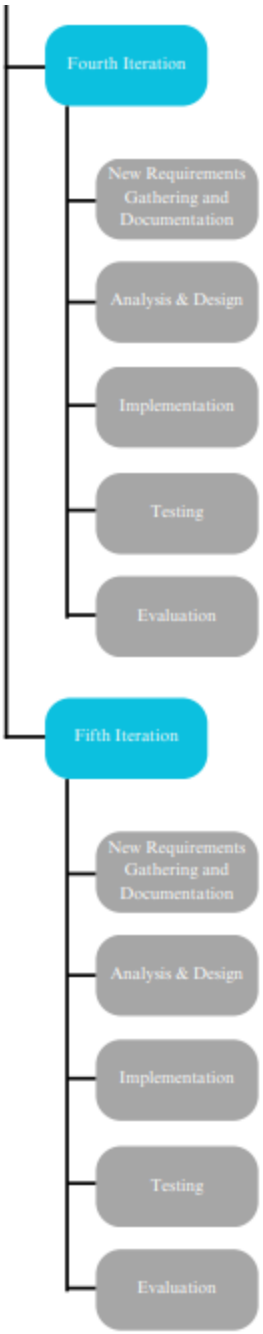

Figure 3.4 Work Breakdown Structure (Continued)

3.3.2 Gantt Chart

Task Name	Duration	Start	End
Initial Planning	77days	29.01.24	14.04.24
Project Planning	25days	29.01.24	22.02.24
Introduction	5days	29.01.24	02.02.24
Problem Background Research	10days	03.02.24	12.02.24
Define Problem Statement	5days	13.02.24	17.02.24
Define Project Objectives	2days	18.02.24	19.02.24
Define Project Scope	1days	20.02.24	20.02.24
Define Project Solution	2days	21.02.24	22.02.24
Literature Reviw	37days	23.02.24	30.03.24
Reviw On Similar Application	12days	23.02.24	05.03.24
Comparison of Image Search Engine	1days	06.03.24	06.03.24
Review On AI Algorithm and Deep Learning Technique	15days	07.03.24	21.03.24
Reviw On Siamese Network	5days	22.03.24	26.03.24
Review On One Shot Learning	4days	27.03.24	30.03.24
Methodology and Workplan	15days	31.03.24	14.04.24
Select SDLC Methodology	7days	31.03.24	06.04.24
Develop Project Workplan	4days	07.04.24	10.04.24
Select Development Tools	4days	11.04.24	14.04.24
Iterative Process	124days	15.04.24	16.08.24
First Iteration	28days	15.04.24	12.05.24
Major Requirements Gathering and Documentation	5days	15.04.24	19.04.24
Analysis & Design	5days	20.04.24	24.04.24
Implementation	12days	25.04.24	06.05.24
Testing	4days	07.05.24	10.05.24
Evaluation	2days	11.05.24	12.05.24

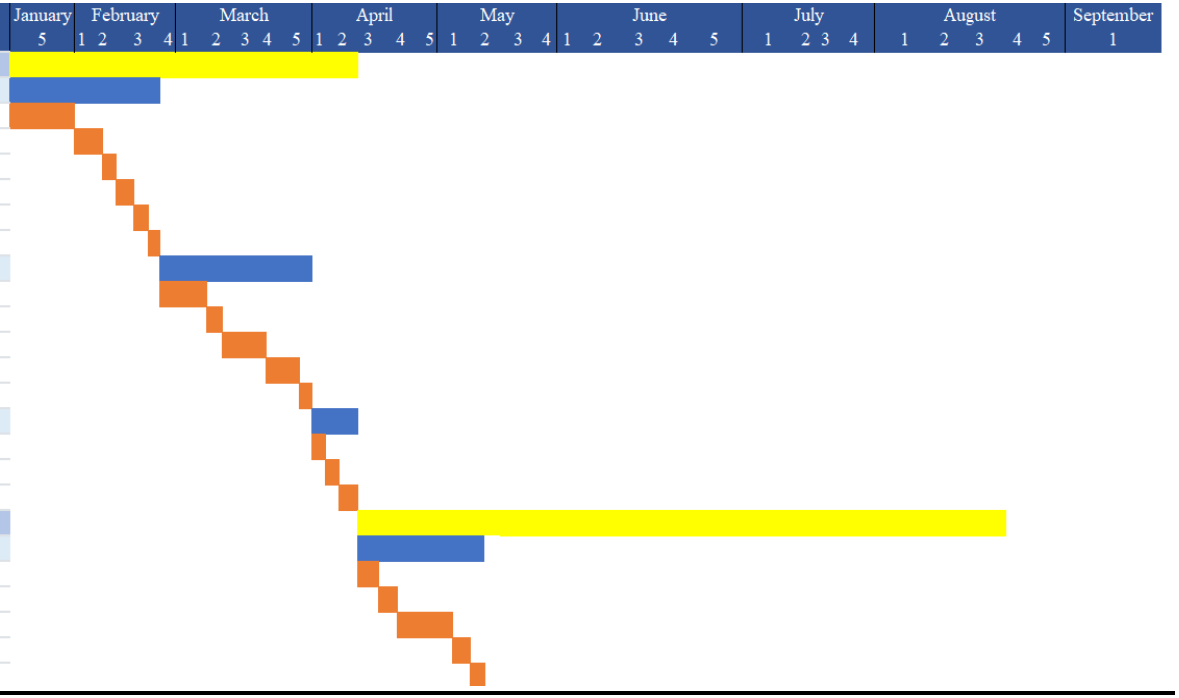

Figure 3.5 Gantt Chart

Second Iteration	24days	13.05.24	05.06.24
New Requirements Gathering and Documentation	5days	13.05.24	17.05.24
Analysis & Design	5days	18.05.24	22.05.24
Implementation	10days	23.05.24	02.06.24
Testing	2days	03.06.24	04.06.24
Evaluation	2days	05.06.24	06.06.24
Third Iteration	24days	07.06.24	30.06.24
New Requirements Gathering and Documentation	2days	07.06.24	08.06.24
Analysis & Design	2days	09.06.24	10.06.24
Implementation	15days	11.06.24	25.06.24
Testing	3days	26.06.24	28.06.24
Evaluation	2days	29.06.24	30.06.24
Fourth Iteration	24days	01.07.24	24.07.24
New Requirements Gathering and Documentation	2days	01.07.24	02.07.24
Analysis & Design	1days	03.07.24	03.07.24
Implementation	16days	04.07.24	19.07.24
Testing	3days	20.07.24	22.07.24
Evaluation	2days	23.07.24	24.07.24
Fifth Iteration	24days	25.07.24	17.08.24
New Requirements Gathering and Documentation	1days	25.07.24	25.07.24
Analysis & Design	1days	26.07.24	26.07.24
Implementation	17days	27.07.24	12.08.24
Testing	3days	13.08.24	15.08.24
Evaluation	2days	16.08.24	17.08.24
Deployment Phase	10days	17.08.24	26.08.24
System Deployment	10days	17.08.24	26.08.24
Report Finalize	10days	27.08.24	05.09.24
Complete Report Writing	10days	27.08.24	05.09.24

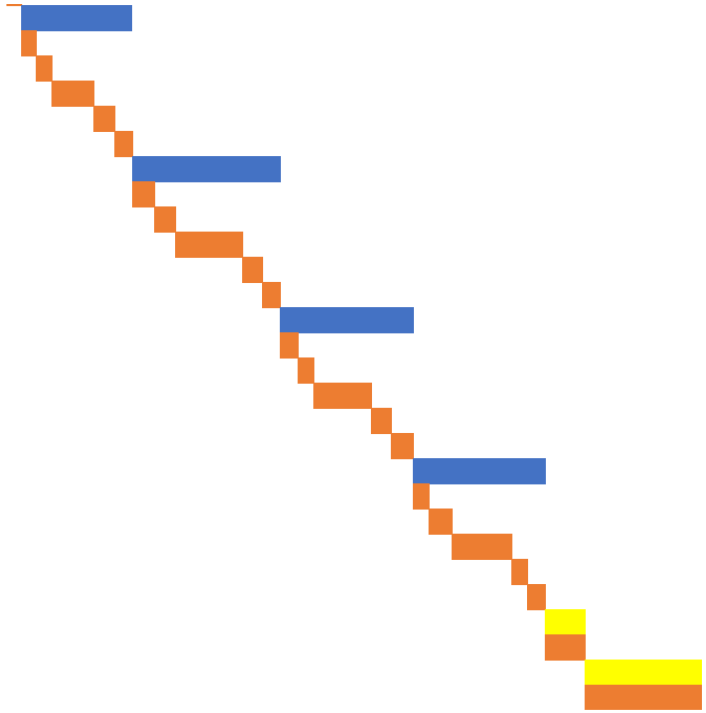

Figure 3.6 Gantt Chart (Continued)

3.4 Development Tools

Development tools can be divided into programming languages, frameworks, integrated development environments (IDE), version control systems, databases, and third-party libraries. This project uses HTML which is markup language, CSS which is styling language, JavaScript and Python which are programming languages, PyTorch as machine learning framework, Flask as backend framework and MongoDB as database. This project will be developed using Visual Studio Code. Git will be used for version control. Finally, the third-party libraries, Numpy and PIL will also be utilized.

3.4.1 HyperText Markup Language (HTML)

HTML is the usual code language used in creating web pages and development of applications. HTML usually describes the content and structure of a webpage by applying tags and attributes. Tags are used to open and close HTML elements while attributes provide more information about elements. Commonly used elements include but not limited to formatting tags, links, images, lists, tables, forms and more. In semantic html elements are used according to their intended purpose for better accessibility and search engine optimization purposes. Online validators can perform this validation task on the html code in order to know if the codes written are valid or not.

3.4.2 Cascading Style Sheets (CSS)

CSS is a language for style sheets that are used to define the layout and presentation of HTML documents. CSS makes it possible for developers to indicate how the various elements in a webpage should appear including color, font, size, space, and position among others. CSS functions through using selectors that put-on styles on html elements thus referring to specific elements or even groups of them. With its help designing a web page becomes easier because it gives flexibility and control over layouts making them look more attractive so that any user feels comfortable while navigating the website.

3.4.3 JavaScript (JS)

JavaScript is a client-side scripting language used for making web pages interactive. Initially designed for enhancing webpage functionality, it's now also utilized for back-end development due to advantages like speed and efficiency. JavaScript's relative simplicity has contributed to its popularity, fostering a large and active community of developers.

3.4.4 Python

Regarded for simplicity and readability, Python is a high-level programming language. It is a general-purpose programming language initially developed, but Its being versatile, easy to use and extensive libraries have made it become popular. In scientific computing, web development, AI or Data analysis among others python is frequently used. Beginners and pro developers love python because of its clear syntax as well as a wide user community that it entails.

3.4.5 PyTorch

PyTorch is a deep learning framework that has made a name for itself through its simplicity and adaptability. This project offers various resources for building and training neural networks, especially those in the field of deep learning. Complex models can be prototyped and tried out with PyTorch’s dynamic computation graph and Pythonic syntax. For instance, it is used extensively by researchers and practitioners in image recognition, natural language processing and more.

3.4.6 MongoDB

MongoDB is a leading NoSQL database acknowledged for its scalability, adaptability and efficiency with unstructured data. It follows a document-based model which provides agile development and fast operations for real-time applications. High availability, JSON-like documents and scalability are some of the reasons why it is preferred by web developers.

3.4.7 Visual Studio Code

Visual Studio Code (VS Code) is a lightweight code editor designed by Microsoft to be used in a variety of applications. Along with syntax highlighting, code recommendations, debugging tools, and version control system integration, it supports a variety of programming languages and frameworks. With the help of numerous plugins that expand its functionality, Visual Studio Code has developed into a sophisticated environment that can efficiently handle a wide range of software development activities. The capabilities that allow for customization and convenient screen navigation when writing or maintaining codes across menus are well-liked by users.

3.4.8 Git

Git is a strong, flexible, and decentralized revision control system that gives developers the ability to store all of their code modifications so they can always go back to earlier iterations in case something goes wrong. Using this technology, programmers may work together more effectively and more readily to track changes made to scripts. Additionally, these codes can be safely backed up on platforms like GitHub to prevent inadvertent loss.

3.4.9 PIL

PIL, also called Pillow, is a library that provides support for numerous file formats as well as having high-performance internal object representation and strong image processing capabilities. It has the ability to perform various picture changing activities like resizing, cropping, rotating and converting images from one format to another. Among its merits are simplicity and ease of use making it be something that an average person who can do some coding easily access. This makes it ideal in web applications, graphical user interfaces and automated image processing scripts.

3.4.10 Numpy

NumPy (Numerical Python) is a package that gives support to large number of multi-dimensional arrays and matrices as well as other mathematical functions operating on these arrays. NumPy, in conjunction with PIL, supports advanced image processing tasks like

normalization, pixel wise operations and feature extraction. It is therefore very fast when dealing with huge sets of images like photographs. What makes it mainly used in scientific computing, data analysis and machine learning applications are array operations by NumPy which are highly optimized.

3.4.11 Flask

Flask is a weightless and various web framework for Python. It eases the process of developing websites through provision of the required resources such as tools and libraries needed in building web applications as well as APIs. Simplicity, flexibility and ease of use are some of the key features that make it one of the most commonly chosen platforms when creating smaller or medium-size websites. This makes it ideal for different kinds of web design activities including routing, templates, handling requests, and maintaining sessions among others.

3.5 Workflow of Development

An intelligent image search engine that uses AI-based similarity detection for a web application has several complicated steps. The first step involves examining the needs of users, holding meetings with stakeholders to spell out goals, user expectations and functional specifications involved. By doing this together as a team, the project is given its limit in terms of what types of images to look for, how similarity will be detected and what users are going to see on the interface among other factors.

After an exhaustive analysis phase of requirement, the design journey unfolds, in which intricate wireframes and mockups are created for the front-end user interface using elaborate tools such as Figma. These visual depictions serve as plans for developing user interfaces that are repeatedly refined through continuous feedback loops with different stakeholders. Accordingly, the production of frontend UI is done with huge attention to details by combining HTML for structural elements, CSS for aesthetic appeal and JavaScript for interactivity. In addition to this development effort, some of the important UI components that must be implemented include a live search box, an intuitive picture display area on page, interactive

refining filters and unified pagination features which facilitate navigation and boost users' experience.

Similarly, the foundation of the backend infrastructure is simultaneously established by creating a strong Flask framework in a dedicated virtual environment. Flask application is designed very well with directories that are set aside to hold templates that consist of HTML documents, static files which include CSS stylesheets and JavaScript scripts, and Python modules which contain the business logic and routing functionalities of the application. One of the most important aspects of Flask API development is creating a set of RESTful API endpoints and specifying how each one may be used to accomplish various tasks, like uploading images and conducting search engine queries. The user preferences, search indices, and image information are using a well-designed document-oriented schema, and these are just some of the diverse data elements managed by an integrated MongoDB database that functions as a powerful storage solution in this backend architecture.

Integrating state-of-the-art AI components is an essential step in the development process. This entails configuring PyTorch and third-party libraries, Numpy and PIL into the Flask environment to give the system enhanced AI capabilities. To realize this, PyTorch which is known for its flexibility and efficiency in deep learning tasks will be employed to implement sophisticated AI algorithms such as Siamese Network used in image similarity detection, One-shot Learning aimed at image recognition and ResNet used in comprehensive feature extraction. In addition to PyTorch, Numpy and PIL is also applied to execute necessary image pre-processing procedures such as resizing, normalization and enhancement that ensure data quality optimality and suitability for ingestion or processing in an AI model.

Testing is comprehensive and ensures the combination of frontend, backend and AI is made perfect. The test makes sure that every part that went into building the platform is functioning as it should. To validate the system, unit testing, integration testing, functional testing and model testing will come next. The test makes sure that every part that went into building the platform is functioning as it should. To validate the system, unit testing, integration

testing, and functional testing will come next. Lastly, model testing is to verify the performance of different deep learning models.

The final step in the development process is to deploy the intelligent picture search engine on a dependable web server configuration so that people may access it with ease. Architectural plans, API specifications, database schemas, and even descriptions of AI models are examples of outcomes. Everything is compiled into a library that may be used for upkeep, bug fixes, and future upgrades. Furthermore, comprehensive training materials and support manuals provide users with the knowledge and tools they need to navigate and operate this intelligent picture finder with efficiency.

The development of an intelligent search engine for images with AI similarity detection mechanisms for web applications involves many nuances and elaborations that essentially reflect a smooth synchrony between meticulous planning, iterative designing, and development cycles, as well as thorough verification and testing, all of which are combined with flawless documentation, deployment on reliable infrastructures, and end user training. By using a comprehensive approach, a user-centric, technologically advanced, and intelligent image search engine is produced. The approach yields value proposition that is unmatched to the stakeholders by increasing the efficiency of use or innovation levels.

3.6 Conclusion

This project will be developed according to the SDLC methodology chosen in this section and the development tools mentioned. The timeline will be strictly adhered to achieve the project objectives. The workflow also fully discussed how the project will be carry out by the tools.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter includes the detailed requirement specification, which defines the functional and non-functional requirements of the application, and the use case analysis, which outlines the interactions and processes of the web application from a user's perspective. This information serves as the foundation for the design and development of the web application.

4.2 Requirements Specification

Requirement specification defines what the application should do and how it should behave. These requirements were obtained through the review of similar applications and based on the recommendations of the project supervisor.

4.2.1 Functional Requirements

Module 1: Image Upload and Search

- The web application shall allow users to upload images.
- The web application shall allow users to search similar image.
- The web application shall allow users to upload images through drag images.

Module 2: AI-Based Image Processing

- The web application shall AI processes uploaded images for feature extraction.
- The web application shall implement a deep learning model for feature extraction from uploaded images.
- The web application shall implement an AI that learn image content features for similarity detection.

Module 3: Search Result Display

- The web application shall utilize AI techniques to compare extracted features and determine image similarity.
- The web application shall display search results in descending order of similarity.
- The web application shall allow users to download the image results.

4.2.2 Non-Functional Requirements

- The web application shall provide efficient image processing and search response times.
- The web application shall provide accurate AI-based similarity detection for search results.
- The web application must be accessible at all times, 24 hours a day.
- The web application shall ensure system reliability with minimal downtime and error handling mechanisms.

4.3 Use Case Diagram

A use case diagram was utilized to analyze and illustrate the users' interactions with the system, providing a better understanding of the requirements. Each user's interaction was depicted as a use case in the diagram. This system consists of one actor and use cases. Figure 4.1 shows the use case diagram of this intelligent image search engine with AI-based similarity detection for web application.

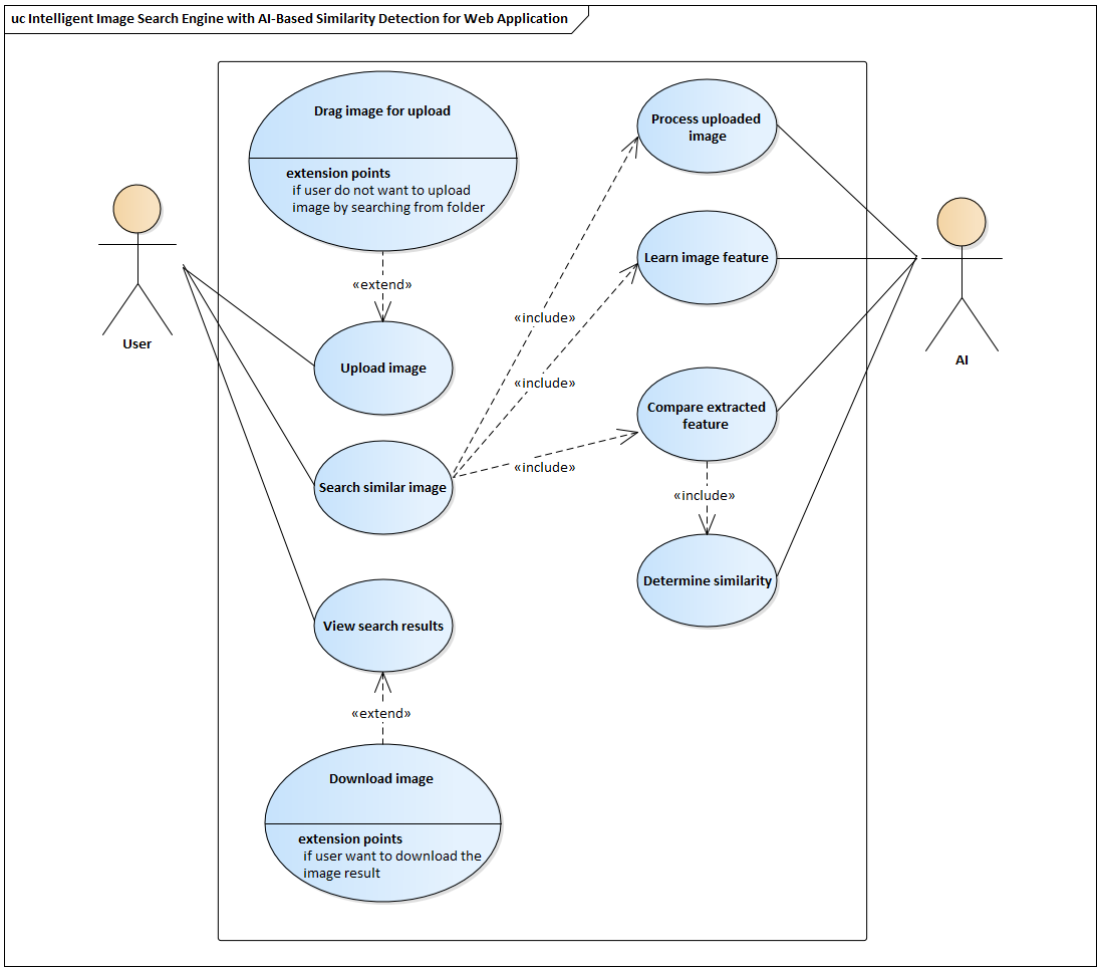

Figure 4.1 Use Case Diagram of Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application

4.4 Use Case Description

4.4.1 Upload Image

Use Case Name: Upload Image	**ID:** UC01	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to upload image		
Brief Description: This use case describes that user can uploads image to the web application.		
Trigger: User wants to upload image to the web application		
Relationships: Association :User Include :N/A Extend : Drag image for upload Generalization : N/A		
Normal Flow of Events: 1. User access to the web application. 2. User selects the image that want to upload from their device. 3. User can upload image by drag the image into the query box. The S-1: Drag image for upload sub-flow performed. 4. The system checks the selected file is in supported image format. If it is not supported image format, perform exceptional flow 3.1. 5. Upon valid image format, the system uploads the image to the web application's server. 6. The system redirects the user to the web application's result display page.		
Sub-flows: S-1: Drag image for upload 1. User selects image from their device. 2. User drags the image into query box on the web application 3. Continue to main flow step 4.		
Alternate/Exceptional Flows: 3.1 Not supported image format uploaded 1. The system prompts the user that the uploaded image is in not supported format.		

2. The system redirect the user to image upload page to reupload the image.

4.4.2 Search similar image

Use Case Name: Search similar image	**ID:** UC02	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to search similar image		
Brief Description: This use case describes that user can search similar image on the web application.		
Trigger: User wants to search similar image on the web application		
Relationships: Association : User Include : Process uploaded image, Learn image feature, Compare extracted feature Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. User access to the web application. 2. User uploads the image query. 3. System processes the uploaded image to extract its features. 4. System learns the features of the uploaded image. 5. System compares the extracted features with other images in the database. If there is no similar image in the database, perform exceptional flow 5.1. 6. System displays a list of images like the uploaded image. 7. User views the image results.		

Sub-flows: N/A
Alternate/Exceptional Flows: 5.1 No similar image in the database 1. The system notifies the user that no similar images were found. 2. The system redirect the user to image upload page to upload different image.

4.4.3 View search results

Use Case Name: View search results	**ID:** UC03	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to view search results		
Brief Description: This use case describes that user can view search results on the web application.		
Trigger: User wants to search view search results on the web application		
Relationships: Association : User Include : N/A Extend : Download image Generalization : N/A		

Normal Flow of Events: 1. User searches image on the web application. 2. System processes the search query and retrieves relevant results. 3. System displays a list of search results. 4. User views the search results. 5. User clicks on search results to view it in a bigger size. 6. User can download the image. The S-1: Download image flow performed.
Sub-flows: S-1: Download image 1. User clicks on the search results. 2. User presses on the download button. 3. User notifies that the image is successful downloaded. 4. Return to main flow step 4.
Alternate/Exceptional Flows: N/A

4.4.4 Process uploaded image

Use Case Name: Process uploaded image	**ID:** UC04	**Importance Level:** High
Primary Actor: AI	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: AI – process the uploaded image		
Brief Description: This use case describes that AI can process the uploaded image.		
Trigger: User wants to search view search results on the web application		
Relationships: Association : AI Include : N/A Extend : N/A Generalization : N/A		

Normal Flow of Events: 1. AI receives the uploaded image from the web application. 2. AI analyzes the image to extract relevant features or information. 3. AI processes the image data using algorithms. 4. AI identifies patterns within the images. 5. AI generates output based on the processed image data.
Sub-flows: N/A
Alternate/Exceptional Flows: N/A

4.4.5 Learn image feature

Use Case Name: Learn image feature	**ID:** UC05	**Importance Level:** High
Primary Actor: AI	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: AI – extract and learn image's features		
Brief Description: This use case describes that AI can extract and learn image feature.		
Trigger: User wants to search view search results on the web application		
Relationships: Association : AI Include : N/A Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. AI receives the uploaded image to learn its features.		

2. AI analyzes the image data to identify key features and patterns. 3. AI extracts relevant features from the image data. 4. AI applies algorithms to learn and recognize image features. 5. AI stores the learned features for future use in image recognition.
Sub-flows: N/A
Alternate/Exceptional Flows: N/A

4.4.6 Compare extracted feature

Use Case Name: Compare extracted feature	**ID:** UC06	**Importance Level:** High
Primary Actor: AI	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: AI – compare the image's features		
Brief Description: This use case describes that AI can compare the image's features		
Trigger: User wants to search view search results on the web application		
Relationships: Association : AI Include : Determine similarity Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. AI receives features extracted from an image. 2. AI retrieves reference features or patterns from its knowledge base.		

3. AI compares the extracted features with reference features using similarity metrics or algorithms. 4. AI generates a comparison result indicating the degree of similarity or difference between the features. 5. AI provides the comparison result for further analysis or decision-making.
Sub-flows: N/A
Alternate/Exceptional Flows: N/A

4.4.7 Determine similarity

Use Case Name: Determine similarity	**ID:** UC07	**Importance Level:** High
Primary Actor: AI	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: AI – determine similarity of uploaded image and image in database		
Brief Description: This use case describes that AI can determine the similarity of images.		
Trigger: User wants to search view search results on the web application		
Relationships: Association : AI Include : N/A Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. AI receives an uploaded image and retrieves reference images from the database. 2. AI extracts features from the uploaded image and reference images.		

3. AI compares the features of the uploaded image with those of the reference images. 4. AI calculates similarity scores or metrics based on the comparison. 5. AI determines the degree of similarity between the uploaded image and the images in the database. 6. AI provides the similarity determination result, indicating the level of similarity.
Sub-flows: N/A
Alternate/Exceptional Flows: N/A

4.5 Activity Diagram

The activity diagram shows the view of flow within a specific use case in detail. This activity diagram will show that how intelligent image search engine with AI-based similarity detection for web application handles user interactions and performs tasks such as image upload, processing and similarity detection. Below is the figures of activity diagram for each use case description.

4.5.1 Upload image

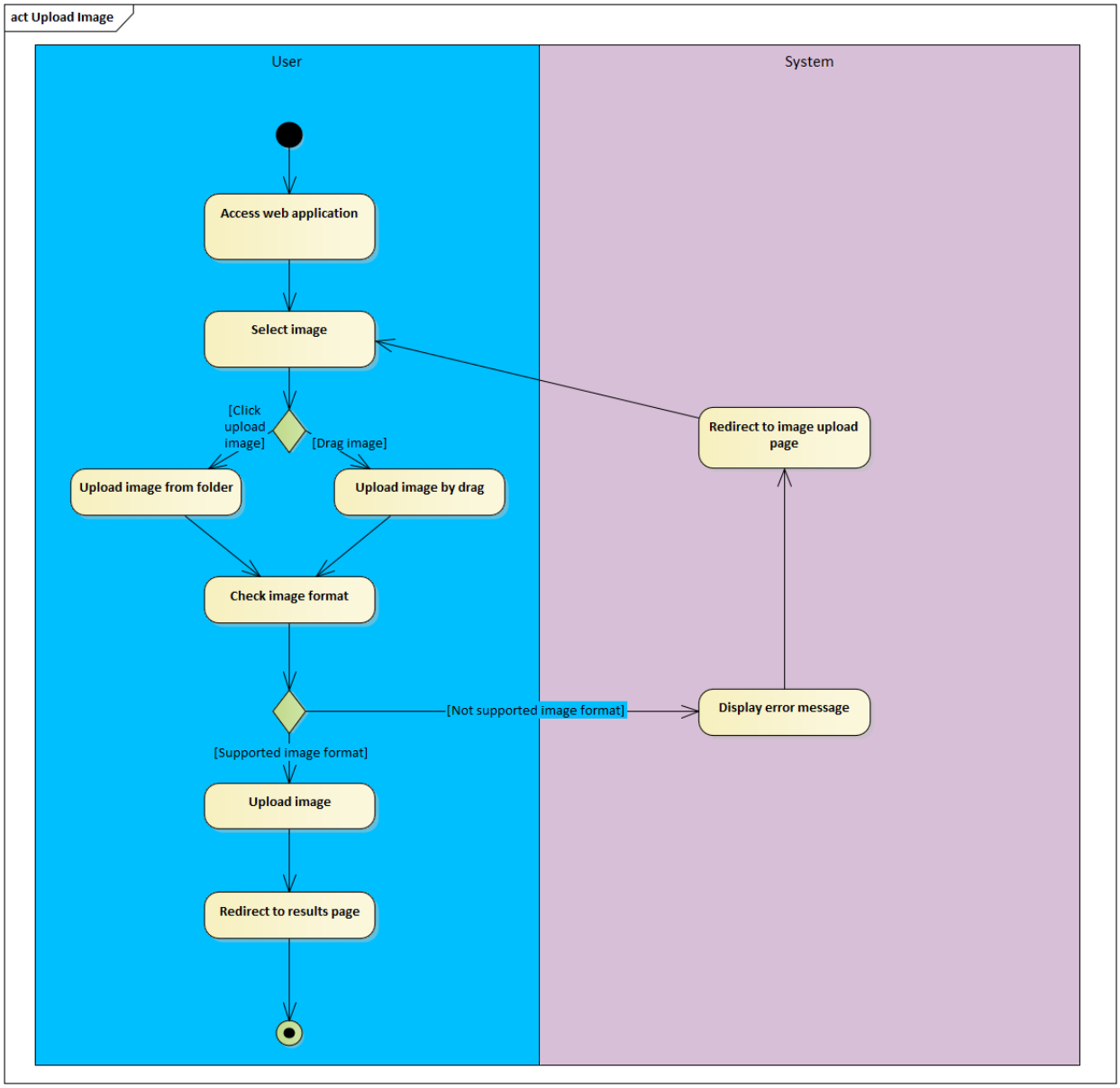

Figure 4.2 Activity diagram for upload image

4.5.2 Search similar image

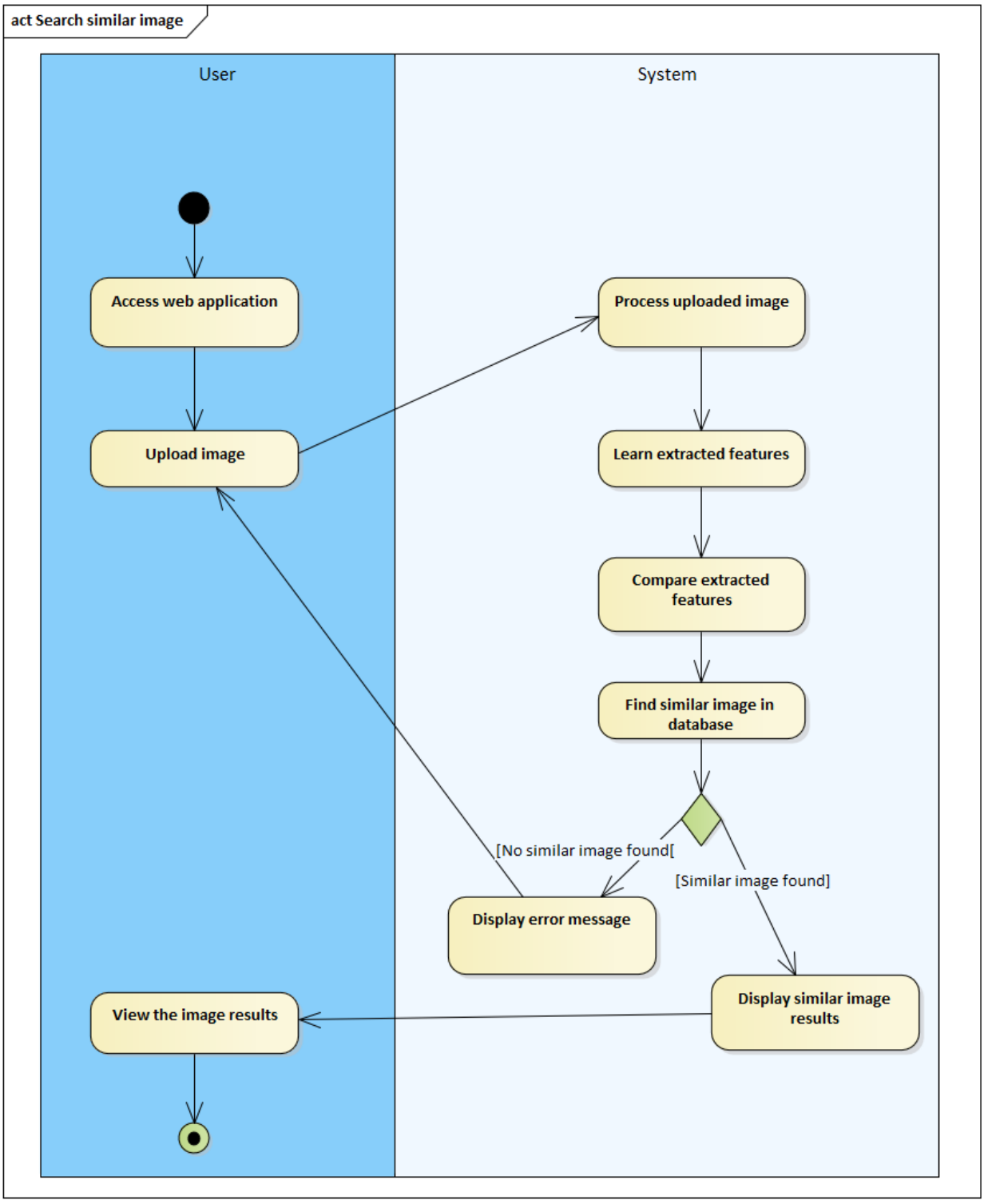

Figure 4.3 Activity diagram for search similar image

4.5.3 View search results

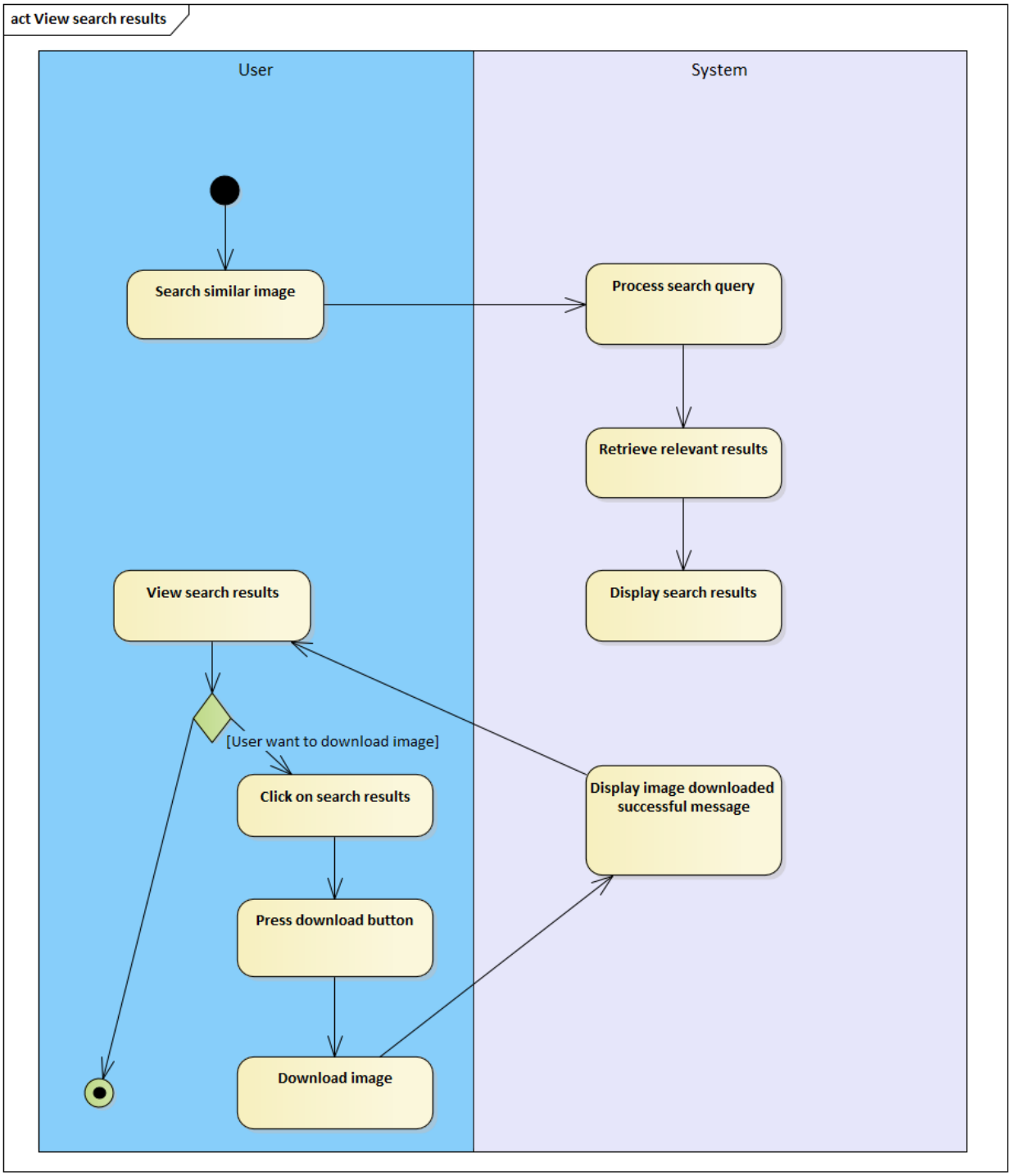

Figure 4.4 Activity diagram for view search results

4.5.4 Process uploaded image

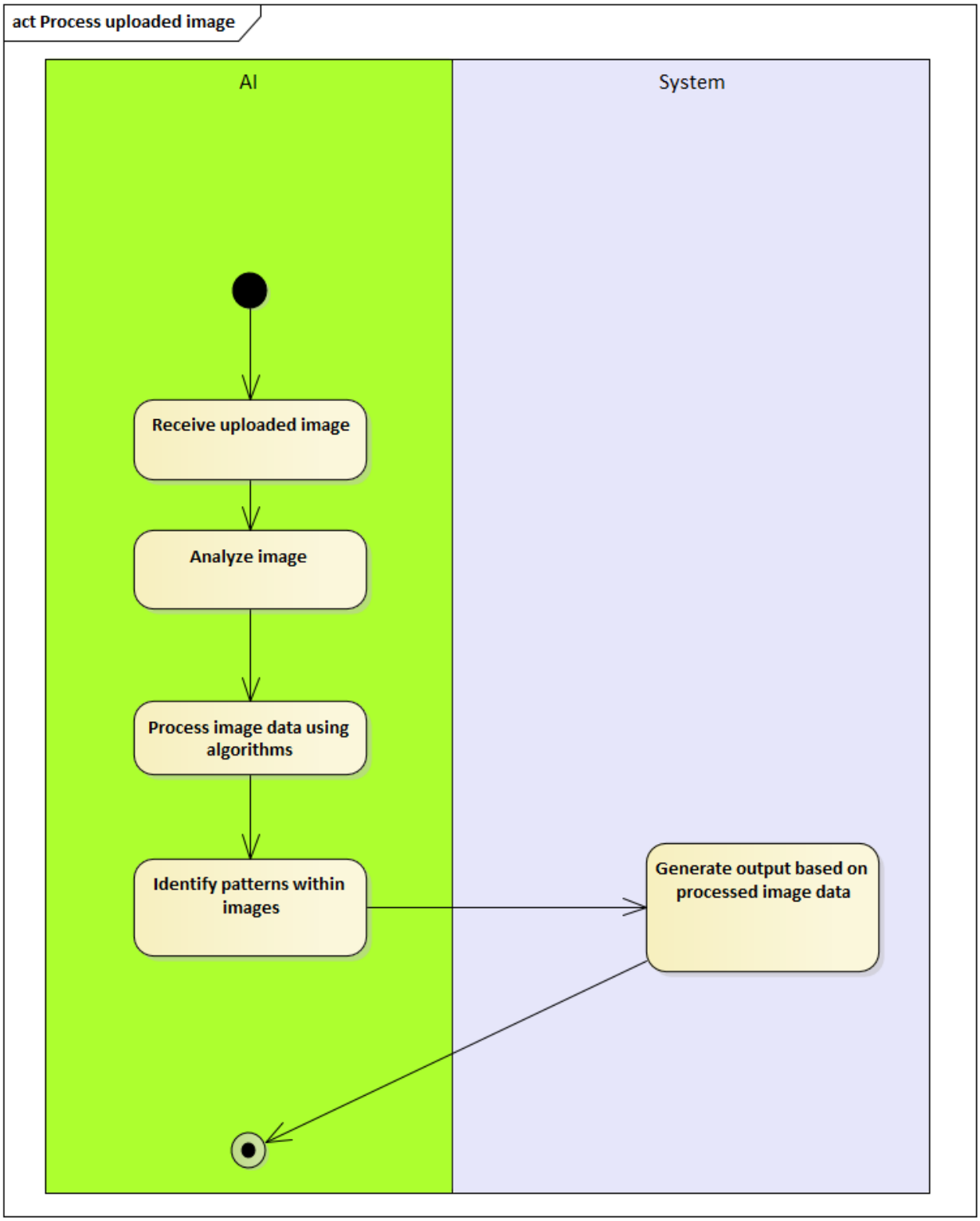

Figure 4.5 Activity diagram for process uploaded image

4.5.5 Learn image feature

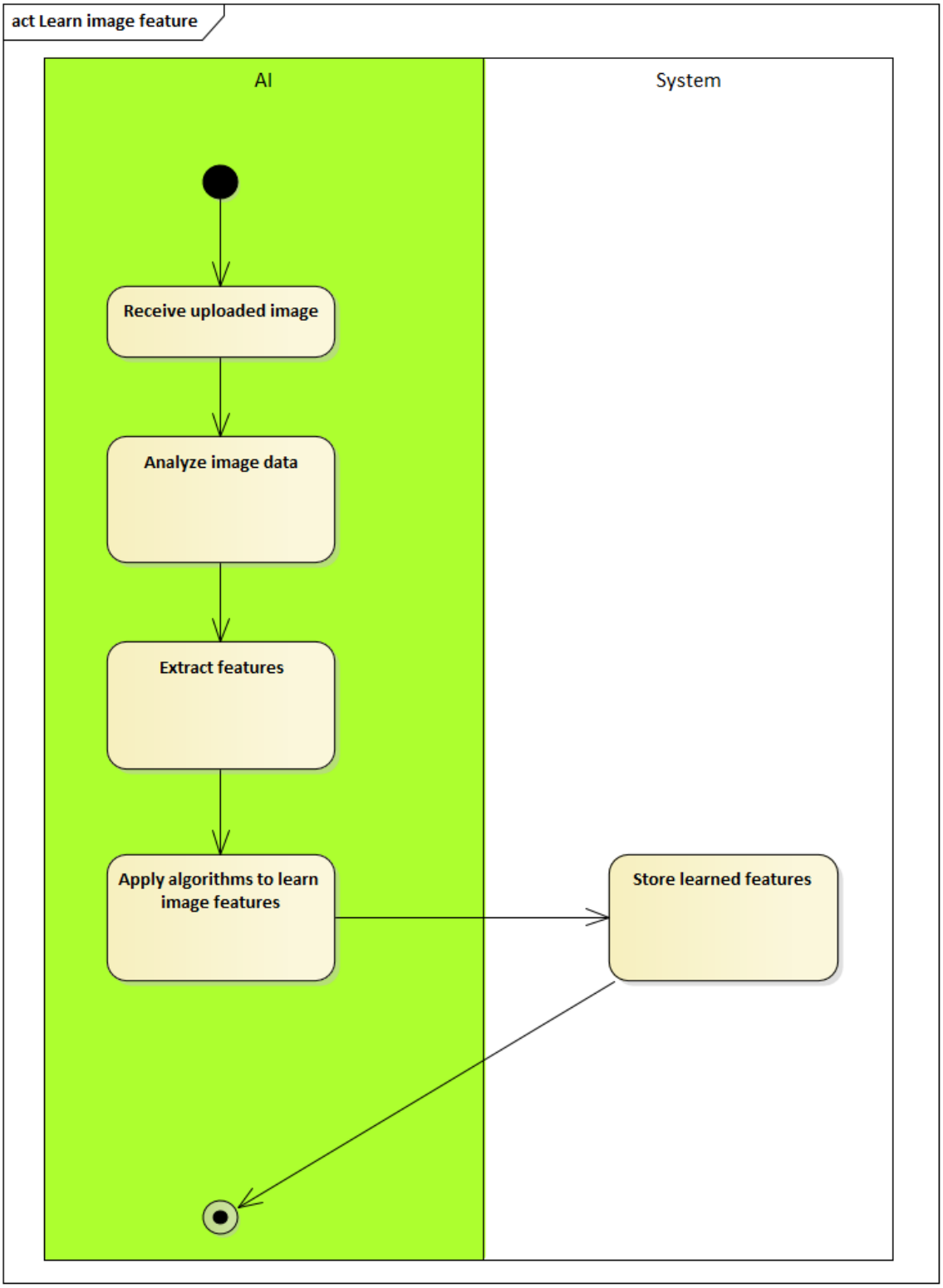

Figure 4.6 Activity diagram for learn image features

4.5.6 Compare extracted feature

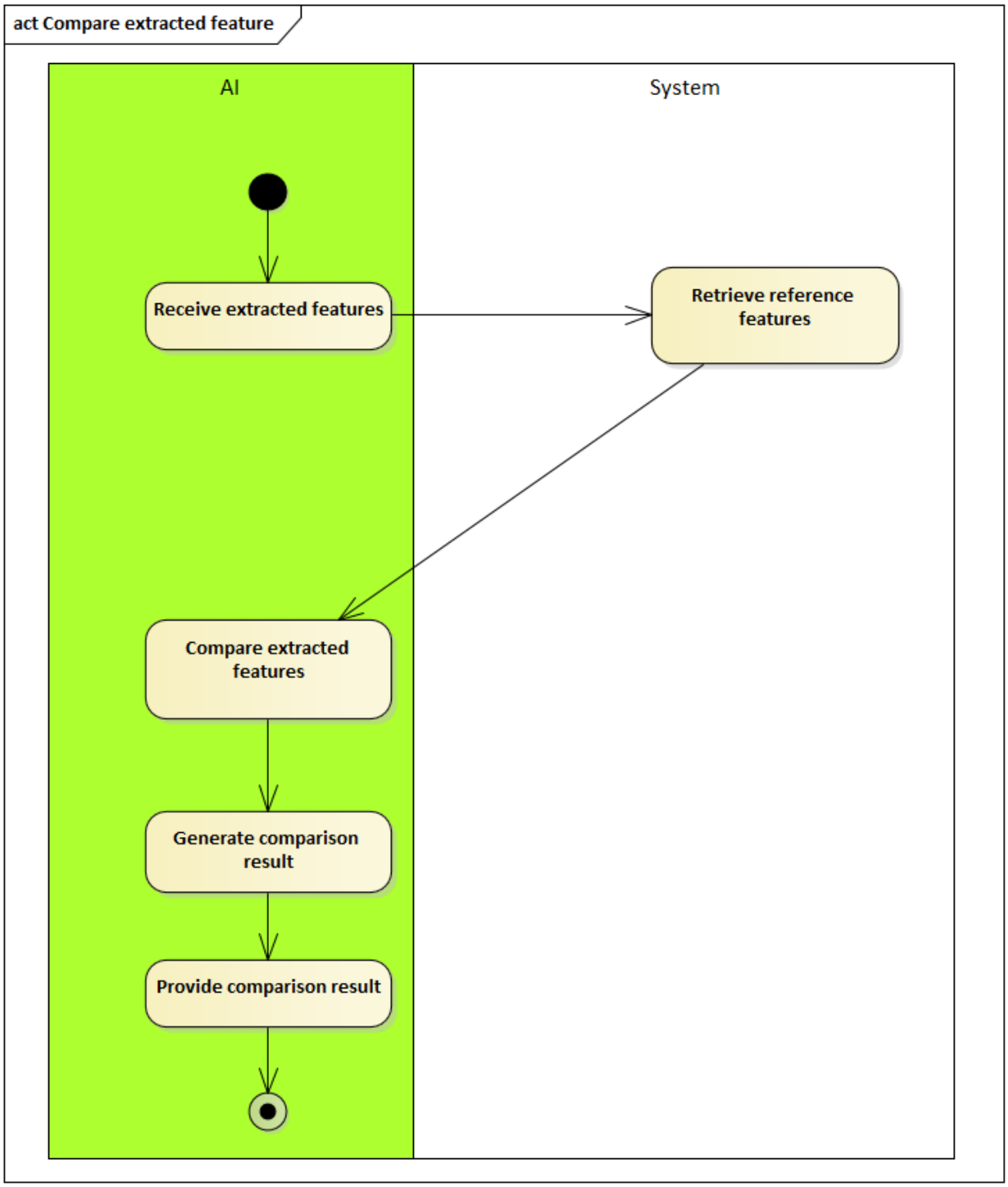

Figure 4.7 Activity diagram for compare extracted feature

4.5.7 Determine similarity

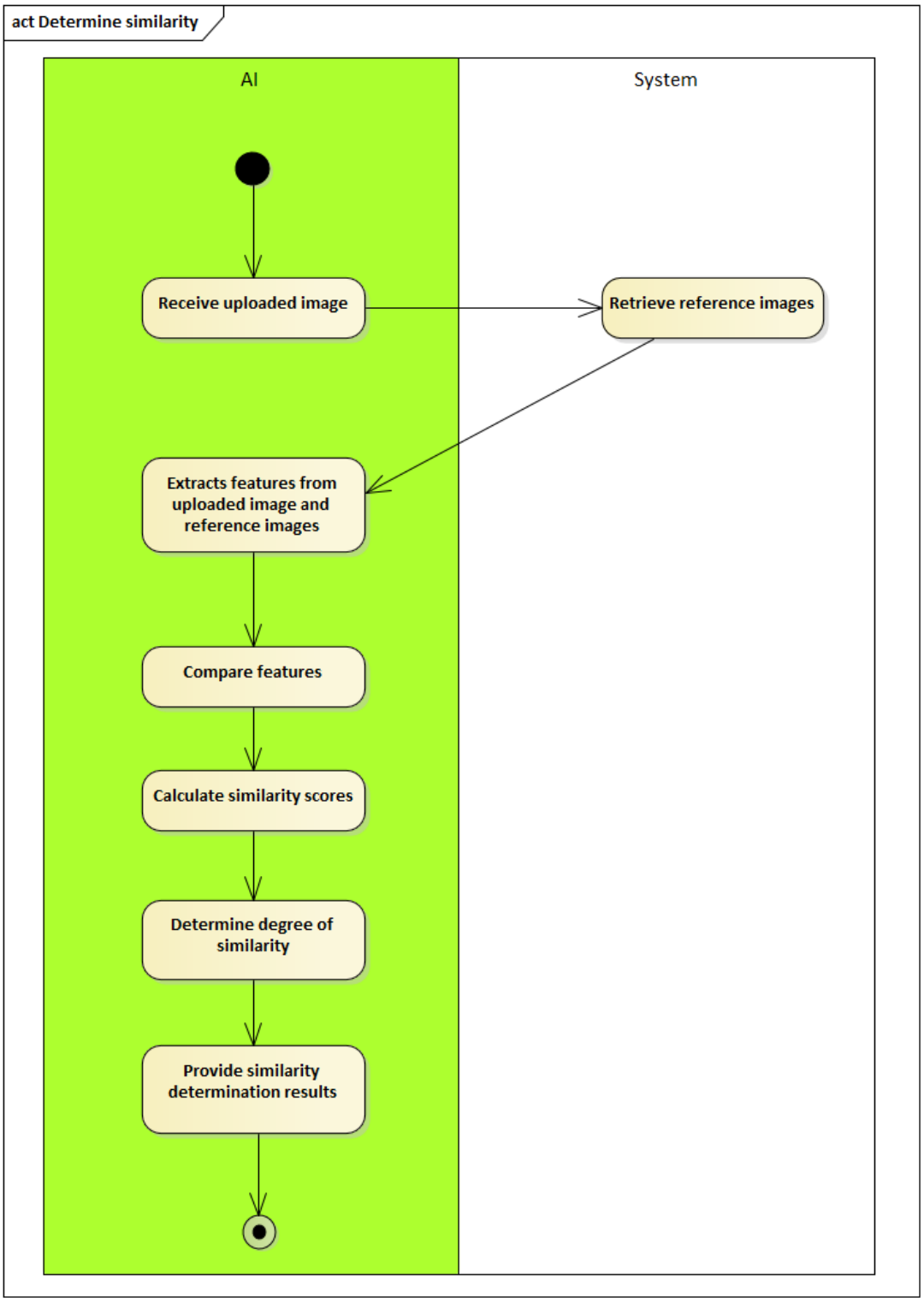

Figure 4.8 Activity diagram for determine similarity

4.6 Data Flow Diagram

The data flow diagram is used to give a structured overview of the data flow within the intelligent image search engine with AI-based similarity detection for web application. It shows the flow of data between different components of the system, including user interactions, image processing, AI engine and database operation.

4.6.1 Context Diagram

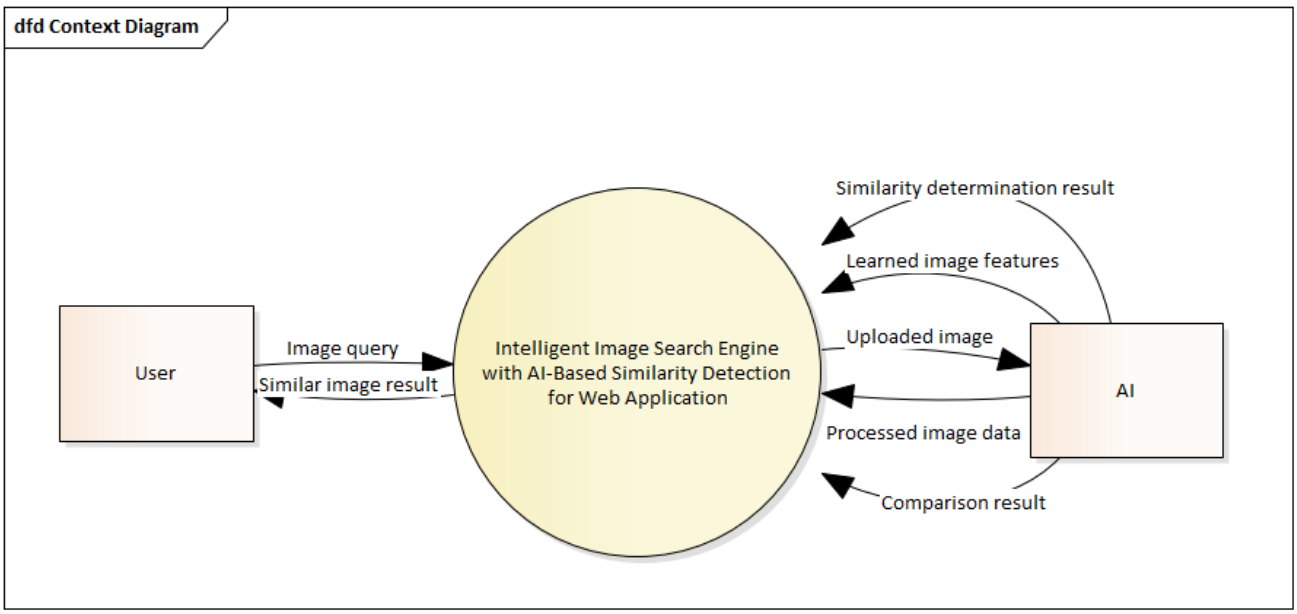

Figure 4.9 Context Diagram

4.6.2 Level 0 Diagram

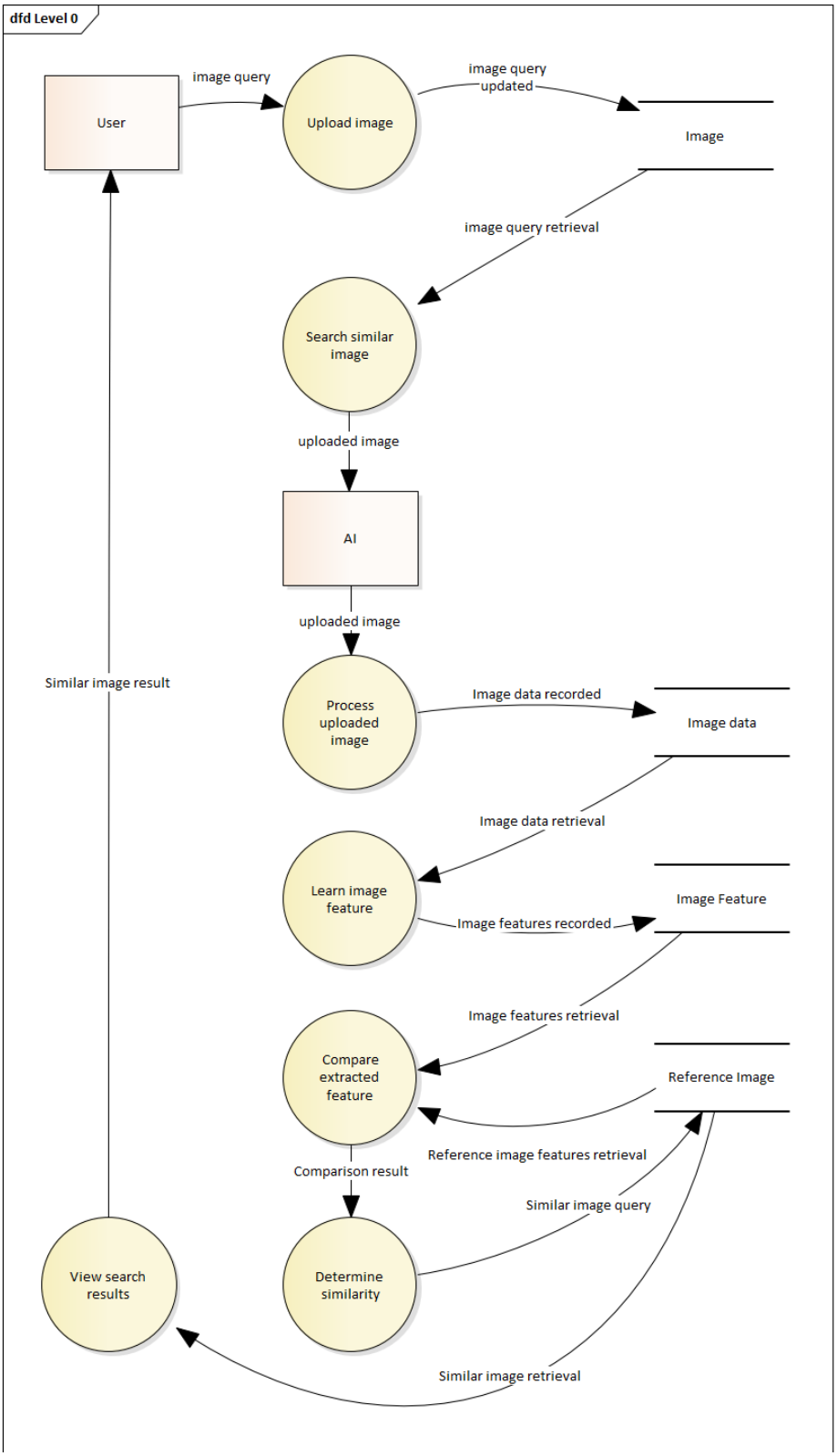

Figure 4.10 Level 0 Diagram

Chapter 5

SYSTEM DESIGN

5.1 Introduction

In this chapter, the system architecture, database, and user interface of the Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application are presented and discussed. The system architecture design focuses on explaining the structure and behavior of the system, which will include the use of frameworks and their relationship. In addition, the system database design is illustrated with the entity relationship diagram. Lastly, the system's user interface was designed to be efficient with high usability.

5.2 System Architecture Design

The intelligent image search engine utilized a client-server architecture that consists of three tiers. Three-tier client-server architecture is a software architecture pattern in which the user interface (presentation), business logic (application server), and database management (data) are separated into three distinct components. Figure 5.1 shows the diagram of the system architecture of the web application. The reason for selecting this architecture is because it allows for greater flexibility and scalability, as changes made in one tier do not affect the other tiers, and each tier can be optimized for its specific role. Additionally, the intelligent image search engine with AI-based similarity detection is using RESTful API which allows other systems to interact with the backend, enhancing modularity and scalability.

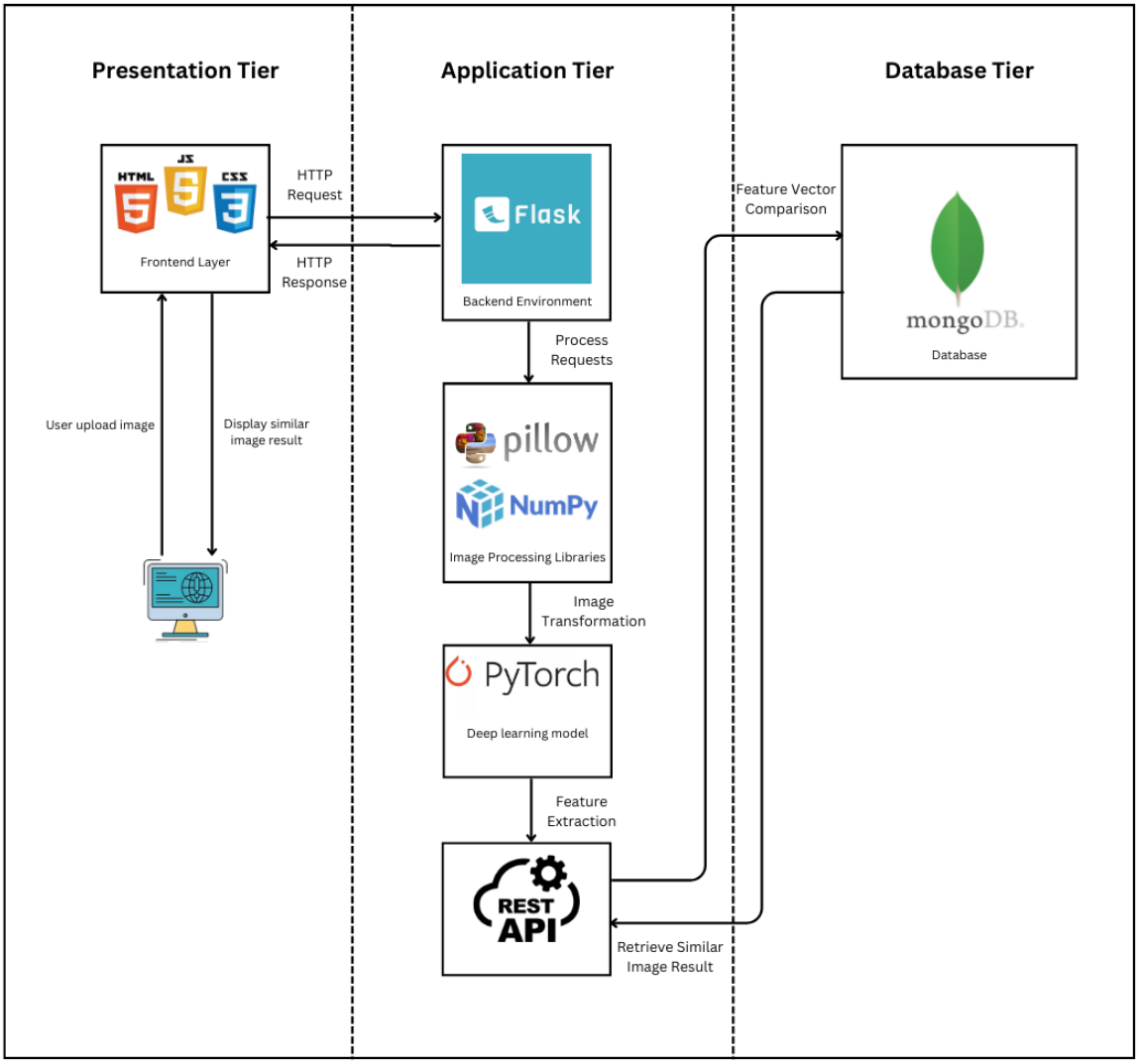

Figure 5.1 System Architecture Design

The presentation tier refers to the frontend layer of the web application, will be created using HTML, CSS, and JavaScript. It is responsible for presenting similar image result to the user and receiving user query image. The intelligent image search engine with AI-based similarity detection for web application includes a user-friendly interface which are upload image page and result page which are done by using HTML and CSS while JavaScript is used to handle user interactions such as image uploads and displays similar image result.

The application tier is the middle layer, responsible for executing the business logic, managing data access and processing, and performing tasks such as image search and similarity detection by using the trained model. Flask is the web framework used to create a backend server environment in the application tier. This backend server environment will continuously operate to process HTTP requests, extracts feature of the user upload images by using a trained Siamese Network model and retrieves similar images result from the database. Flask is chosen to be used due to its flexibility that can process various types of requests and can be scaled independently and the RESTful API that can exposes endpoints that allow other systems to interact with the backend, enhancing modularity.

Furthermore, the trained model used in application tier has been trained to recognize image similarity by extract the feature vector and compare it. The training process are collecting and preprocessing image data to create training and validation sets. From the training, Siamese Network know how to identify features to measure the similarity between images. Once the training result accuracy is achieving expectation, the trained model is saved and integrated into Flask. Flask will load the trained model to use to extract the image feature and compare the feature vector with the images in database and retrieve the highest similarity score images.

Lastly, the database tier is responsible for storing images and their feature vector and also retrieving the similar images after the comparison. It includes a database management system (DBMS) that handles data storage, retrieval, and manipulation. This system used MongoDB for storing image feature vector and GridFS for storing images. The reason to choose MongoDB and GridFS due to MongoDB is a NoSQL database which suitable for storing image feature vector which allows for flexible and scalable data storage and GridFS is a specification for storing and retrieving large files such as images in MongoDB.

5.3 Database Design

Database design process such as defining data elements, data relationship and normalizing data was conducted before the actual database was built. In this section, the Entity Relationship Diagram (ERD) and Data Dictionary will be included to illustrate an overview of the structure of the database used for the intelligent image search engine with AI-based similarity detection for web application.

5.3.1 Entity Relationship Diagram

An Entity Relationship Diagram (ERD) is a valuable tool in database design and development to ensure that the database design meets the requirements. Figure 5.2 shows the ERD of the Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application. This relation database contains 3 tables, namely fs.chunks, fs.files and image_features.

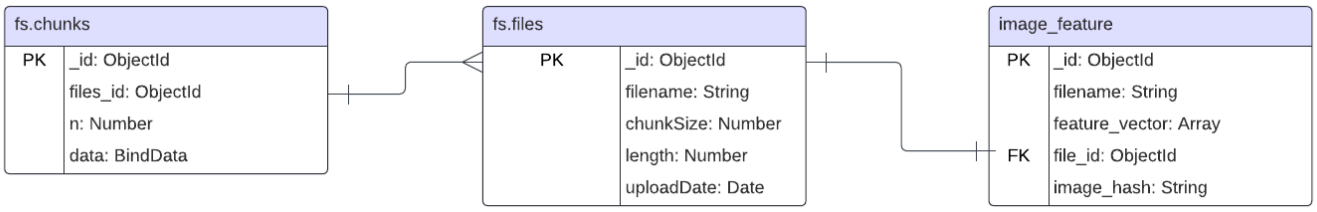

Figure 5.2 Entity Relationship Diagram

5.3.2 Data Dictionary

Table Name	Description
fs.chunks	Stores chunks of files for GridFS.
fs.files	Metadata for files stored in GridFS.
image_features	Contains image information and feature vectors extracted from the images.

Table 5.1 Description of Database Tables

Fields	Field's Description	Type	Example
_id [PK]	Unique identification number for a chunk	ObjectId	667d65204aba340d3044e308
files_id [FK]	Reference to the file in the fs.files collection	ObjectId	667d65204aba340d3044e307
n	Chunk sequence number	Number	0
data	Binary data of the chunk	BinData	Base64 encoded string

Table 5.2 fs.chunks Collection Data Dictionary

Fields	Field's Description	Type	Example
_id [PK]	Unique identification number for a file	ObjectId	667d65204aba340d3044e307
filename	Name of the uploaded file	String	train/baseball/001_1719493920.png
chunkSize	Size of each chunk	Number	261120
length	Size of the file	Number	37760
uploadDate	Date and time of upload	Date	2024-0627T13:12:00.864+00:00

Table 5.3 fs.files Collection Data Dictionary

Fields	Field's Description	Type	Example
_id [PK]	Unique identification number for an image	ObjectId	667d65204aba340d3044e309
filename	Name of the uploaded image	String	train/baseball/001_1719493920.png
feature_vector	Extracted feature vector of the image	Array	[0.12, 0.32, …]
file_id [FK]	Reference to the file in the fs.files collection	ObjectId	667d65204aba340d3044e307
image_hash	Hash of the image content for duplicate detection	String	6ada4a129c8fe7c9eeb3a10931041b44

Table 5.4 image_features Collection Data Dictionary

5.4 User Interface Design

User interface design can present abstract requirements to the user through visual representation, ensuring they meet the user's expectations. Furthermore, when the design is approved, it can speed up the subsequent front-end development and reduce unnecessary changes.

5.4.1 Mock Up

The high-fidelity mock-up was created by using Figma. This mock is divided into two parts which are image upload page and image result page. The components in each page are also designed to fulfil specific requirements.

5.4.1.1 Image upload page

Image upload page is the first page of intelligent image search engine with AI-based similarity detection for web application because it allows users to upload the image to trigger the similar image retrieval process. Figure 5.3 shows the UI design of the image upload page.

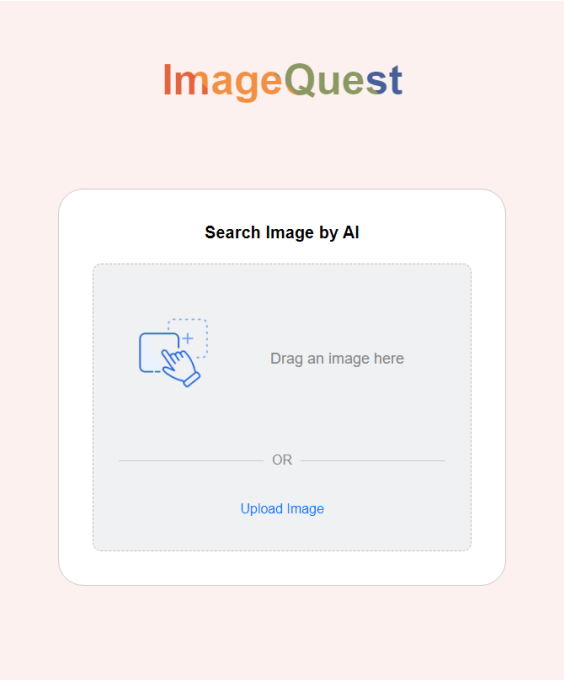

Figure 5.3 Image Upload page

5.4.1.2 Drag image for upload

In image upload page, there is a component which allow users to drag and drop the image and the image will automatically upload and process in the backend. Figure 5.4 shows the drag image process.

Figure 5.4 Drag image for upload

5.4.1.3 Upload image from folder

In image upload page, the component has an upload link which users can use it to choose the image that want to upload from the folder. Figure 5.5 shows the upload image from folder process.

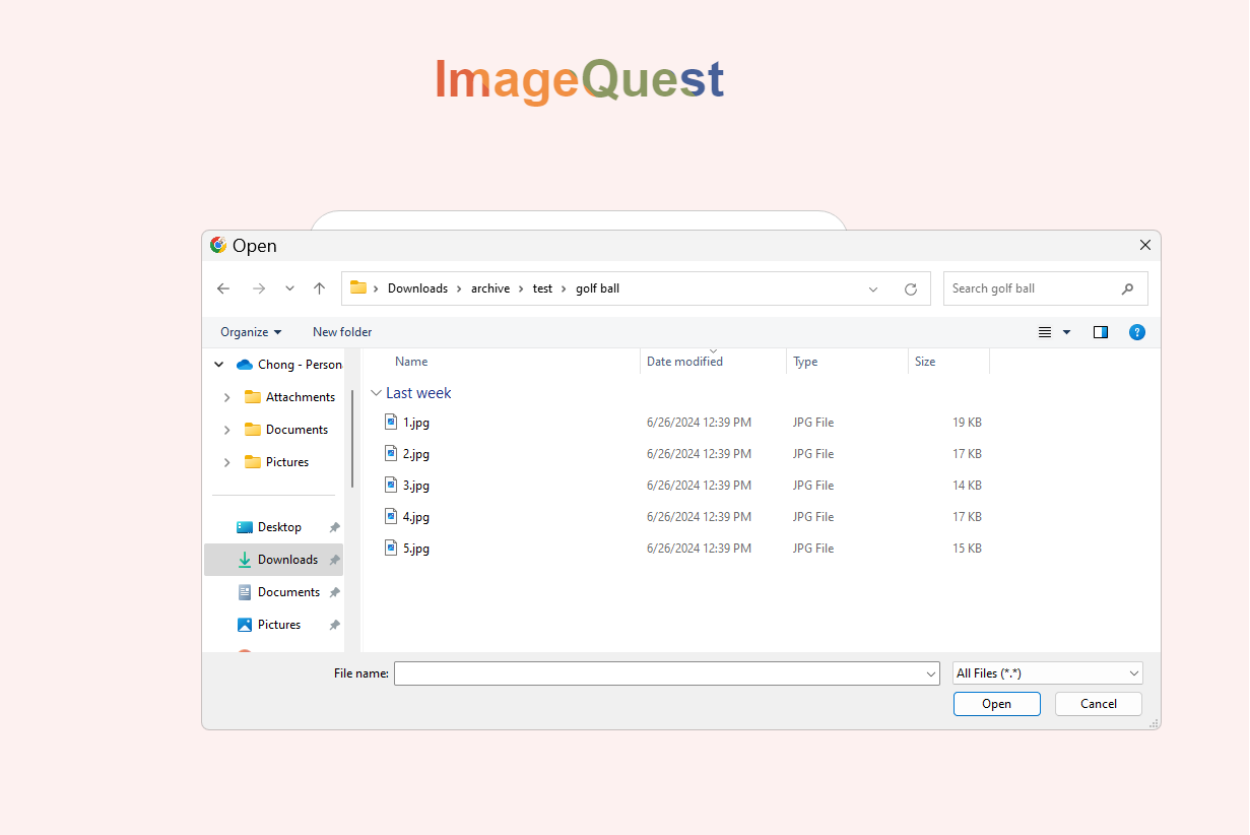

Figure 5.5 Upload image from folder

5.4.1.4 Loading indicator

A loading indicator is shown after image is uploaded by users to show that similar image searching function is running. Figure 5.6 shows the loading indicator process.

Figure 5.6 Loading indicator

5.4.1.5 Image results page

Image result page is the second page of intelligent image search engine with AI-based similarity detection for web application to display the similar image result to users. Figure 5.7 shows the UI design of the image result page.

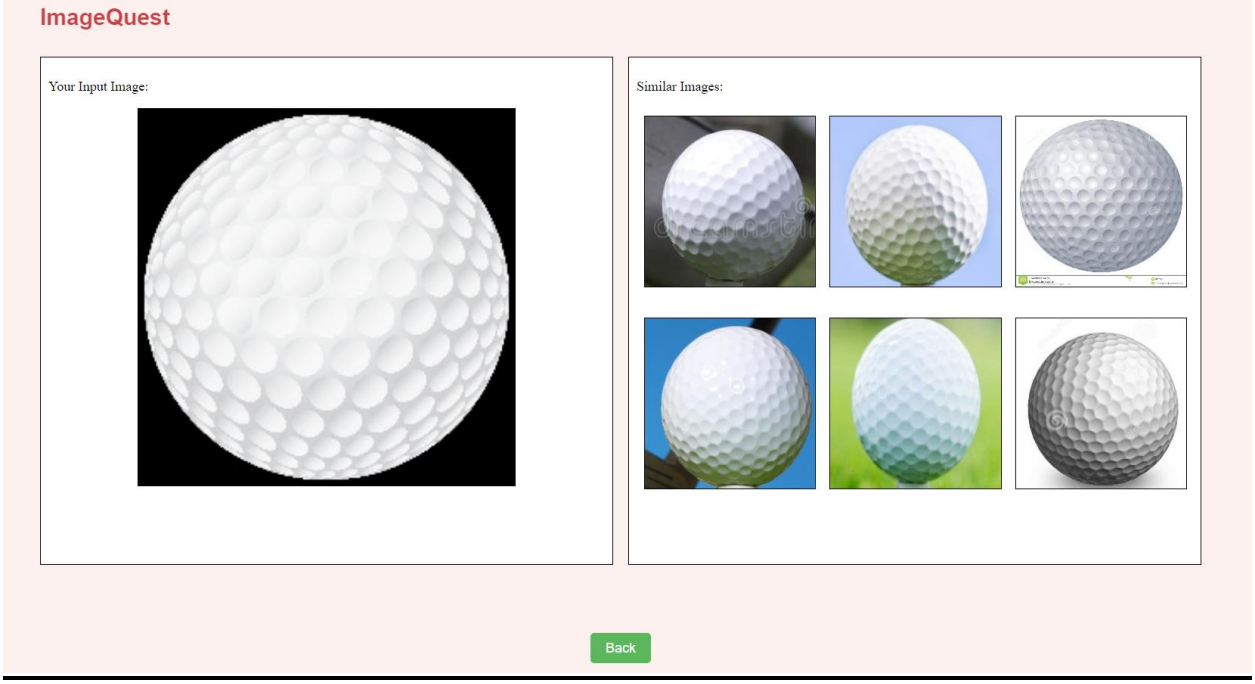

Figure 5.7 Image results page

5.4.1.6 Download image

In image result page, there is a download window when users click on the similar image result and it allows users to download the image. Figure 5.8 shows the pop-out download window.

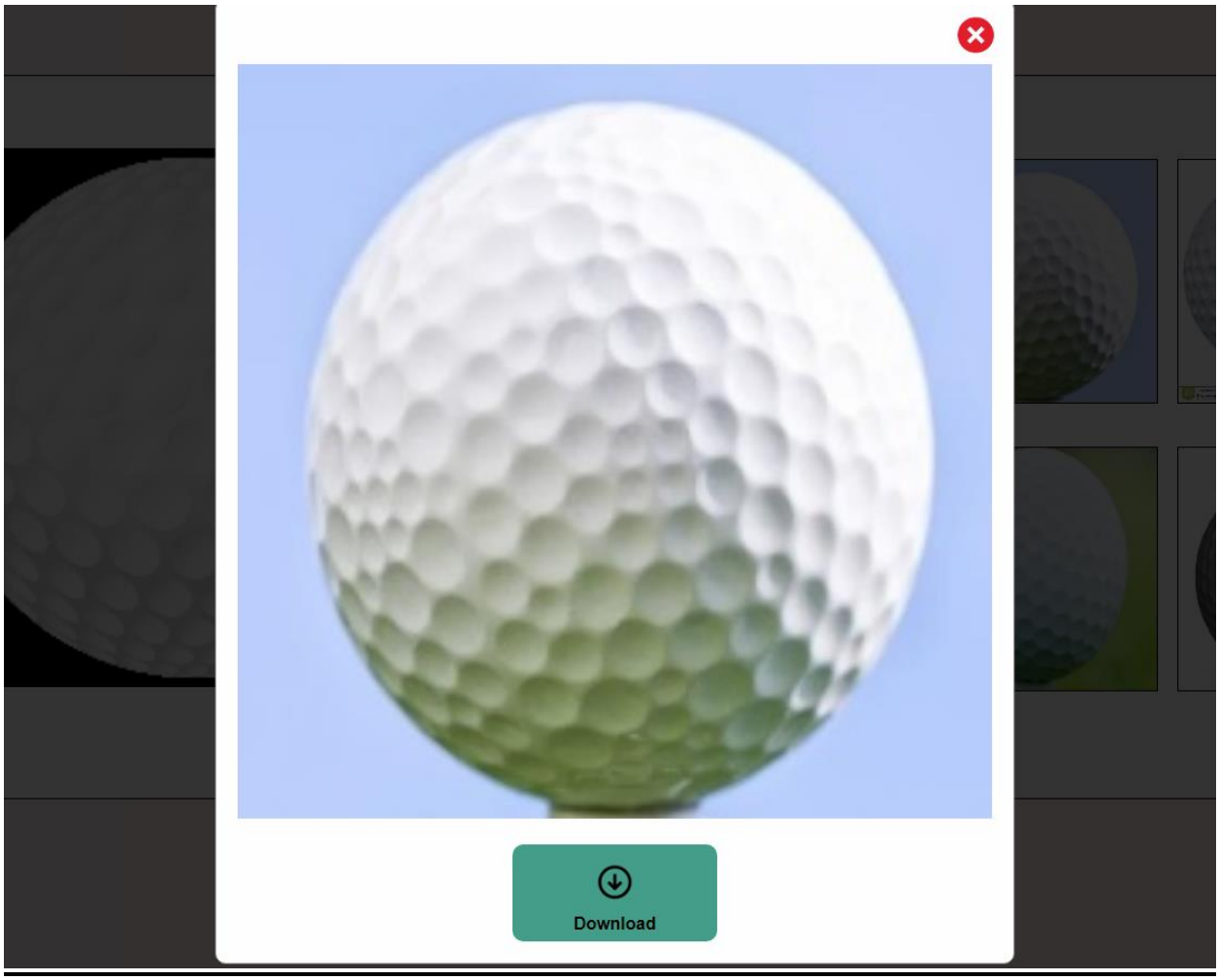

Figure 5.8 Download window

5.4.1.7 Download successful notification

From the download window, users pressed the download button to download the image and when it is successfully downloaded, a notification will come out to notice that the image had been downloaded successful. Figure 5.9 shows the download successful notification.

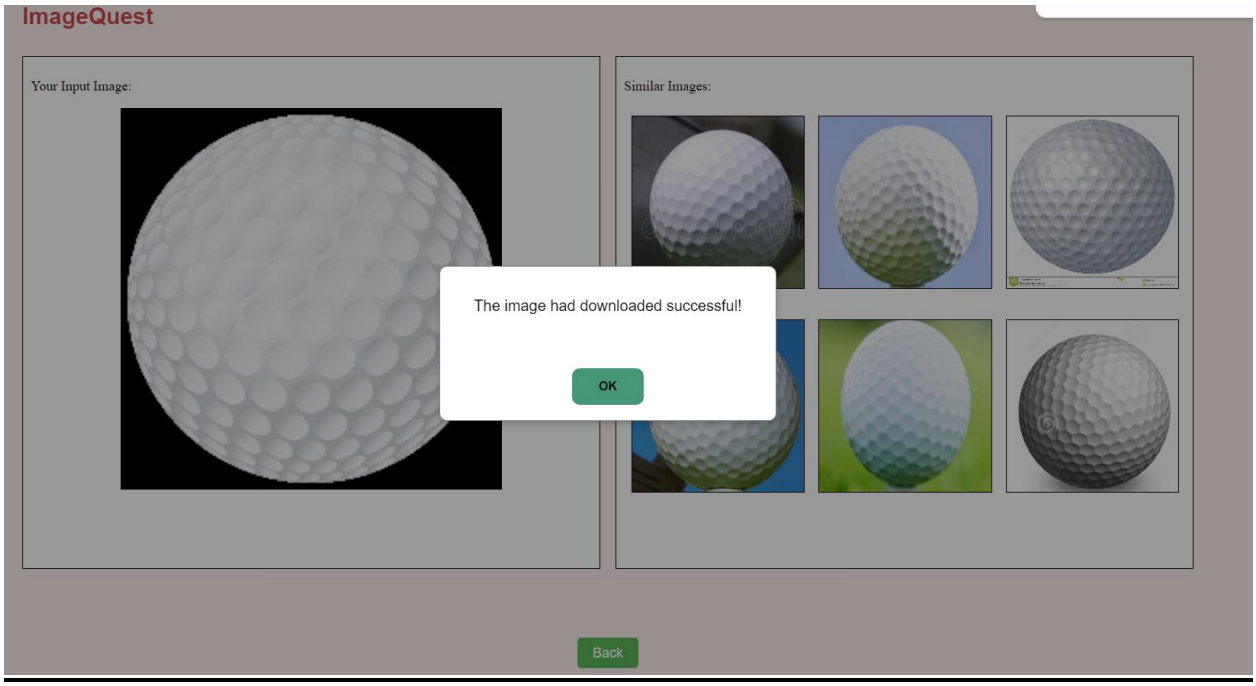

Figure 5.9 Download successful notification

5.4.1.8 No similar image detected notification

While the searching process ended and no similar image detected from the database, a no similar image detected notification is shown to notice the users no similar image detect from the database. Figure 5.10 shows the no similar image detected notification.

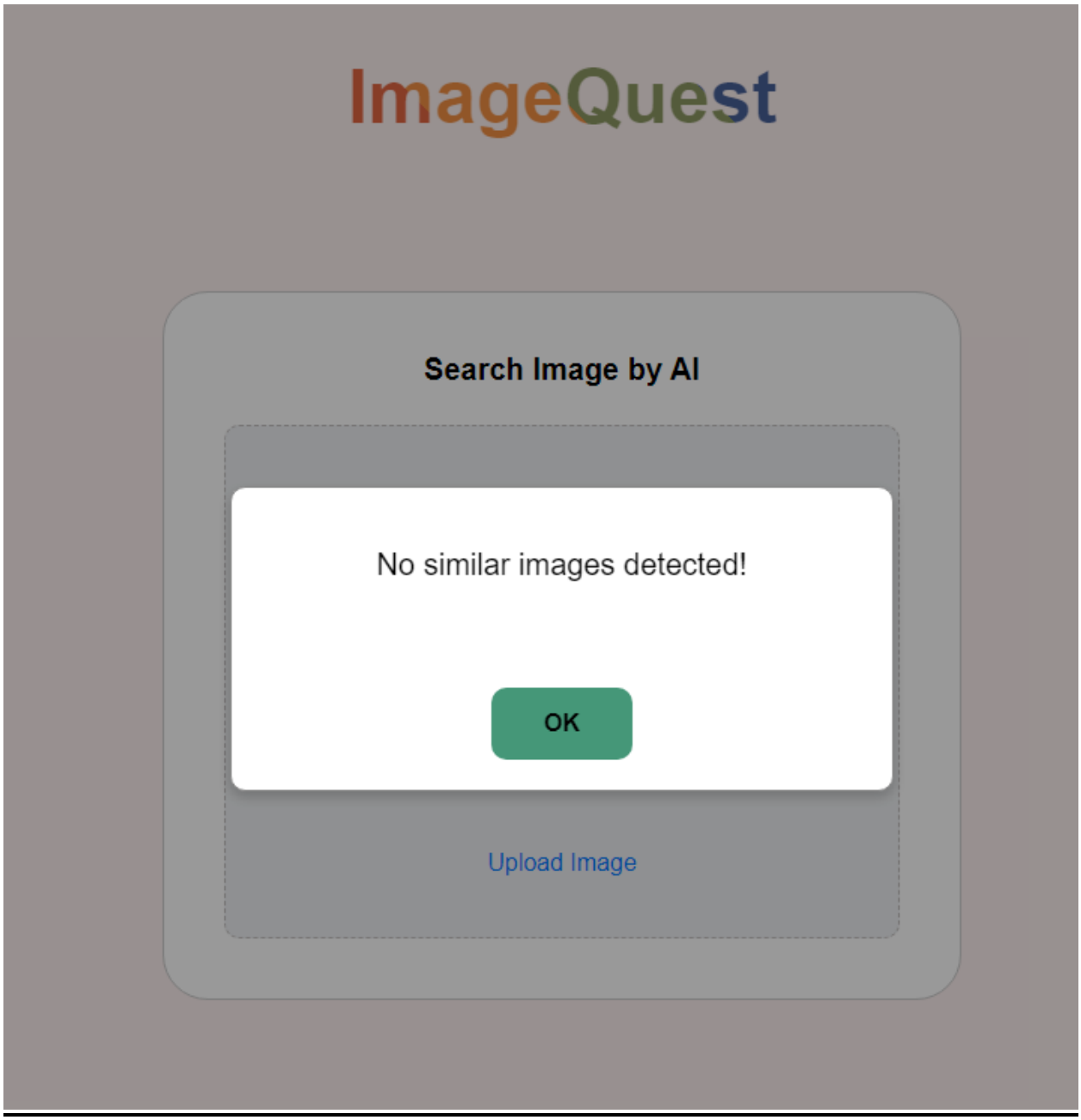

Figure 5.10 No similar image detected notification

5.4.1.9 Not supported format notification

From image upload page, if users upload any other format of files which is not the image format, a notification of not supported image format is shown to the users to notice them there is a wrong file upload which is not image format. Figure 5.11 shows the not supported image format notification.

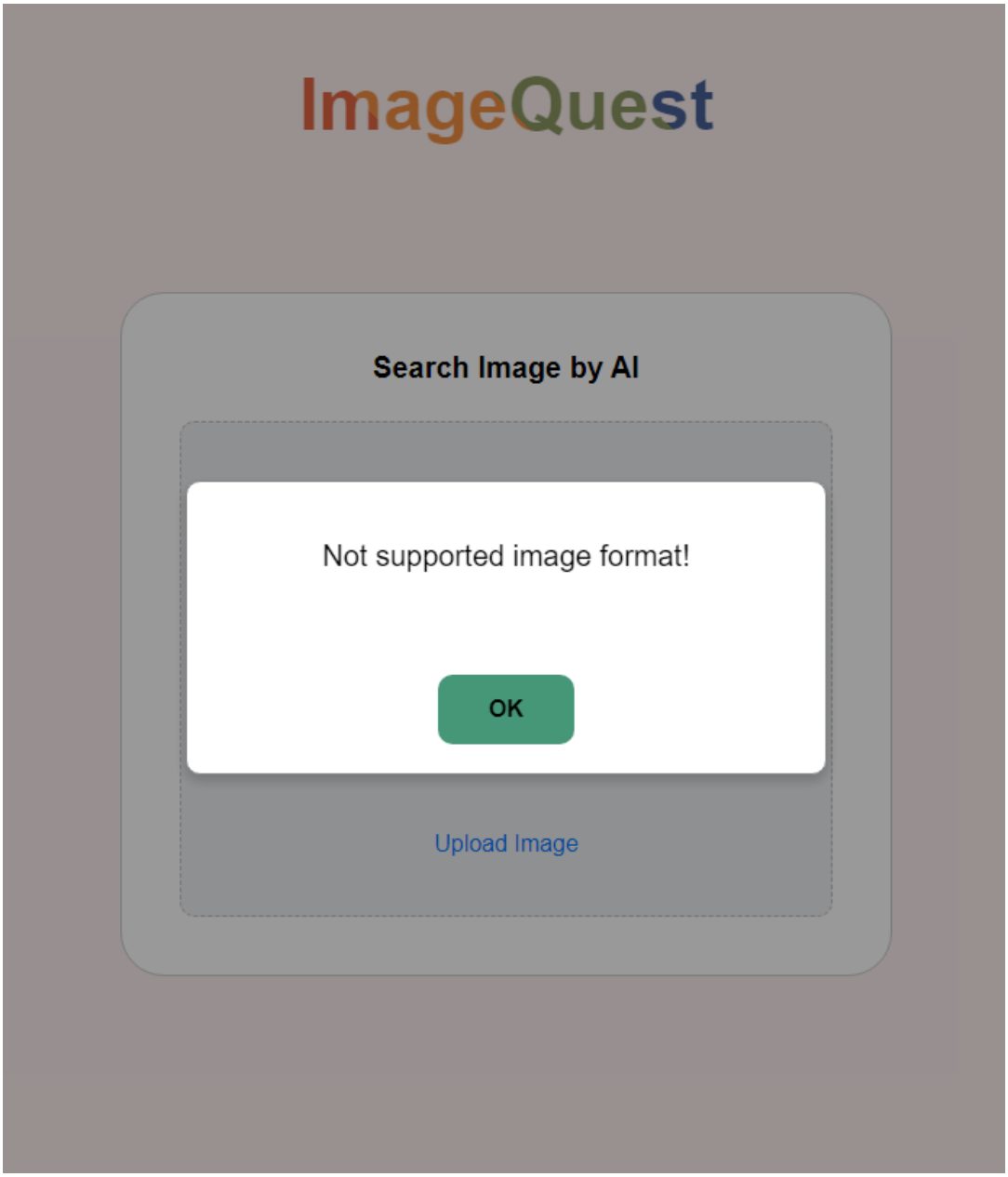

Figure 5.11 Not supported format notification

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This section outlines the implementation of the Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application, separating it into three key areas: frontend, backend and model training. The frontend implementation incorporates the creation of two pages utilizing HTML, CSS, and JavaScript for user interaction. The backend implementation concentrates on handle upload image, image processing, retrieve similar image results from database and API management using Flask. The model training part covers the setup and execution of machine learning model which used for feature extraction and image similarity detection by using PyTorch. A comprehensive overview of the implementation process is presented in this chapter.

6.2 Frontend Implementation

Before the development of each page, the project files undergo a series of initialization settings, which include:

1. Setup of different folders for HTML, CSS, and JavaScript
2. Inclusion of required fonts
3. Management of project assets
4. Initialization of JavaScript functionality
5. Establishment of navigation and routes
6. Setting of project constants

These settings are organized into their respective folders to enhance project maintainability.

6.2.1 Image Upload Page

The image upload page handles the image upload process which allow users to upload their query image and it also function for drag-and-drop interaction, upload image from folder, display error messages and display loading indicator.

6.2.1.1 Drag-and-Drop Interaction

The drag-and-drop functionality enhances the user experience as it provides an easy and efficient way to upload images. This can be achieved by a user dragging an image file into the specific area on the page and automatically have it uploaded for processing. It is home.html file where this feature is implemented and defines drag-and-drop area. The appearance of this area that visually sets it apart from others is done in home.css file. Drag-and-drop events are handled by event listeners in home.js. These include “dragover”, while a file is being dragged over a particular zone, this event prevents defaults browser actions and shows drop area, “dragenter”, by dragging files into the designated zone, similar to dragover, this event highlights the drop area, “dragleave”, when one drags out of this zone for dropping, such action returns back the previous look of them and “drop”, in case the file has been dropped into either place marked as so, this event seizes it again while still returning back other characteristics of such areas before calling up the function which will handle uploading that specific one.

6.2.1.2 Upload Image from Folder

Users can upload pictures directly from folder via the page. An upload link and a hidden file entry element are responsible for this. When the user clicks on the “upload image” link, this action will call up a previously hidden input type=file which will allow the user to search and select an image from their computer. This feature is found in the home.html file where placed the upload link and file input element. The home.css file is what makes sure that the upload link appears as an interactive element. Finally, event listeners for the upload link and file input are found in home.js. When clicked, this trigger opens a dialog box where one can select different files from their computer. When changed, these elements cause capturing of selected document and then calls function for handling such files to begin uploading it.

6.2.1.3 Display Error Messages

At the image uploading and processing stages, there could be several errors which include unsupported formats of images or no similar images in the database. The application handles these situations by displaying error messages in modal dialogs. These dialogues indicate to the user that something is wrong and suggest what should be done next in such a case. The models are defined in home.html file and styled in home.css to ensure they look the same as other parts of application. Show and hide functions for these models are included in home.js. The "showCustomAlert" function pops up a modal with an error message customized to its content. It's triggered when backend API response contains such errors as unsupportable format of image. The "showNoResultAlert" method is particularly meant for notifying users whenever there are no similar images found. This makes it clear for the user about what happened after their query was made.

6.2.1.4 Display Loading Indicator

The loading indicator that is shown to provide feedback on image upload and processing. The users are notified that their action is being processed by the indicator, which prompts them to wait until it is over. To be boldly seen and match the design of the application, this loading screen must be defined in home.html and styled in home.css files, through home.js to control how the loading indicator appears in the page. When file upload starts, main content of the page disappears, and a loading screen comes up showing that uploading and processing are taking place. After upload and processing have been finished, loading screen becomes invisible again while main content returns. If everything goes well with process completion, then user will be directed to results page. In case if any error occurs will display corresponding error message.

6.2.2 Image Result Page

The image result page is where one can get a display of image similarity search outcomes and it normally comprises uploaded picture and retrieved related images from the backend. These

include similar images display, zooming into pictures for better view, download buttons, successful downloaded notifications as well as a ‘back’ link that returns users to upload page.

6.2.2.1 Display Similar Images

The uploaded image is organized and displayed on the page with a grid of other images fetched from the backend. This helps users to visually compare their query image against other similar ones in the database. The layout for displaying input image and similar images is defined by result.html file. It has sections for Input Image, similar images grid and modal dialogues for viewing larger images or downloading them. The result.css file styles the result page to achieve a responsive and organized layout. Flexbox layout in which items are centered within main container, grid layout for similar image display that ensures presentation is neat and well-organized, modal styles that maintain consistency and usability when viewing bigger images or downloading them, and responsive design elements allow the layout to adapt to different screen sizes. Result.js deals with dynamic content updates as well as any user interactions such as fetching similar images from the backend API based on the uploaded image, upload an image to the backend, send a request with both an image and a URL parameter telling where the database is located then process such response so as show these similar images.

6.2.2.2 View Images in a Large Format

Modal dialog box for viewing larger images which enable users easier to see any similar image clearly in large format. Result.html has the definition of the modal dialog box that contains larger images. These include the close button, link to download and display of the big image. When an individual selects a similar picture, open Modal is activated. It opens the modal and sets the URL of the selected image as its source. The user can then click on a close button to have it closed completely.

6.2.2.3 Download Image Result

The user can download any similar image that is displayed on the result page. This feature is incorporated into the modal dialog, enabling users to easily download the image when viewing it. The link to download is contained in the modal dialog defined within the result.html file. It

uses an anchor tag which looks like a button. Setup of downloadButton makes it possible for new link element to be created when user clicks on download button which initiates downloading of the image. Once the downloading starts, both modal and custom alerts are hidden and shown respectively to acknowledge successful downloads.

6.2.2.4 Providing Download Success Notification

To make the user experience better, a notification appears on the screen to inform that the image has been successfully downloaded. This way users know when their download is done. The success notification modal is defined in result.html, and it contains a message and an OK button for closing this notification. The “showCustomAlert” function is called after an image has been downloaded successfully to display the notification modal. To close this notification, the user must click on the OK button which will result in activating the “closeAlertButton” event handler making modal hide.

6.2.2.5 Back to Image Upload Page

There is a back button show in image result page which allow users to return to the Image Upload Page. This provides an improved user experience and navigation by giving users a one-click option to go back and upload another image. A clickable button is in the result.html file as the back button and the style of the back button on the result.css file is designed to make it stand out visually enough to be clicked. This is seen in its hover effects for increased interactions. In order to take users back to the Image Upload page when they are through with their activities, a click on this will bring about such an action; thus, there exists an event listener referring an individual to the prior URL or even a different URL depending on what has been specified in relation to uploading the same image.

6.3 Backend Implementation

Flask is chosen to implement the backend functionality. The reason to choose this framework was based on its simplicity and ability to create adaptable interfaces because this project works with images and connects with MongoDB database. Before adding more features, a link must be established between the backend server and these databases. Hence, it is important to

provide the third-party libraries ‘pymongo’ and ‘gridfs’ with the appropriate configurations for the database.

6.3.1 Upload

Backend server allows users to post images by making a POST. The uploaded images are processed and saved into a MongoDB database using GridFS. The characteristics of an image are extracted and stored for later use in similarity searches. The file that is uploaded is confirmed to be an image by the server itself. In case it is not properly selected or simply not an image, then the server responds with an error. Correctly formatted image files are then processed and stored in a database together with their feature vectors. The picture is first verified by the server before being pre-processed using “preprocess_image” to make it suitable for features extraction later and it is performed by the trained model loaded from a .pt file.

6.3.2 Search and Retrieve Similar Images

The backend server performs the operation of searching for and fetching images that are like a given input image. This is done by extracting traits from the input image and comparing them with those stored in the database. It is this function that computes on how much similarity exists between the input image’s feature vector and their corresponding ones found in database, thus bringing out the most similar images. The searching process includes preprocessing an input image using “preprocess_image”, feature extraction using “extract_features” and calculating similarity scores against stored feature vectors which the features extraction done by the trained model loaded from a .pt file. A threshold level of similarity is used to ascertain which images are returned as results, thereby weeding out less significant matches while ensuring retrieval of only those images that look alike in all aspects.

6.3.3 Download Image

The backend server allows users to download images saved in the database. Images are obtained from GridFS and sent to the client. The requested image has its ID verified by the server, thus getting the image data from GridFS. Thereafter, it sends back the image file to the client including with appropriate MIME type so that it is compatible with different web

browsers as well as client applications. This functionality enables efficient management and retrieval of images within the application.

6.3.4 Save Dataset to Database

The dataset needs to save in the database are run through a pre-trained model by the backend server, which then saves its features to MongoDB database for future similarity searches. The file collection is there for storing image files using GridFS and the other one to store image feature vectors. The server computes unique hash values for each image during upload so that duplicates can be detected. In case such an image with the same existing hash exists in the database, it's ignored hence avoiding duplication of storage. Therefore, efficient use of storage space and integrity of the picture library should be maintained. If a ZIP archive is uploaded, each image inside it is taken out separately, processed and saved individually. The "preprocess_image" function converts each image into a model input-appropriate tensor while "extract_features" uses a trained Siamese network loaded from a .pt file to produce feature vectors and save it into database.

6.3.5 Redirect to Frontend

The backend server also comes with ways of routing users to front-end application. As a result, the front-end and back-end are integrated, making it possible for users to open the application with web interface. The" /" endpoint redirects the user to the homepage of the frontend application. Users can thus easily locate or access the main screen where they carry out various operations such as uploading images as well as searching for similar images. When accessing "/results" page, a user is redirected here from backend by using this route. It takes in an image URL and database URL as parameters and returns search results for similar images through it. It works by providing continuity between backend image processing and frontend display of results.

6.3.6 API List

The backend service provides different API endpoints for the functionalities of upload image, retrieve similar images, download image and save dataset to database. Table 6.1 shows the endpoints, HTTP method, parameters and description of the different routes.

Endpoint	**HTTP method**	**Parameter**	**Description**
/api/upload	POST	file, db_url	Uploads an image and stores it in the database.
/api/get_image/<id>	GET	db_url, image_id	Downloads an image by its ID.
/api/get_similar_images	GET	db_url, image	Retrieves images that similar to the query image.
/api/upload_zip	POST	file, db_url	Uploads dataset from a ZIP file and saves to the database.

Table 6.1 API List of Backend Service

6.4 Model Training

Several steps are crucial in the model training for effective training of the Siamese Network on image similarity tasks. The model training does not take place at the backend though rather, it is done independently to give a .pt file which is used elsewhere in the backend. Some major parts of code related to Model training as well as key components and processes based on the provided code will be outlined such as data preparation, model definition, loss function, training loop and testing.

6.4.1 Data Preparation

Data preparation is important for organizing and preprocessing the dataset before training. The dataset is stored in a directory named “train”. The data_loader.py file is responsible for loading image pairs together with their labels that specify whether the images belong to one class. This “CustomPairDataset” class creates these pairs and applies some forms of augmentation as random horizontal flip, random rotation, color jitter, and random resized crop to make the model more robust. These transformations teach the model how to handle different image distortions and lighting conditions. To illustrate, random horizontal flip flips some images horizontally at random making them look as if they were mirrored while rotating some pictures randomly by some degrees is what constitutes random rotation which helps model identify objects from different directions. Color jitter varies brightness, contrast, saturation and hue of images such that they resemble images taken at various lighting conditions while random resized crop resizes and crops images randomly at a given size to enable models learn from different parts of an image.

The datasets are split into training and validation sets by the data loader. Batching is done with given batch sizes to enable efficient computing. Each batch has a pair of images and their respective labels. The training and validation data loaders have been set up with such parameters as batch size, number of worker threads for data loading, as well as whether to pin memory to speed up data transfer to the GPU. Additionally, it defines the function get_train_validation_loader within its file “data_loader.py” that returns both training and validation data loaders which makes sure that during training loading and preprocessing of data

is done efficiently. In addition, the function get_test_loader uses similar transformations as those used in validation to create a test dataset loader

6.4.2 Model Definition

There is a detailed model definition in the model.py file which is about Siamese Network that uses ResNet-18. However, the deep residual learning framework with 18 layers of this ResNet-18 model has been changed to make it produce feature vectors of size 256 instead of the usual fully connected layer that outputs class probabilities. This size was chosen to achieve a balance between richness of feature representation and computational efficiency.

Several important parts are a part of architecture of modified ResNet-18 in Siamese Network. The base model is ResNet-18, a pre-trained convolutional neural network with 18 layers, consisting of convolutional layers, batch normalization layers, ReLU activation functions, and residual blocks. It allows for easier backpropagation through gradients by enabling gradient flows more smoothly as they flow back through these connections – they are called ‘residual connections. Instead of the last fully connected layer, there is a linear layer that outputs a 256-dimensional feature vector at the end of ResNet-18. To modify this entails dropping out the original fully connected layer and adding another one such that it has input size equal to number of features outputted by ResNet-18 and output size 256 before introducing non-linearity using ReLU activation function. The 256-dimensional feature vector represents the image in a high-dimensional space where similar images have closer representations.

Similarity scores in the Siamese Network are generated by finding the absolute difference between two images’ feature vectors and then putting this result through another fully connected layer, which outputs a single similarity score. The architecture consists of a “forward” method that takes two input images, calculates their feature vectors using forward_once method, computes absolute difference between vectors and passes it through fully connected layer to produce similarity score; it also has a “forward once” approach for processing one image through ResNet-18 to get its feature vector.

This is why it is ideal for one-shot learning situations because unlike traditional models that need lot of data to learn well, one short learning focuses on recognizing new instances only with a single or very few examples. In one-shot learning, the model should be able to recognize or compare pictures using very limited number of samples. A Siamese Network is specially designed for tasks that require comparisons between pairs of inputs and thus highly applicable in such scenarios as one-shot learning. Comparing feature vectors allows the Siamese Network detects picture resemblance where data is scarce. This technique is especially relevant when it is difficult to obtain numerous samples with labels, for example in face recognition or signature verification. It leverages Siamese Network to generalize from very limited examples making it a powerful way for tasks that have a small amount of data.

6.4.3 Loss Function

While training, BCE is the loss function used with logits. The BCE loss is crucial in binary classification tasks such as determining if two images are similar or not. It calculates the difference between predicted similarity scores and actual labels from image pairs. The formula for BCE loss is:

$$\text{BCE Loss} = -[y\log(\sigma(x)) + (1-y)\log(1-\sigma(x))]$$

The sigmoid function applies to the similarity scores $\sigma(x)$. Sigmoid function brings down input scores to a range of 1 and 0 which represent probabilities. Choosing BCE loss as our preferred option effectively penalizes incorrect predictions thus driving the network toward generating close feature vectors for similar images and far apart ones for different images. This loss helps in learning a robust metric for image similarity by the network. The aim is to minimize this loss which means accurate predictions on whether pairs of images are similar. By using BCE Loss, the Siamese Network learns to distinguish between images that are alike and those that are different making it an essential constituent in training process for image similarity tasks.

6.4.4 Training Loop

The whole training and validation processes are taken charge by the file trainer.py. It is upon this file to control everything that happens during training and validation. It loads data,

initializes a model, specifies an optimizer and sets learning rate scheduler, implements training and validation loops as well. The training loop has several critical steps: First, the Trainer class initializes the training process by loading configuration settings – this could be done by preparing data loaders for processing image pairs through augmentation and transformations. This includes various parameters like batch size, number of epochs, learning rate, optimizer type (Adam or SGD). Then, additional information is required to train on more than one observation at a time using mini-batch gradient descent. To do so, we need to load in each batch both training and validation data by using data loaders which would entail any necessary transformations and augmentations. This will ensure that each iteration of the training set will see different image pairs making the model generalize better.

The Siamese Network processes the input image pairs during a forward pass for similarity scores generation. While feeding the images through a modified ResNet-18, computing their feature vectors, and calculating the absolute difference between these vectors, the images are passed through to attain similarity scores on them by fully connected layer. To arrive at these predicted similarity scores, one computes BCE loss that is the difference between them and actual labels. The reason is that it acts as a measure of error in predictions hence drive optimization process. Gradients are computed via backpropagation and used to update model parameters. During backpropagation gradients of loss with respect to model parameters are calculated which helps optimizer adjust weights towards minimizing losses. Given this configuration, Adam or SGD can be used as optimizers for updating model parameters using these computed gradients. Its adaptive learning rate could make Adam more desirable than others because this may result in faster convergence.

To save memory and computational time, mixed precision training is used with PyTorch's automatic mixed precision package (AMP). Mixed precision training means that both 16-bit and 32-bit floating point numbers can be used, providing the ability to reduce consumption of memory as well as accelerate computations while not affecting model accuracy adversely. The learning rate is adjusted by the “ReduceLROnPlateau” scheduler that lowers it when validation accuracy plateaus thus helping better convergence. This scheduler prevents overfitting and therefore allows further model improvements even after early rapid gains. If there is a GPU available, then during the training process its computational power is also

harnessed to speed up calculation and manage bigger batch sizes. The forward and backward passes are accelerated by the GPU which dramatically reduces the training time.

6.4.5 Testing

After training, the model's performance is appraised on a different test dataset. The Trainer class method, test loads the saved best model during training and runs it over the test data. The evaluation also involves computing and displaying some of the outcomes of tests such as accuracy which measures how good the model can generalize its operation when exposed to novel unseen information. This stage is important in guaranteeing that the model goes past training dataset and be utilized for accurate determination of image similarity in practical situations. In testing phase, forward passes are made through the network and final accuracy is calculated by comparing predicted similarity scores with actual labels. This exhaustive testing guarantees that there exists real world application where it should determine likeness between previously unseen images other than being only efficient on training data sets.

6.5 Implementation Workflow

This section reveals how detailed workflow of the web application is run, explaining how data, requests as well as replies travel through the system beginning from the initial user interaction to come up with a final result. In this regard, such elements that are important are highlighted by this workflow, which involves interactions between front-end and back-end components.

The workflow begins when a user interacts with the front-end of the system managed by HTML and JavaScript files (home.html, home.js, result.html and result.js). This process starts when a user selects or drags an image into the upload area on the home page (home.html). The home.js script captures this event and prepares a FormData object containing the image file plus all other additional information such as database URL. This FormData object is then sent to the server using a POST request to /api/upload endpoint via Fetch API.

After receiving the POST request on /api/upload endpoint, Flask server (test.py) checks if the uploaded file is an image and verifies its format for suitability of further operations. Should it meet the validation rule, then the said web application will use Pillow library to convert into RGB format. In this case, GridFS temporarily holds such processed images enabling efficient handling of vast files in an integrated manner with MongoDB. Hence, a unique file identifier is created for this image to be used during subsequent processes.

This followed by resizing and normalizing pixel values using preprocess_image function to get correct model ready formats and scales for the image. The preprocessed image is then passed through a Siamese Network as defined in model.py which extracts its feature vector. The network takes in an image as input and uses a modified ResNet-18 architecture to produce a 256-dimensional feature vector. This model is loaded from a .pt file that contains both trained weights as well as architecture of the network, ‘model’. To obtain that feature vector, forward_once method under extract_features function calls up on this model.

The extracted feature vector is then checked against stored feature vectors in the MongoDB database. The find_similar_images function gets all stored feature vectors and computes similarity scores using cosine similarity. This includes determining the dot product and norms of vectors so as to find out the cosine similarity between the uploaded image and stored images. Similarity scores are sorted in a descending order while only those that exceed a set similarity threshold are considered as valid matches. The file IDs of these similar images are collected for inclusion in the response.

Then, server provides JSON response with URLs of similar images, URL of uploaded image, and processing time. When no similar images are found, an error message appears in the response. This JSON response is sent back to the client. The response is handled by home.js script on the client side. If any similar images were found, user will be redirected to result page (result.html) containing relevant data included in URL parameters. Fetching and displaying similar images from the responses are among the functions of result.js on the results page. This means that a user can see them or do some actions with these pictures like clicking on one to

view a larger version in a modal. Thus, more things such as downloading the images are done here.

This approach guarantees smooth user experience when they upload an image and get shown related ones from the database. Every single phase is created in a way that enables efficient data handling coupled with being responsive to users' needs, as opposed to model's feature extraction, which would generate appropriate resemblance scores by relying on the pre-trained .pt file.

6.6 Summary

In conclusion, the Intelligent Image Search Engine with AI-Based Similarity Detection for Web Application could be improved in terms of algorithm efficiency, data handling and model optimization. To make it compute faster and use less computational power, thus having a shorter response time and better accuracy, an improvement can be made by refining the data preprocessing steps, optimizing the model training process, and enhancing the integration between the frontend and backend.

CHAPTER 7

TESTING

7.1 System Testing

In this section, the testing activities that took place in developing the Intelligent Image Search Engine with AI-based similarity detection for a web application are reviewed. The tests mainly concentrate on code/application testing, specifically on four critical domains: unit testing, functional testing, integration testing, and API testing.

7.1.1 Test Plan

Test plan plays an important role in enhancing the efficacy and effectiveness of the testing process. It describes strategies, scope, resourcing, and schedule of the testing activities. The main aim is to discover and repair defects that ensure that the system meets specified requirements.

7.1.1.1 Test Scope

The scope of the testing involves both frontend and backend components of web application. Testing activities are grouped into basic unit testing, functional testing, integration testing, and API testing. These exercises are necessary for checking individual components, assessing how well the entire system works from a user's perspective as well as examining how different parts interact with each other in one web application. This scope ensures that there is a comprehensive coverage on all features and functionalities of the web application.

7.1.1.2 Test Basis

The test basis contains project requirement specification document, system use case diagram, system entity relationship diagram (ERD), and system API list. They serve as reference points for creating detailed test cases that cover all aspects of a system. Through these foundational

documents, test cases can be aligned with objectives of the project thereby making sure that all essential functionalities are tested.

7.1.1.3 Test Items

The modules or services for testing are the combination of both frontend and backend. The frontend section looks at Home Page and Result Page. Users have a direct effect on these two crucial interfaces hence making sure they work properly is important for users. In the backend, the following modules are tested, Image Upload, Image Search, and Image Processing. It involves main activities of the application like user uploads processing images and returning similarity results respectively. The application's major functionality depends on them hence it is necessary that they function correctly. Table 7.1 outlines the modules or services of both frontend and backend.

Frontend	
Module Name	**Description**
Home Page	The initial page that users see when they navigate to the web application. It should load quickly and present the user with an interface to upload images.
Result Page	The page that displays the results of the image search, including similar images and their details. It should be accurately populated based on the user's query.
Backend	
Service Name	**Description**
Image Upload	Handles the uploading of images from the user. It should accept various image formats and store them for processing.

Image Search	Processes the uploaded images to find similar images based on AI algorithms. It should return relevant results efficiently.
Image Processing	Applies AI-based similarity detection to the uploaded images. It should be accurate and fast in analyzing image features.
Image Download	Handles the retrieval and downloading of processed images. It should ensure that images can be downloaded successfully by the users.

Table 7.1 Modules or services to be tested

7.1.1.4 Test Strategy

Behavioral testing is used as a test strategy whereby it examines external behavior of an application without considering its internal design. In this approach, black box techniques are employed to ensure that system response is correct regardless of different types of inputs. It can be done by focusing on input/output of an application whereby we can check if an application behaves under different scenarios as expected. This method allows finding functional problems and ensures if the software meets its required specification.

7.1.1.5 Test Conditions

The tests are always conducted under well-characterized conditions. Unit testing are run by developers in a manual way using several tools as unit test for Python. These kinds of tests make sure that isolated components behave exactly as expected. Functional and integration tests rely on automation tools such as Selenium for browser automation to mimic user interactions and check if the application is operating properly. API tests will use Postman to handle the request and response to check the API can work success to show status 200. These tests confirm how the application is supposed to work from the perspective of end-users and whether it functions as an integrated system.

7.1.2 Entry and Exit Criteria

Starting point and ending point of testing phase is essential in entry and exit criteria. It structures the testing process, so that before starting or completing any test there are some pre-requisite conditions that need to be satisfied according to these criteria.

7.1.2.1 Entry Criteria

- There is a detailed test project schedule which lays down the timelines and milestones for testing activities.
- All functionalities are fully developed and are running consistently making the application ready for testing.
- The required resources, including test databases and configurations, have been established in the testing environment to facilitate testing.

7.1.2.2 Exit Criteria

- All planned tests cases executed and passed showing that the application is working as it should be.
- All known errors are fixed, verified and this makes sure no serious problems are found in the application.
- No critical defects yet not closed meaning that all major issues were resolved.

7.1.3 Unit Testing

Purpose of the basic unit testing is to verify that individual units or components of the application works as expected. The framework used for unit tests in this web application was unit test in Python. Some of these tests include loading webpage, handling image upload requests and checking whether required parameters are missing.

The Home Page Loading Test confirms that the home page is getting loaded properly by sending a GET request to its URL and then verifying response status code. It should return a status code of 200, meaning it successfully loaded. This test makes sure that the basic functioning of the web server is operational so that one can access the home page.

The Image Upload Test checks if images uploaded to this app can be handled by sending a POST request with an image file to upload endpoint and then check for response status code. In this case, one expects a status code of 200 together with a success message which indicates that image has been rightly uploaded. This test ensures that uploading function operates well enough also allowing processing of picture files through applications.

The Error Handling Test is a verification process that checks if the application is able to handle missing parameters, by sending an incomplete data POST request and verifying the response status code. The expected result is a status code 400 with an error message which shows that the application has correctly identified and reported the missing parameters. This test verifies whether the application is capable of handling mistakes without alarming users. Table 7.2 shows the test case of unit testing.

Test Case	Test Detail	Test Instruction	Test Data	Expected Result	Actual Result	Condition
UT01	Load Home Page	Send a GET request to the home page URL	-	Response status code shows 200	HTTP request sent	Pass
UT02	Upload Image	Send a POST request with an image file	test.jpg	Response status code shows 200	HTTP request sent	Pass
UT03	Handle missing parameters	Send a POST request with incomplete data	1.txt	Response status code shows 400 and an error message	Error message sent	Pass

Table 7.2 Test Case of Unit Testing

7.1.4 Functional Testing

Functional testing in this case is defined as a process that entails verifying whether an application works according to the specified requirements and performs the anticipated tasks. Selenium was used for browser automation in these functional tests to simulate user actions. All these included opening the home page, uploading images and checking what has happened to the page contents after that.

The Home Page Navigation Test allows users open web application on a browser and go to home page, expecting home page title will match with expected title. This test assures accessibility of User Interface (UI) and correct rendering of home pages.

The Image Upload Interaction Test is intended to check if the users can upload images through the web interface by using Selenium for image upload automation. The output should be a success message and proper processing of uploaded image(s). It checks whether upload feature works from users' standpoint or not and if application can handle pictures correctly.

To verify that the result page shows the correct information, we perform a verification test for the Result Page. After uploading an image and reviewing the displayed information, you will then navigate to the result page. The expected result is accurate display of image similarity results. This test checks if uploaded images are correctly processed by the application and accurate output given.

The image download verification test is used to verify whether there is a possibility of downloading an image from the search results. By clicking on a related image and downloading it, users can perform this test. When the download is successful, then the

expected result will be that the picture has been downloaded. This test ensures that the user's ability to operate download works properly. Table 7.3 shows the test case of functional testing.

Test Case	Test Detail	Test Instruction	Test Data	Expected Result	Actual Result	Condition
FT01	Navigate to home page	Open the web application and navigate to the home page	-	The home page title matches the expected title	Title matched	Pass
FT02	Upload Image through UI	Use Selenium to automate the image upload process	test.jpg	Success message and correctly processed image	Image uploaded	Pass
FT03	Verify result page content	Navigate to the result page after uploading an image	-	Correct display of image similarity results	Results displayed	Pass
FT04	Verify image download	Click on a similar image and download it	First similar image result	Image should be downloaded successfully	Image downloaded	Pass

Table 7.3 Test Case of Functional Testing

7.1.5 Integration Testing

Integration testing is intended to check that different parts of the application work well together and that the entire system functions as it should. Integration tests involve checking up on a complete workflow from image upload to similarity detection and ensuring that the frontend interacts with the backend services including database and image processing.

Through uploading an image, running it through the system and verifying result page display similarities, The Full Workflow Test checks over the overall process starting from an image upload and finishing at displaying similarity results. What is expected is that an image will be uploaded, processed successfully and results shown suitably. It ensures that every process happens as it should across all components for smooth operations.

By uploading multiple images in one request and checking the response, The Multiple Image Upload Test evaluates how multiple uploads are handled at ago. As a result, all images should be sent without complications during submission or retrieval. Thus, this test ensures that numerous uploads can be carried out by an application effectively without any glitches in these processes.

This test was meant to confirm that the application had been instructed well to interact with the correct database. It is uploading an image and then checked for features stored in them, to determine if it’s working in accordance with the expected result by confirming whether images’ features were stored properly in databases. Thus, this test confirms whether the application interacts appropriately with its databases and see to it that all required details are being put in place.

The aim of the image download test is to ensure that a user can download an image successfully after it has been uploaded and processed. It requires uploading an image, processing it, getting its ID and finally downloading the image using this ID. The idea here is that the frontend and backend components work synchronously in achieving this purpose, as well as the expectation that the download functionality works correctly: if such a thing happens, then one can be sure about successful downloading of an image. Table 7.4 shows the test case of integration testing.

Test Case	Test Detail	Test Instruction	Test Data	Expected Result	Actual Result	Condition
IT01	Test full workflow	Upload an image, process it, and check the result page	test.jpg	Image uploaded, processed, and results displayed correctly	Results correct	Pass
IT02	Retrieve similar images	Upload an image, get image ID, and retrieve similar images	test.jpg	Similar images retrieved successfully	Similar images retrieved	Pass
IT03	Test database interaction	Upload a zip folder and check the database	train.zip	Image features stored correctly in the database	Data stored	Pass
IT04	Test image download	Upload an image, process it, get image ID and download	test.jpg	First similar image result downloaded successfully	Image downloaded	Pass

Table 7.4 Test Case of Integration Testing

7.1.6 API Testing

API testing ensures that the backend services of an application are running correctly and can handle requests and responses efficiently. It involves testing endpoints for uploading pictures, uploading zip files, getting similar images, and downloading images. The aim is to make sure that the API endpoints function seamlessly while yielding expected results. The following API tests were performed using Postman.

The upload image endpoint uploads a single image file through endpoint POST /api/upload. This endpoint takes in two parameters, file which is the image file to be uploaded and db_url which is the database connection URL. After this upload, it should return an expected response of a successful upload of the image and JSON response with image ID.

The upload zip file endpoint on the other hand uploads zip file with multiple images via endpoint POST /api/upload_zip. This endpoint has two parameters namely file being the zip folder to be uploaded and db_url which is the database connection url. The result will be a successful upload of the zip file and JSON response indicating success should be returned.

The retrieve similar images endpoint is used to retrieve similar images through endpoint GET /api/get_similar_images. This endpoint has two parameters which are image and db_url. The expected response should be that similar images are successfully retrieved and a JSON response with image data is returned.

The download image endpoint, GET /api/get_image/{image_id}, allows users to download an image using its ID. Db_url as well as other parameters are required by this endpoint. The anticipated response is that the image can be downloaded successfully. Table 7.5 shows the test case of API testing.

Test Case	Test Detail	Test Instruction	Test Data	Expected Result	Actual Result	Condition
API01	Upload image	Send a POST request to '/api/upload/ with an image file and db_url	image file, db_url	Image uploaded successfully, and a JSON response containing the image returned	Image uploaded successfully	Pass
API02	Upload zip file	Send a POST request to '/api/upload_zip' with a zip file and db_url	zip file, db_url	Zip file uploaded successfully, and a JSON response indicating success returned	Zip file uploaded successfully	Pass
API03	Retrieve similar images	Send a GET request to '/api/get_similar_images; with the image ID and db_url	image ID, db_url	Similar images retrieved successfully, and a JSON response containing the	Similar images retrieved successfully	Pass

				image data returned		
API04	Download image	Send a GET request to '/api/get_image/{image_id}' with the db_url	image ID, db_url	Image downloaded successfully	Image downloaded successfully	Pass

Table 7.5 Test Case of API Testing

7.2 Model Testing

Model testing is a phase for validation of machine learning models to evaluate and compare the performance of several convolutional neural network (CNN) architectures as base models within the Siamese Network architecture. The primary objective is to identify the most effective model for feature extraction from the Kaggle 30types of balls dataset, which has 3595 images. The models tested include ResNet18, VGGNet16, AlexNet, LeNet, and GoogleNet V3. Each of these models brings unique architectural features and strengths to the table, making them suitable for different aspects of image classification tasks. By using these models within a Siamese network architecture, leverage their pre-trained capabilities on large-scale datasets (such as ImageNet) to enhance the feature extraction process, which is crucial for tasks like similarity detection and image matching.

7.2.1 Evaluation Criteria

The base models are tested under same conditions to ensure a fair comparison which 50 epochs for train set and 10 epochs for test set. The key metrics for the evaluation are:

7.2.1.1 Train Set

Process Time: The whole duration during which the model processes all data in a dataset.

CPU Time: The time that is utilized by CPU when performing both training and inference.

Highest Accuracy: The highest fraction of images correctly classified out of total number tested

Average Accuracy: The average of fraction of images correctly classified out of total number tested per epoch.

Average Training Time: The mean period it takes to complete one cycle over training data.

Average Loss: The average of evaluation of how well predictions made by the model compare with actual results per epoch.

Average Precision: The average of true positives divided by all positive predictions generated by a model per epoch.

Average Recall: The average of true positives divided by true positive instances present in dataset per epoch.

7.2.1.2 Test Set

Average Test Accuracy: The average of fraction of images correctly classified out of total number tested per epoch.

Average Test Loss: The average of evaluation of how well predictions made by the model compare with actual results per epoch.

Average Test Precision: The average of true positives divided by all positive predictions generated by a model per epoch.

Average Test Recall: The average of true positives divided by true positive instances present in dataset per epoch.

7.2.1.3 Dataset

The dataset was the Kaggle 30 Types of Balls dataset, which contains 3595 pictures of different ball types. In this regard, it is a very diverse dataset that can be used to evaluate how CNNs perform at feature extraction. (Gerry, 2023)

7.2.1.4 Evaluated Models

ResNet18: Residual connections are utilized in this network to address the vanishing gradient problem and make it effective in training deep networks.

VGGNet16: It is famous for its simplicity and depth as well as using consistent convolution layers to extract detailed features.

AlexNet: One of the oldest designs of deep architectures, it remains efficient despite having a relatively shallow structure compared to modern counterparts.

LeNet: A simple early version of CNN architecture made for simple classifying images used here as a base line.

GoogleNet V3 (Inception V3): It is characterised by complex architecture with an inception module enabling efficient multi-scale processing.

The purpose of computational efficiency and overall model performance among these models through Siamese Network architecture tests for image feature extraction tasks is to identify the best blending of accuracy. These tests will provide useful pointers on whether they are appropriate for other tasks like this and help in deciding how to choose and optimize models in the future. Next sections will explore a more detailed evaluation of each model using both quantitative metrics and qualitative analysis of their performance. There will be an extensive critique of the strengths and weaknesses of all models with reference to our application so that we can establish the most ideal architecture.

7.2.2 ResNet-18

In the case of ResNet that pre-trained on ImageNet employed the variant named Residual Networks family because of skip connections that guard against the vanishing gradient problem, it is a preferred option for deep networks. The architecture contains 18 layers and uses residual blocks to enable efficient training and improve performance.

For the train set evaluation metrics of ResNet18 model, average processing time was found to be 5171.97 seconds while average CPU time was given as 5171.97 seconds. The highest accuracy got to 0.9867 with an average accuracy of 0. 9800.The average training time per epoch was 95.51seconds while the loss averaged at about 0. 1390.For precision and recall, this model showed impressive scores averaging at approximately one. Ideal for this dataset's feature extraction is what these facts suggest about efficiency of ResNet18. The Figure 7.1, Figure 7.2, Figure 7.3 and Figure 7.4 shows a visual representation of the model’s performance for train set.

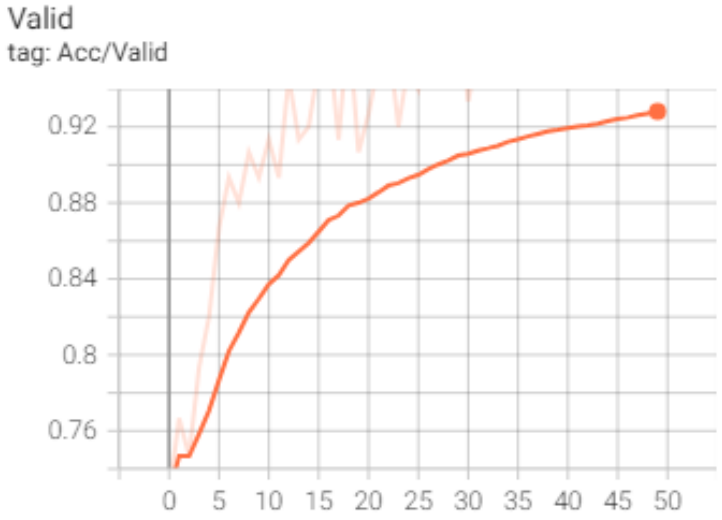

Figure 7.1 Training Accuracy over 50 Epochs (ResNet)

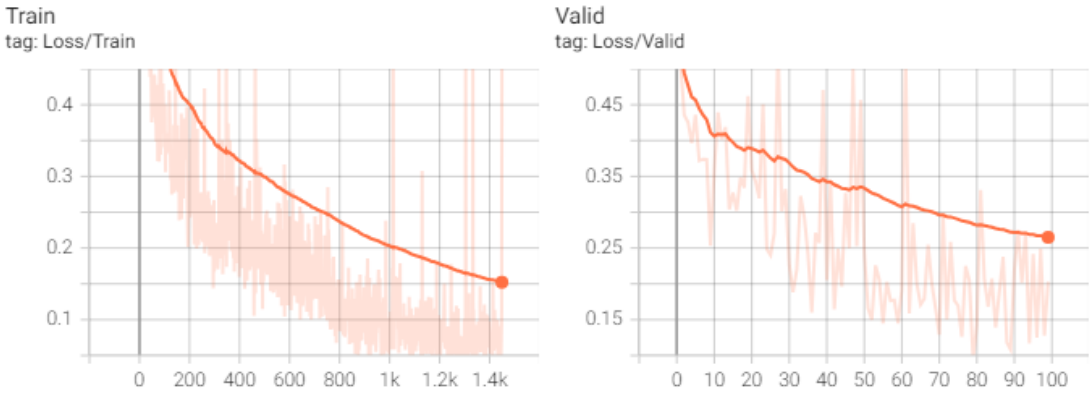

Figure 7.2 Training and Valid Loss over 50 Epochs (ResNet)

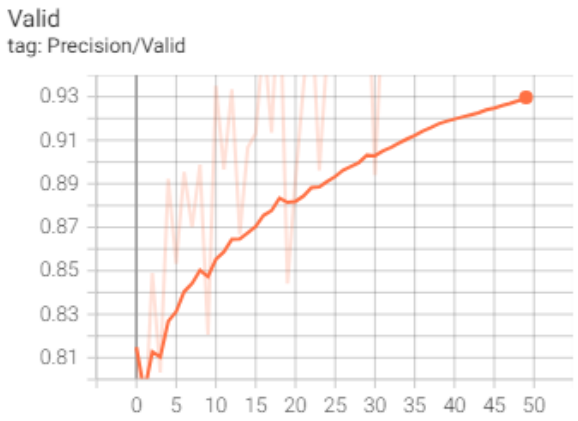

Figure 7.3 Training Precision over 50 Epochs (ResNet)

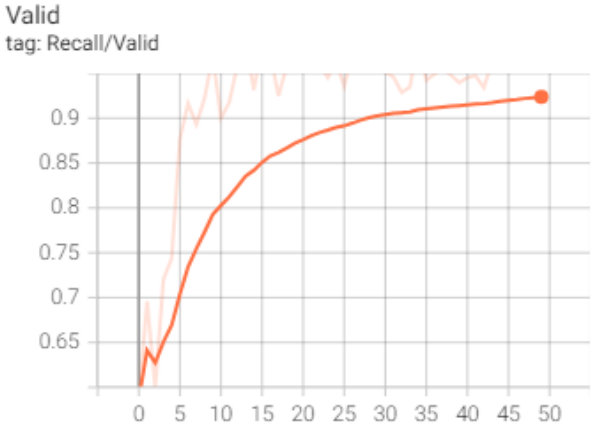

Figure 7.4 Training Recall over 50 Epochs (ResNet)

The metrics that evaluated the test set of ResNet18 indicate a high degree of efficiency. On average, test accuracy was 0.9260 and average test loss was 0.1845. Average precision and recall were also high, with 0.9518 and 0.8940 respectively. These results imply that the model generalizes well to new data as well as exhibits high accuracy and efficacy in feature extraction on the test dataset. The Figure 7.5, Figure 7.6, Figure 7.7 and Figure 7.8 shows a visual representation of the model's performance for test set.

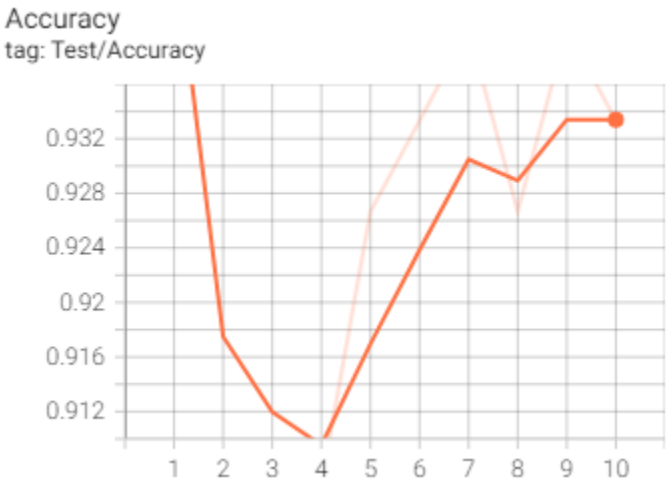

Figure 7.5 Test Accuracy over 10 Epochs (ResNet)

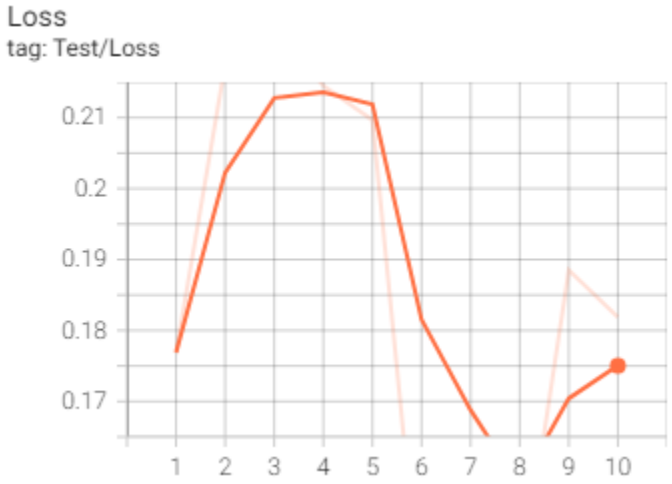

Figure 7.6 Test Loss over 10 Epochs (ResNet)

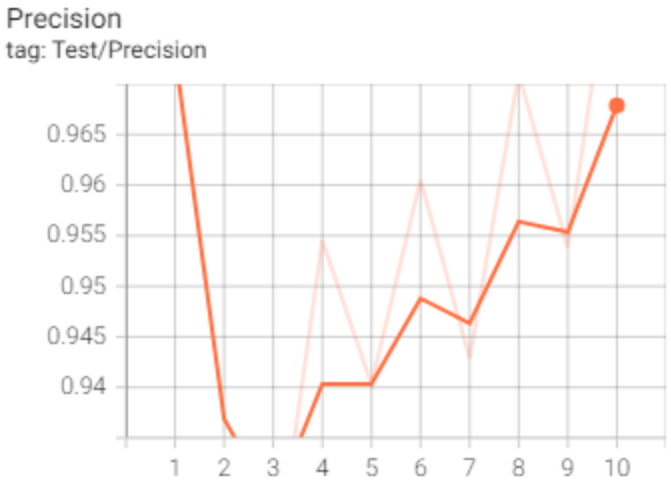

Figure 7.7 Test Precision over 10 Epochs (ResNet)

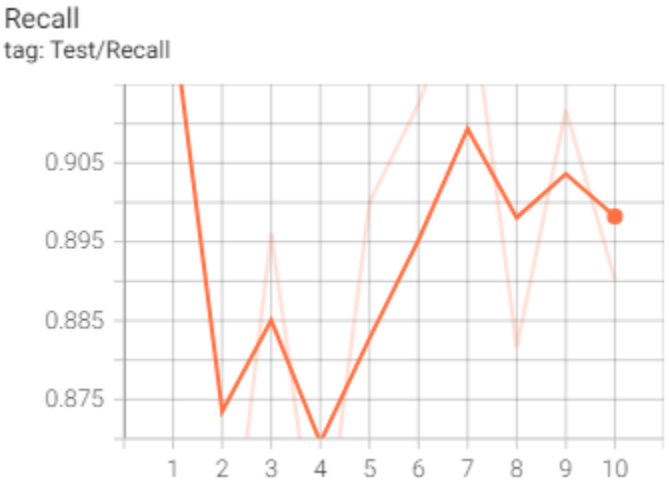

Figure 7.8 Test Recall over 10 Epochs (ResNet)

ResNet18's deep architecture and residual learning abilities account for its high accuracy and low loss. The presence of skip connections in ResNet18 prevents the vanishing gradient problem hence allowing better features to be learned by the model from data. Additionally, impressive precision and recall values suggest that ResNet18 can accurately identify images within this dataset. Finally, it is highly efficient in feature extraction which makes it a strong candidate for this task.

7.2.3 AlexNet

AlexNet, another model used for evaluation purposes, is renowned as one of the models that have contributed to developments in deep learning throughout history. AlexNet was pre-trained on ImageNet as well and subjected to testing with similar conditions with those applied to ResNets. AlexNet's architecture consists five convolutional layers followed by three fully connected layers making it deeper than LeNet but simpler compared to modern architectures like ResNet and Google Net.

Finally, for the train set evaluation metrics the average total time taken by AlexNet was 2781.74 secs while the average CPU time was also 2781.74 seconds having highest ever accuracy of 0.9267; averaged at 0.8600.The average training time per epoch was quite short at 51.87 seconds which is an indication that it takes relatively shorter time during training as compared other models used during our project.Their average loss was found out to be 0.2908 whereas their precision averages were recorded at 0.9167 and their recall averages stood at 0.7746 respectively. The Figure 7.9, Figure 7.10, Figure 7.11 and Figure 7.12 shows a visual representation of the model's performance for train set.

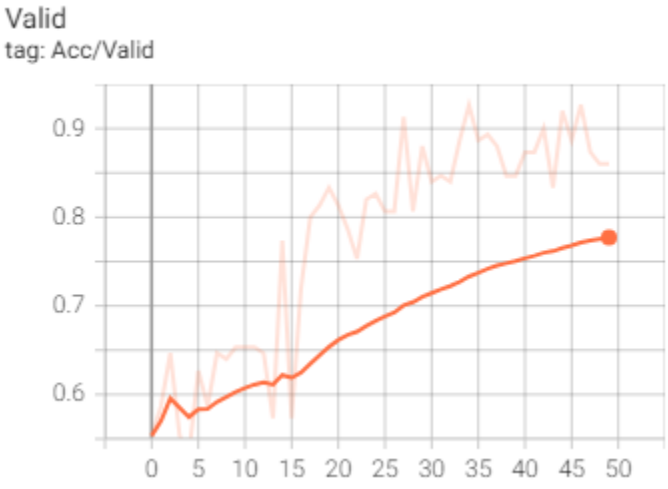

Figure 7.9 Training Accuracy over 50 Epochs (AlexNet)

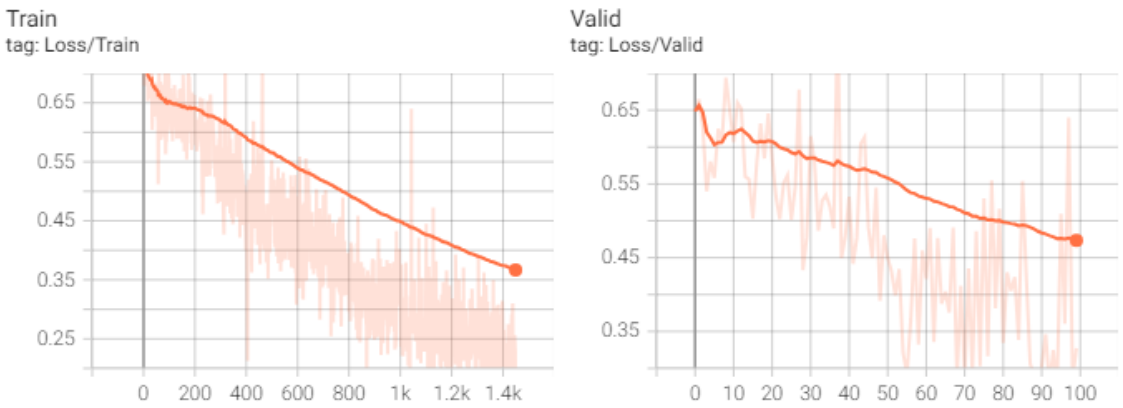

Figure 7.10 Training and Validation Loss over 50 Epochs (AlexNet)

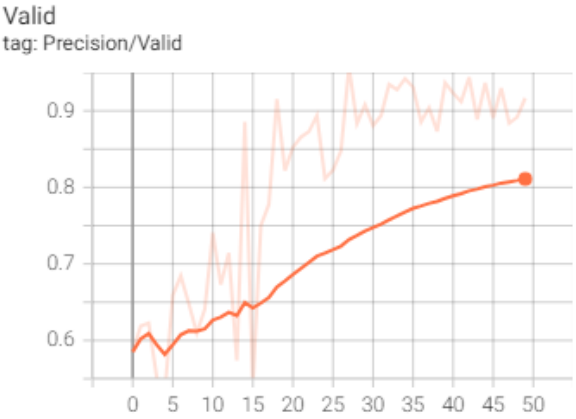

Figure 7.11 Training Precision over 50 Epochs (AlexNet)

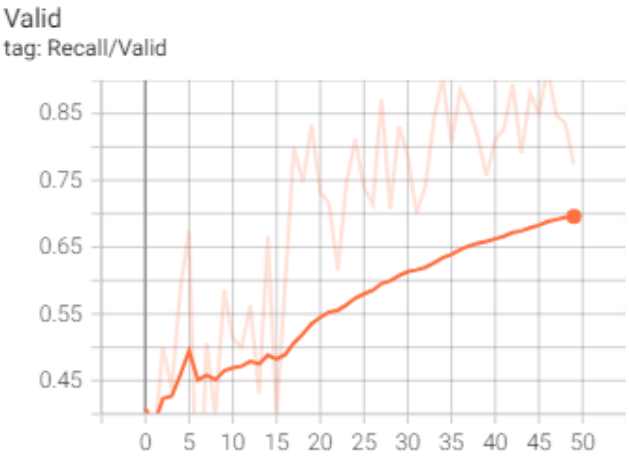

Figure 7.12 Training Recall over 50 Epochs (AlexNet)

The test set evaluation metrics for the AlexNet model shows a strong performance, although with some drop from training set metrics. The average testing accuracy was 0.8647 and on average, the test loss was 0.3607. Results also indicate that the precision and recall scores are high since an average precision of 0.8583 and an average recall of 0.8739 were recorded respectively. These results suggest that the model still maintains good generalization capacity and performs well in feature extraction in the test dataset. The Figure 7.13, Figure 7.14, Figure 7.15 and Figure 7.16 shows a visual representation of the model's performance for test set.

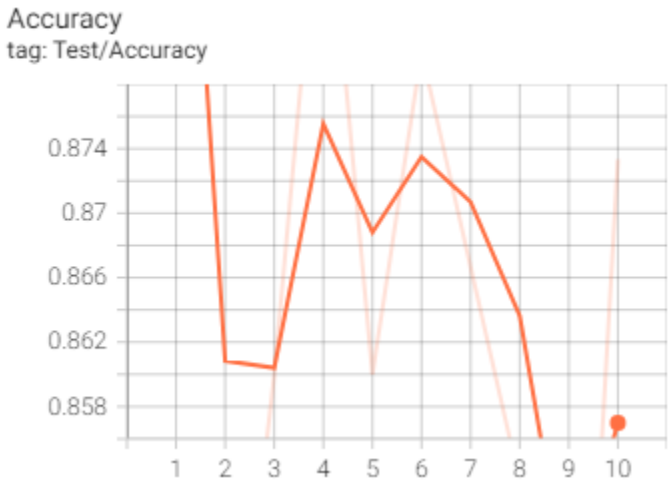

Figure 7.13 Test Accuracy over 10 Epochs (AlexNet)

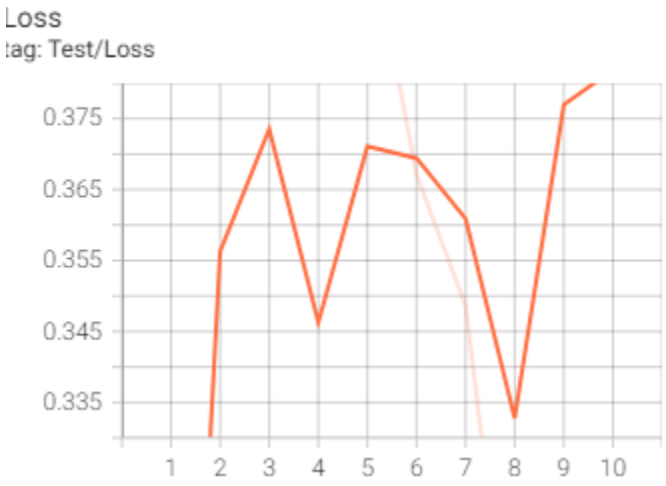

Figure 7.14 Test Loss over 10 Epochs (AlexNet)

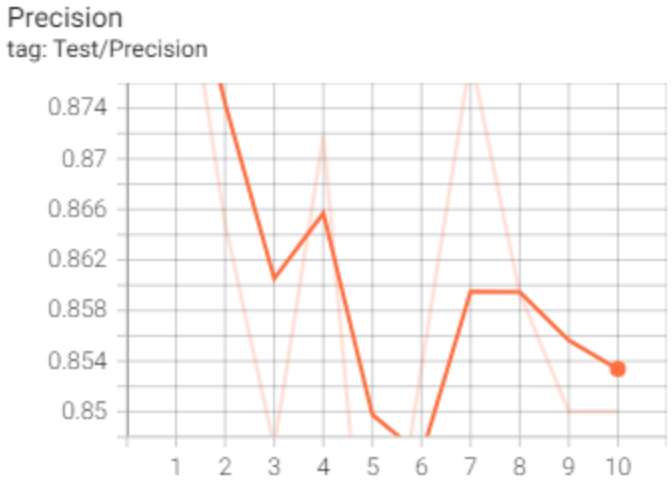

Figure 7.15 Test Precision over 10 Epochs (AlexNet)

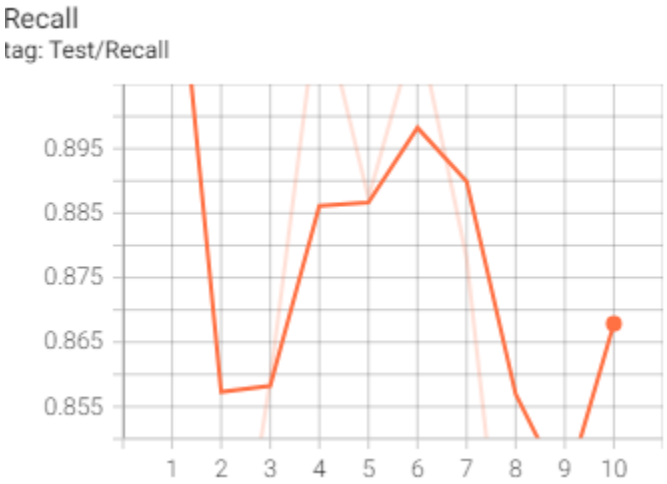

Figure 7.16 Test Recall over 10 Epochs (AlexNet)

AlexNet results indicate a good performance with accuracy and precision, but not as much compared to ResNet18. This means that it is an option due to its reasonably loss and per epoch training time albeit not as best as ResNet18. Nevertheless, AlexNet's architecture may be effective but might not be as strong in extracting features from more complex datasets thus the lower recall value than that of ResNet18.

7.2.4 LeNet

LeNet, one of the foremost convolutional neural networks for recognizing digits, was also tested. Including such a basic architecture would inform on how well it stands against more complex architectures. It has two convolutional layers before two fully connected layers which makes it have fewer hidden neurons compared to all other models examined.

The average processing time for LeNet was 2513.11 seconds with the same average CPU time consumed by this model. The highest achieved accuracy was 0.7267 while that of the model averaged at 0.6800 respectively. The shortest training time on average for any of the tested models was seen in this model with 55.19 seconds spent per epoch on overedits average loss value was observed to be 0.6232; whereas precision averages were of about 0.6714 and recalls were about 0.6528 which are above their respective averages. The Figure 7.17, Figure 7.18, Figure 7.19 and Figure 7.20 shows a visual representation of the model's performance for train set.

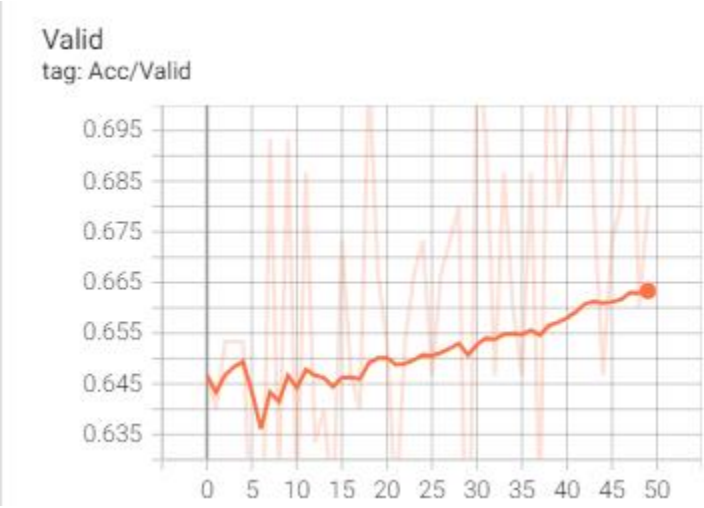

Figure 7.17 Training Accuracy over 50 Epochs (LeNet)

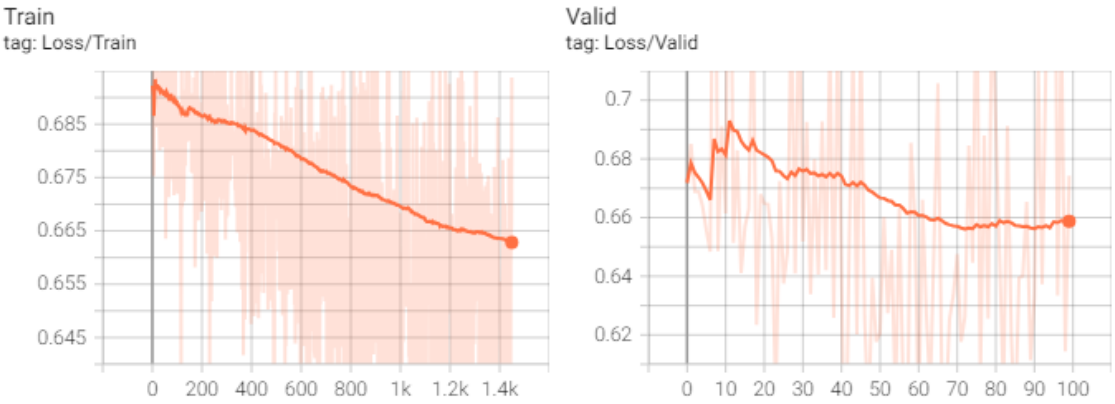

Figure 7.18 Training and Validation Loss over 50 Epochs (LeNet)

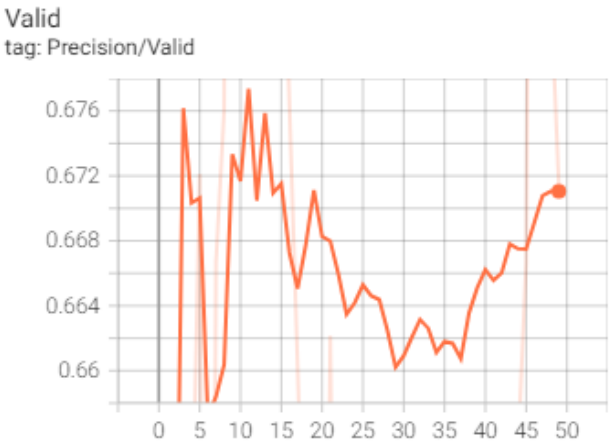

Figure 7.19 Training Precision over 50 Epochs (LeNet)

Figure 7.20 Training Recall over 50 Epochs (LeNet)

For test set evaluation metrics. The average test accuracy was 0.6640 and the average test loss was 0.6441. For precision and recall, their scores were 0.6400 and 0.7367 respectively. This finding implies that in terms of precision-recall balance on the trial dataset, the model is acceptable, but it displays distinct underperformance vis-à-vis training set. Thus, there could be overfitting risks thus giving some pointers on how to improve the model like using regularization techniques or diversified training data sets to address such issues. The Figure 7.21, Figure 7.22, Figure 7.23 and Figure 7.24 shows a visual representation of the model's performance for test set.

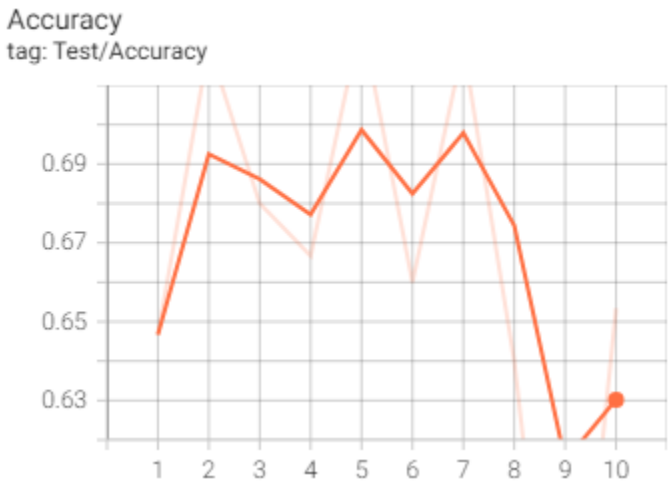

Figure 7.21 Test Accuracy over 10 Epochs (LeNet)

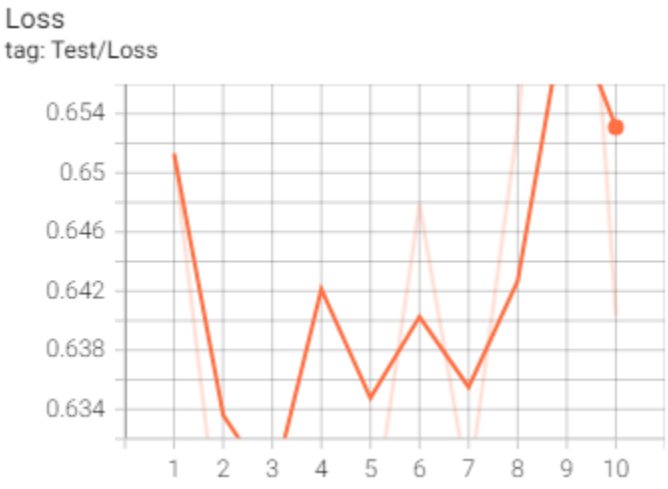

Figure 7.22 Test Loss over 10 Epochs (LeNet)

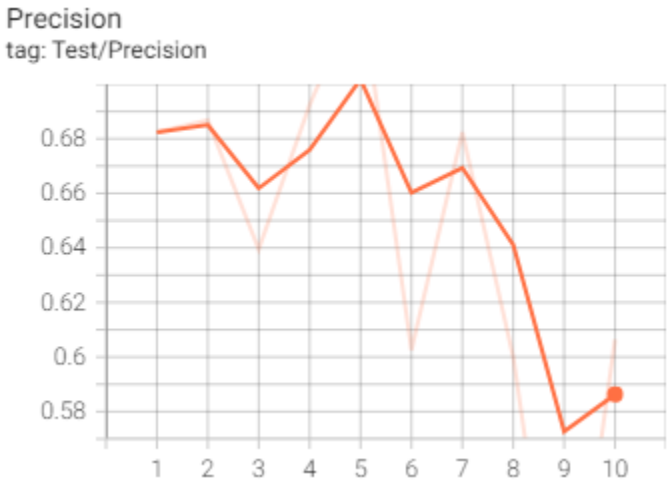

Figure 7.23 Test Precision over 10 Epochs (LeNet)

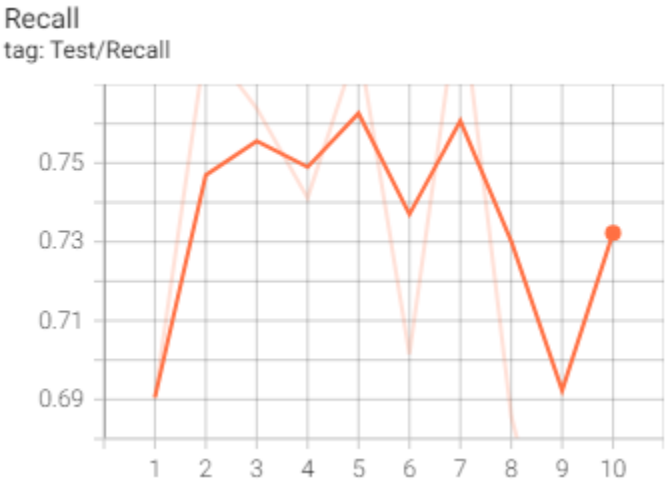

Figure 7.24 Test Recall over 10 Epochs (LeNet)

LeNet's performance was modest compared to the other models. This is because it is not very precise and has high loss due to its simplicity hence it may not be suitable for this dataset which is complex. However, the LeNet architecture used in this thesis was only sufficient to perform the basic tasks of image recognition as compared to more intricate patterns that can be found in a more complicated set of data.

7.2.5 VGGNet-16

The research also employed VGGNet16 model which is known for its depth and simplicity. Built on ImageNet, VGGNet16 works well for feature extraction because of its homogeneity among different layers. The VGGNet16 comprises 13 convolutional layers and 3 fully connected layers totaling to 16 layers with small (3x3) convolutions.

According to the results, VGGNet16 took up an average time of 65268.37 seconds which was the longest among all the models tested. The highest accuracy was at 0.7200 with an average accuracy at 0. 6324.The average training time per epoch was also substantial as 1305.36 seconds. The overall mean of precision and recall were approximately equal as well as their mean value was around one half or less from each other in case of opening any significant difference between them. The Figure 7.25, Figure 7.26, Figure 7.27 and Figure 7.28 shows a visual representation of the model's performance for train set.

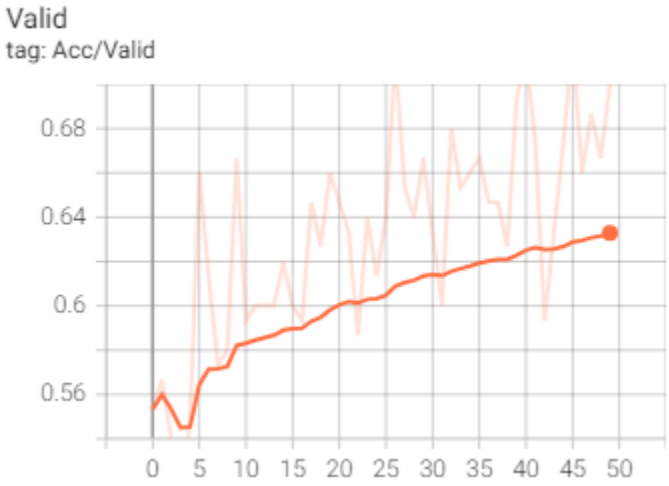

Figure 7.25 Accuracy over 50 Epochs (VGGNet)

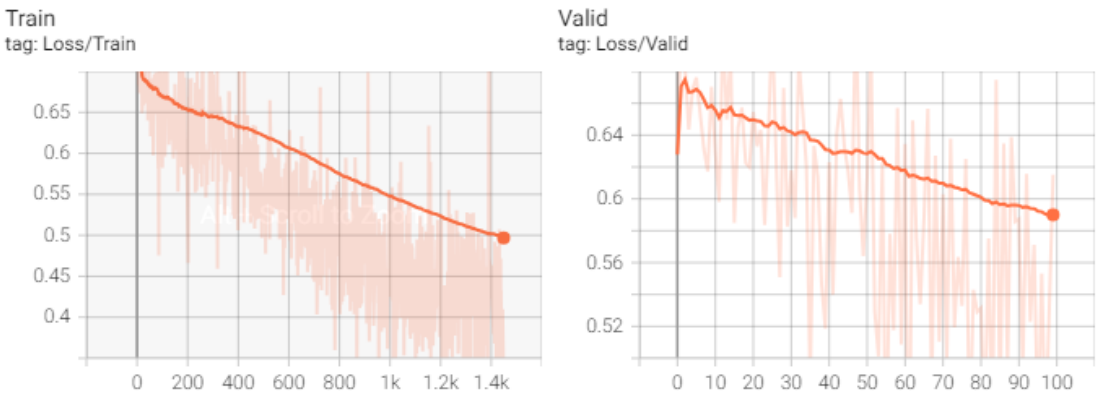

Figure 7.26 Training and Validation Loss over 50 Epochs (VGGNet)

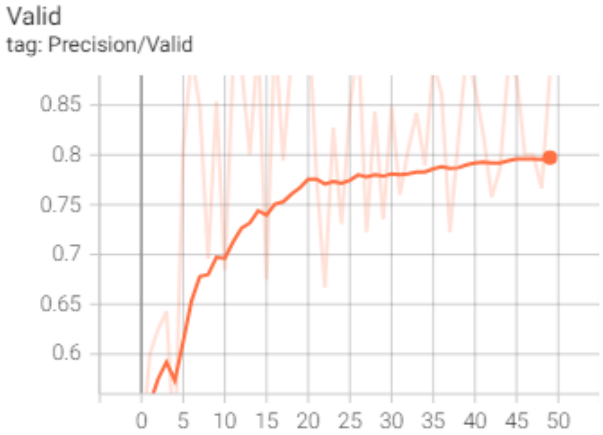

Figure 7.27 Training Precision over 50 Epochs (VGGNet)

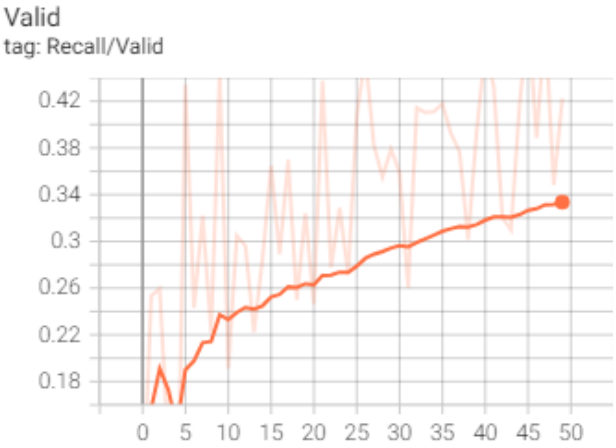

Figure 7.28 Training Recall over 50 Epochs (VGGNet)

VGGNet's test set evaluation metrics show that it has improved in performance as compared to the training set. The mean test accuracy was 0.7160 and the mean test loss was 0.5573. The precision and recall scores were 0.6549 and 0.9007 respectively. These results are indicative of test generalization on the model, with a significant increase in recall, hence identifying a number of more relevant instances now over time However, for precision, this is not so true because it has decreased indicating a tradeoff between precision and recall.In conclusion, We can say VGGNet demonstrates good performance on the test dataset which shows its potential in high recall based tasks. The Figure 7.29, Figure 7.30, Figure 7.31 and Figure 7.32 shows a visual representation of the model's performance for test set.

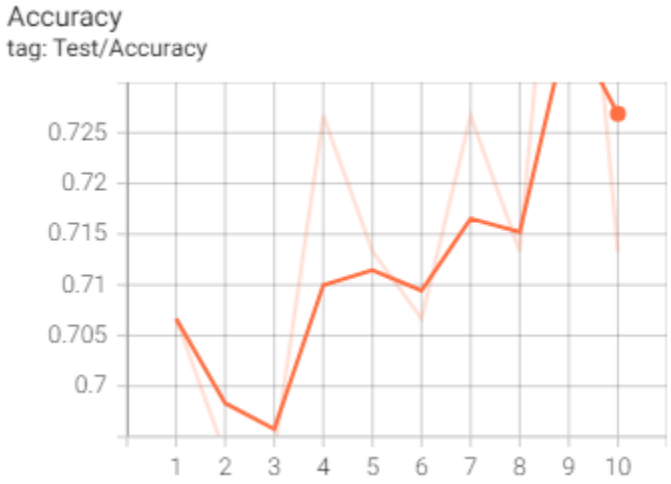

Figure 7.29 Test Accuracy over 10 Epochs (VGGNet)

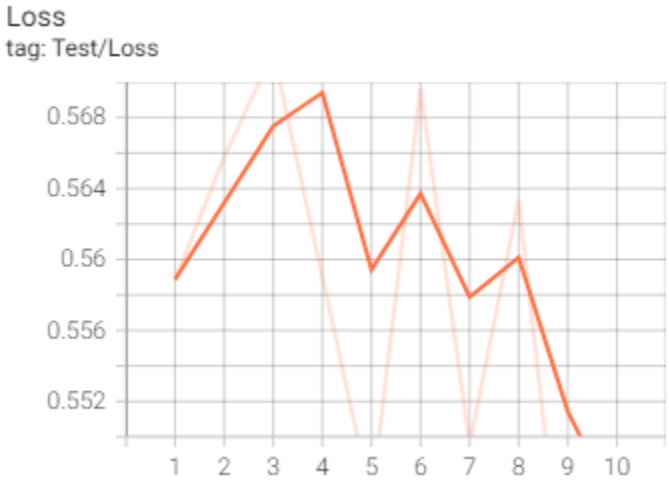

Figure 7.30 Test Loss over 10 Epochs (VGGNet)

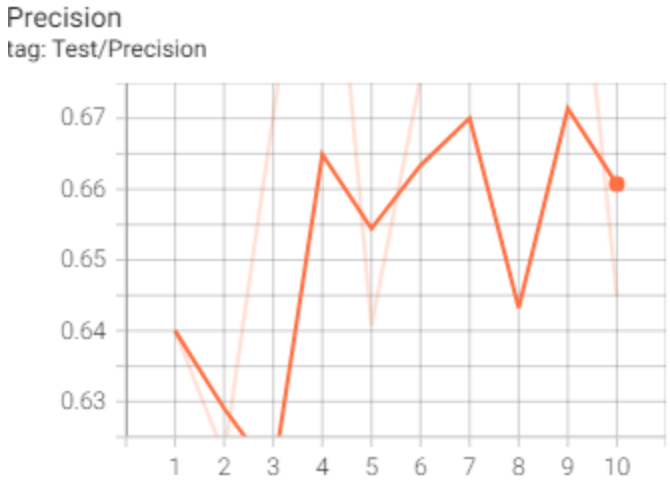

Figure 7.31 Test Precision over 10 Epochs (VGGNet)

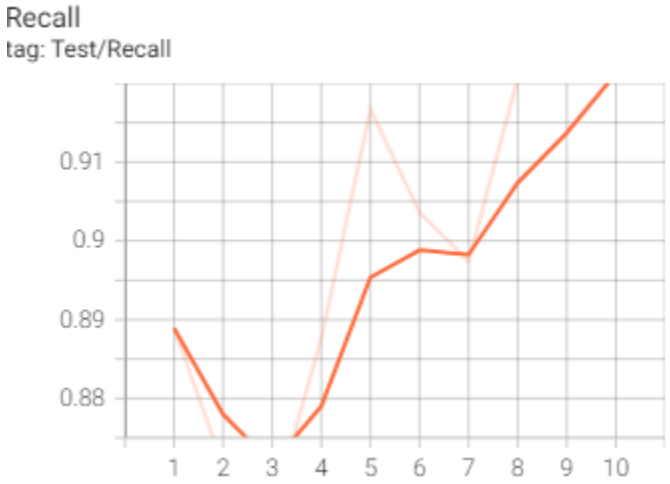

Figure 7.32 Test Recall over 10 Epochs (VGGNet)

While VGGNet16's convolutional network accuracy and precision were lower than expected, its long computation times and high requirements make it less of a practical choice for this task. The deep architecture of VGGNet16 that is good at capturing fine-grained features may have resulted in overfitting as indicated by its comparatively low validation accuracy and relatively higher loss in contrast to other models.

7.2.6 GoogleNet-V3

GoogleNet V3 which is also referred to as Inception V3 was the model under experimentation. With its complex architecture and efficiency, GoogleNet V3 was subjected to the same conditions. Inception was developed with a novel concept of “Inception modules” that can perform more efficient computations through incorporating different filter sizes within layer.

On average, GoogleNet V3 took 5154.86 seconds to process with the same average CPU time. The highest accuracy obtained was 0.7267 while the average accuracy reached 0.6333 on average across runs. The training required an epoch time of 94.70 seconds on average. On average, the loss amounted to 0.6456, precision and recall averaging at 0.6420 and 0.6667 respectively. The Figure 7.33, Figure 7.34, Figure 7.35 and Figure 7.36 shows a visual representation of the model’s performance for train set.

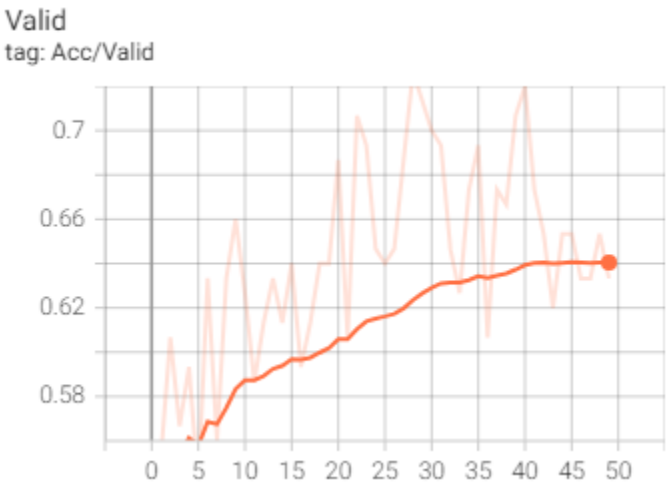

Figure 7.33 Training Accuracy over 50 Epochs (GoogleNet)

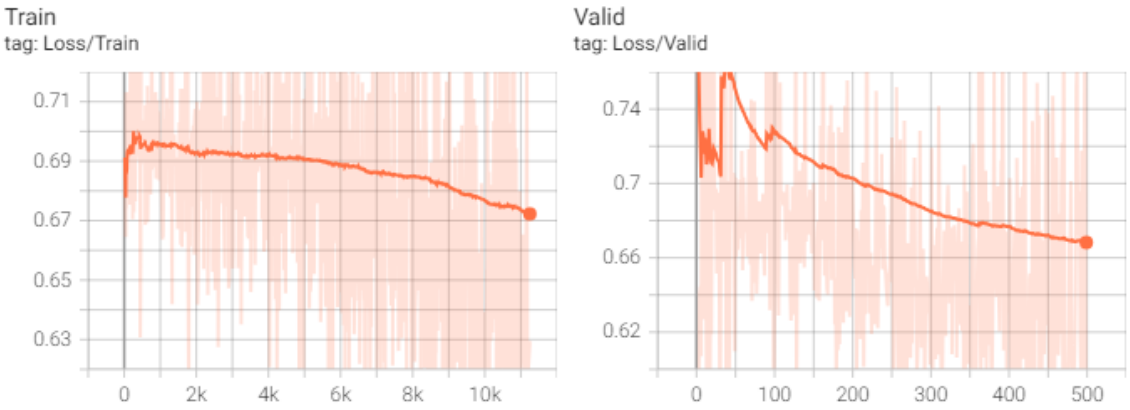

Figure 7.34 Training and Validation Loss over 50 Epochs (GoogleNet)

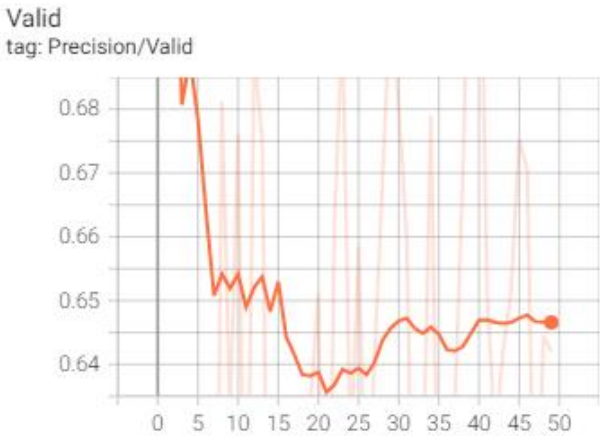

Figure 7.35 Training Precision over 50 Epochs (GoogleNet)

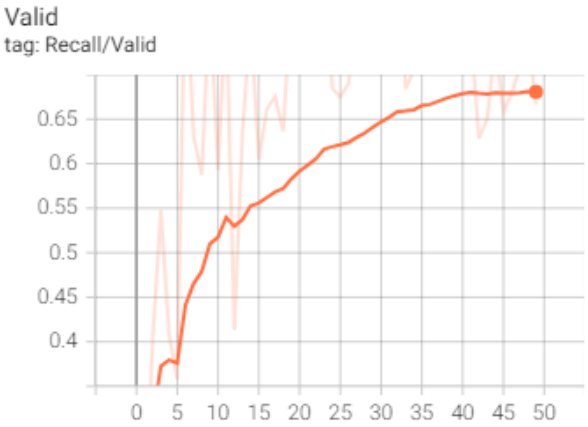

Figure 7.36 Training Recall over 50 Epochs (GoogleNet)

For the test evaluation metrics for the GoogleNet model indicate that it has performed worse than when being trained. The mean of testing accuracy was 0.6113 and testing loss had an average value of 0.6592. The precision score was 0.5992, while recall obtained a value of 0. 6393.These findings show that there is lower precision on the test dataset and imply that this model may not generalize well to new data. This suggests potential overfitting during training and emphasizes the necessity for better adjustment or more solid training data so as to improve performance of this model for unseen data. The Figure 7.37, Figure 7.38, Figure 7.39 and Figure 7.40 shows a visual representation of the model's performance for test set.

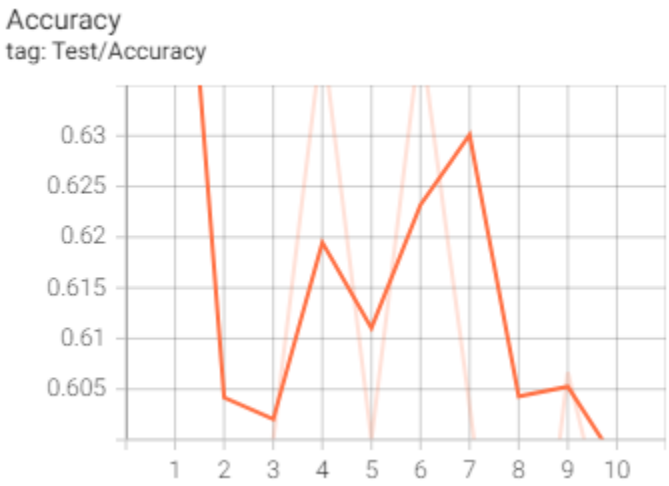

Figure 7.37 Test Accuracy over 10 Epochs (GoogleNet)

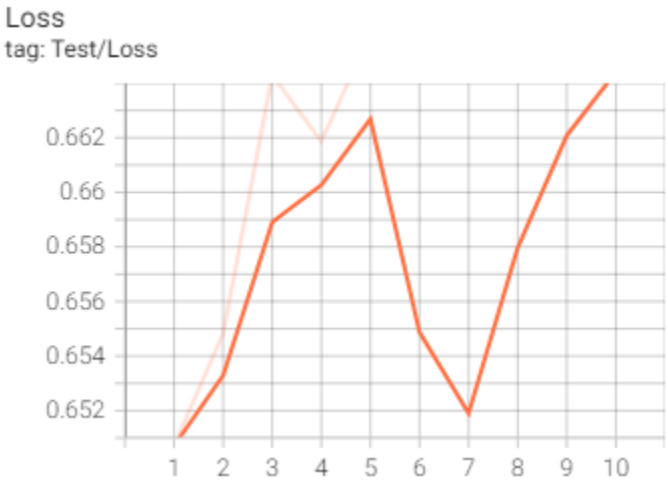

Figure 7.38 Test Loss over 10 Epochs (GoogleNet)

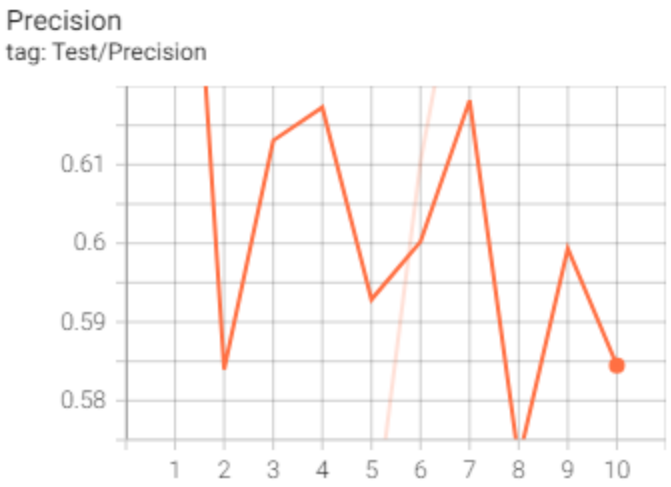

Figure 7.39 Test Precision over 10 Epochs (GoogleNet)

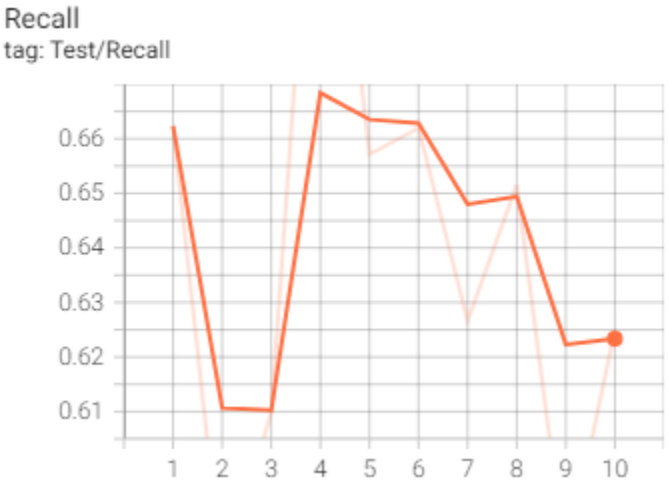

Figure 7.40 Test Recall over 10 Epochs (GoogleNet)

GoogleNet V3 had average performance in terms of accuracy and recall, but it fell short of ResNet18 or AlexNet. With a complex architecture and its efficiency, it can be one great model, but not the best for this job. The GoogleNet V3 inception modules provide a mix of computation efficiency and performance, whereas the overall accuracy and loss metrics for the model indicate that it may not be as appropriate to this dataset as ResNet18.

7.2.7 Comparison of Base Models

The Table 7.6 show the performance metrics of all tested models in 50 epochs for train set:

Model	Process Time (secs)	CPU Time (secs)	Highest Accuracy	Average Accuracy	Average Training Time (secs)	Average Loss	Average Precision	Average Recall
ResNet-18	5171.97	5171.97	0.9867	0.9800	95.51	0.1390	1.0000	0.9655
AlexNet	2781.74	2781.74	0.9267	0.8600	51.87	0.2908	0.9167	0.7746
LeNet	2513.11	2513.11	0.7267	0.6800	55.19	0.6232	0.6714	0.6528
VGGNet-16	65268.37	65268.37	0.7200	0.6324	1305.36	0.5912	0.7966	0.3327
GoogleNet-V3	5154.86	5154.86	0.7267	0.6333	94.70	0.6456	0.6420	0.6667

Table 7.6 Performance metrics of tested models for train set

The Table 7.7 show the performance metrics of all tested models in 10 epochs for test set:

Model	Average Test Accuracy	Average Test Loss	Average Test Precision	Average Test Recall
ResNet-18	0.9260	0.1845	0.9518	0.8940
AlexNet	0.8647	0.3607	0.8583	0.8739
LeNet	0.6640	0.6441	0.6400	0.7367
VGGNet-16	0.7160	0.5573	0.6549	0.9007
GoogleNet-V3	0.6113	0.6592	0.5992	0.6393

Table 7.7 Performance metrics of tested models for test set

Given this contrast, ResNet18 is found to be the best model for our project. It has the highest average accuracy and precision, but a small loss rate and reasonable training and processing times. This Siamese network architecture features a good trade-off between performance and computational efficiency in ResNet18 that makes it suitable for feature extraction. Models like VGGNet16 on the other hand, despite their depth have high computation requirements as well as long training durations that are less practical for this task. The fact that VGGNet16 has lower accuracy and higher loss indicates potential overfitting issues, so it does not fit well for the given dataset. While AlexNet performs relatively well; however, it cannot match up with ResNet18 when it comes to precision and recall scores showing its ineffectiveness in capturing complex features. LeNet together with GoogleNet V3 though efficient in some respects do not provide an equivalent level of exactness or reliability as compared to ResNet18 hence making them less viable options for this study.

In conclusion, this shows that ResNet18 is best suited for the purpose because of its positive and efficient work. It can accommodate deeper networks by way of residual connections which enables it to provide higher accuracy and precision than any other model in the context of Siamese network feature extraction. This comprehensive assessment demonstrates why choosing a proper architecture matters with respect to various tasks; in this case, ResNet18 has proven to be resistant when working with some issues in the Kaggle 30 Types of Balls dataset.

7.3 Conclusion

To conclude, after the software system and machine learning models were subjected to a rigorous testing, numerous possible problems have been discovered and addressed. The Intelligent Image Search Engine with AI-based similarity detection for a web application has been significantly improved in terms of its quality and reliability through comprehensive testing.

Unit testing, functional testing, integration testing, API testing as well as user acceptance testing were conducted during system testing to ensure that every component of the web application was critically appraised. This method helped discover and resolve several

mistakes thereby improving the strength of the system within the required specifications and meeting users' expectations. Model evaluation was done by evaluating various CNN's architecture within Siamese network framework. ResNet18 emerged as the top-performing offering highest trade-off in terms of accuracy, precision and computational efficiency.

Through conducting extensive testing in these two crucial areas, we have guaranteed that the result does not only conform to the necessary conditions and guidelines but also guarantees a dependable and excellent performance. This twofold method of testing has become an indispensable part of the development process thus resulting in creating a strong, effective, and user-friendly system.

CHAPTER 8

CONCLUSION AND RECOMMENDATION

8.1 Conclusion

This project has successfully developed an intelligent image search engine with AI-based similarity detection for web application and successfully addresses several critical problems. Firstly, the inefficiencies and inaccuracies in traditional image search engine have been resolved. This project has taken into consideration of the problem that the result retrieved not from specific place and solved by based on the database which means the result will retrieve from the specific place to who's that configure the database. With the implementation of content-based image search method, the problem of inaccuracies of keyword-based image search method and inefficiencies of textual input for image search engine also solved and make it more user-friendly that user no need think about what keywords need to use to search the image but just provide the image query and find the similar images. The AI model will handle the feature extraction of images and process the task of similarity detection then provide the most similar image to the user.

Furthermore, by using AI model in image search engine, it can improve the similarity detection efficiency for the tasks of compare the feature vector from the images in the database. The following project's objectives were achieved.

1. To design web application to facilitate user interaction and integrate the AI-based image search engine into the web application, allowing users to upload images and view search results

2. To implement a deep learning model for extracting features from uploaded images and learning the features of image content

3. To utilize AI techniques to compare the extracted features and determine the similarity between the query image and images in the database and list the results

The initial objective has been fulfilled since the development of the web application has been accomplished, enabling the user to upload image in home page through drag-and-drop method or upload from folder method. Real-time feedback through loading indicators to show the image processing process which are the upload image feature extraction, feature vector comparison and similarity detection which done by the AI model and lastly show the similar image results to the user in result page.

The second objective has been achieved by utilized the pretrained base model which is the ResNet-18 to perform the feature extraction task. The final fully connected layer of ResNet-18 had been replaced by a new layer to output 256 feature vectors to allow the system to learn and extract features from the images which is then use for comparison and similarity detection.

Finally, the third objective has been achieved by integrated a Siamese Network model for similarity detection task. Siamese Network uses pairs of images to learn and identify similarities. When in comparison phase, the model compares the feature vectors of the query image and images in the database and then computes similarity scores. After that, the similarity scores will be use which the highest similarity score will be the most similar image and list out the images in descending order in result page.

8.2 Limitations

Despite the accomplishment of the project's objectives, there remain some limitations in the project that could be enhanced.

The first limitation is the processing time for image analysis. An occasional long response time happens due to simultaneous image uploads and similarity detection through deep learning models which require a lot of computational power for their algorithms. This pause hampers the user experience greatly especially when large images are involved or there are many requests in a second. Many times, users must wait for some seconds before they can proceed with their operations, and it can be inconvenient and annoying too. The need for high computation resources becomes a bottleneck particularly in real-time situations where

immediate feedback is expected from input devices. To make the application more efficient for real-time use and improve user satisfaction, reducing the response time is necessary. One could do this by optimizing algorithms, acquiring stronger hardware or moving complex calculations to cloud-based services.

The second limitation is the limited user interface features. It has a basic functionality for image upload and result display. However, it does not have such advanced features as batch image uploads, image search history and more interactive results displays. Such features are necessary for improved user interaction and satisfaction. The current interface doesn't exploit the system's capabilities fully but can be made to be a more versatile and user friendly by adding these features, for example batch image uploads would allow users to process multiple images at once thereby saving time and effort. Incorporating a search history feature in the system will enable users to easily revisit their previous searches thus enhancing usability. Interactive result displays that provide additional details about each result and offer sorting or filtering options could also improve user experience further.

The third limitation is database scalability. Presently, a specific database is used to save and retrieve images. More so, if the stored image number increases, there may be problems with scalability of the system leading to slower search rates and an increased storage necessity. This deficiency hampers the efficient handling of large datasets by this application. Therefore, it is necessary for data base to be scalable to maintain performance and reliability as the data grows. This can involve optimizing database queries, utilizing distributed databases and implementing caching strategies to lower work on primary database among other potential solutions. Further, regular maintenance and optimization of the database structure can assist in managing larger datasets more efficiently.

8.3 Recommendation for Future Work

The following are four recommendations for future work that could be explored for this project.

8.3.1 Optimize Image Processing Algorithms

Optimization needs to be implemented in response to the limitations of processing time. This may involve algorithmic optimizations and replacing inefficient libraries from third parties on top of other measures. Moreover, hardware accelerators such as GPUs or cloud-based solutions for heavy computations can also improve performance significantly. The net effect will be to reduce response time and make it possible for the system to take care of more complicated image processing tasks faster. Improving the efficiency of the algorithm ensures smoother user experience particularly during high traffic periods or while handling large image datasets.

8.3.2 Enhance User Interface Features

As only having basic features, the current interface can be improved with more advanced ones. Such additional characteristics include bulk uploads of pictures, search history and other interactive result displays. It would save users time and enhance efficiency in case they could upload several photos at once. If a search history feature is integrated, it would bring practical benefits to users by allowing them to easily go back to their past searches. These new improvements will allow for a much richer user experience when sorting, filtering or viewing detailed information about images is finally put into place. Ultimately, these changes will make the system more flexible and user-friendly thus meeting different individual needs within its use context.

8.3.3 Improve Database Scalability

To make the database scalable and efficient, several optimizations can be carried out. Among them are improving database structure and modifying queries to run more effectively on much bigger data sets. Employing distributed databases or incorporating caching techniques can help in handling larger data sets by redistributing the load as well as reducing the time it takes to respond back. Moreover, regular maintenance and optimization of the database structure also helps in addressing scalability issues to keep it high-performing and reliable while storing more and more images. Enhancing the scalability of a database is therefore important because it ensures that many users are reached with the application as well as large image repositories are created thus making this system robust enough for dealing with increased traffic.

8.3.4 Enhance AI Model

To further improve the accuracy and efficiency of similarity detection, more advanced AI models could be explored. Better feature extraction and similarity metrics can be obtained by experimentation with Vision Transformers (ViTs) or other convolutional neural networks (CNNs). For instance, Vision Transformers have shown promise in several image processing tasks and that could make it better at detecting subtle similarities between images. Furthermore, model performance can be improved by fine-tuning techniques on domain-specific datasets which makes the models more specialized and accurate for specific use cases. To keep pace with new techniques and ensure users receive the best possible results, AI models should be updated and refined continuously.

REFERENCES

1. (No date a) *(PDF) a review: From keyword based image retrieval to ontology based image retrieval*. Available at: https://www.researchgate.net/publication/235898006_A_REVIEW_FROM_KEYWORD_BASED_IMAGE_RETRIEVAL_TO_ONTOLOGY_BASED_IMAGE_RETRIEVAL (Accessed: 04 April 2024).

2. (No date a) *(PDF) image classification approaches for segregation of plastic waste based on resin identification code*. Available at: https://www.researchgate.net/publication/358833106_Image_Classification_Approaches_for_Segregation_of_Plastic_Waste_Based_on_Resin_Identification_Code (Accessed: 04 April 2024).

3. (No date a) *(PDF) the Google Lens analyzing quality: An analysis of the possibility to use in the educational process*. Available at: https://www.researchgate.net/publication/340050262_The_Google_Lens_analyzing_quality_an_analysis_of_the_possibility_to_use_in_the_educational_process (Accessed: 04 April 2024).

4. (No date a) *(PDF) toward aircraft recognition with Convolutional Neural Networks*. Available at: https://www.researchgate.net/publication/312303454_Toward_aircraft_recognition_with_convolutional_neural_networks (Accessed: 04 April 2024).

5. (No date a) *(PDF) web image search engine evaluation*. Available at: https://www.researchgate.net/publication/236855070_Web_Image_Search_Engine_Evaluation (Accessed: 04 April 2024).

6. (No date a) *(PDF) world wide web based image search engine using text and image content features*. Available at: https://www.researchgate.net/publication/245141838_World_Wide_Web_Based_Image_Search_Engine_Using_Text_and_Image_Content_Features (Accessed: 04 April 2024).

7. (No date a) *Architecture of the resnet-18 model used in this study....* Available at: https://www.researchgate.net/figure/Architecture-of-the-ResNet-18-model-used-in-this-study_fig3_354432343 (Accessed: 04 April 2024).

8. (No date) *(PDF) scalability challenges in web search engines*. Available at: https://www.researchgate.net/publication/226351869_Scalability_Challenges_in_Web_Search_Engines (Accessed: 04 April 2024).

9. Albizu Garcia (2019). *Why Visual Search Will Be One of the Biggest Digital Marketing Trends of 2019*. [online] Social Media Today. Available at: https://www.socialmediatoday.com/news/why-visual-search-will-be-one-of-the-biggest-digital-marketing-trends-of-20/545999/.

10. *An interactive approach for filtering out junk images from Keyword-Based Google search*
11. *results* (2009). https://ieeexplore.ieee.org/abstract/document/5159448.

12. Arnold, V. (2023). *AI in Image Recognition: Benefits, Applications and Challenges*. [online] neuroflash. Available at: https://neuroflash.com/blog/ai-in-image-recognition-benefits-applications-and-challenges/#:~:text=Accuracy%20and%20Efficiency%3A%20AI%20in.
13. Author links open overlay panelAnuja khodaskar a *et al.* (2015) *Promising large scale image retrieval by using intelligent semantic binary code generation technique*, *Procedia Computer Science*. Available at: https://www.sciencedirect.com/science/article/pii/S1877050915006924?ref=pdf_download&f r=RR-2&rr=86ef5aafdf3c7568 (Accessed: 04 April 2024).

14. Bangar, S. (2022). *Resnet Architecture Explained*. [online] Medium. Available at: https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d.

15. Boesch, G. (2021). *VGG Very Deep Convolutional Networks (VGGNet) - What you need to know*. [online] viso.ai. Available at: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.

16. Chen, M. (2023). *6 Common AI Model Training Challenges*. [online] Oracle.com. Available at: https://www.oracle.com/my/artificial-intelligence/ai-model-training-challenges/ [Accessed 4 Apr. 2024].

17. CheshmehSohrabi, M. (Mozaffar) and Sadati, E.A. (2021). Performance evaluation of web search engines in image retrieval: An experimental study. *Information Development*, p.026666692110102. doi:https://doi.org/10.1177/02666669211010211.

18. *Contextual image search with keyword and image input* (2014). https://ieeexplore.ieee.org/abstract/document/7033765.

19. encord.com. (n.d.). *What is One-Shot Learning in Computer Vision*. [online] Available at: https://encord.com/blog/one-shot-learning-guide/.

20. GeeksforGeeks. (2020). *Understanding GoogLeNet Model - CNN Architecture*. [online] Available at: https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/.

21. Gerry (2023). *30 Types of Balls Updated- Image Classification*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/gpiosenka/balls-image-classification.

22. IBM (2023). *What are Convolutional Neural Networks? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/convolutional-neural-networks.

23. ieeexplore.ieee.org. (n.d.). *IntentSearch: Capturing User Intention for One-Click Internet Image Search | IEEE Journals & Magazine | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/6104063.

24. J, S.B. (2021). *A friendly Introduction to Siamese Networks*. [online] Medium. Available at: https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942.

25. kaggle.com. (n.d.). *AlexNet Architecture: A Complete Guide*. [online] Available at: https://www.kaggle.com/code/blurredmachine/alexnet-architecture-a-complete-guide.

26. Li, M. and Ma, W.-Y. (n.d.). Image Search Engine. *Encyclopedia of Multimedia*, pp.323–328. doi:https://doi.org/10.1007/0-387-30038-4_100.

27. Reynolds, A.H. (n.d.). *Anh H. Reynolds*. [online] Anh H. Reynolds. Available at: https://anhreynolds.com/blogs/vgg.html.

28. Rock (2021). *What Are Artificial Intelligence Algorithms And How Do They Work*. [online] Rock Content. Available at: https://rockcontent.com/blog/artificial-intelligence-algorithm/.

29. Varshney, P. (2020). *LeNet Architecture: A Complete Guide*. [online] kaggle.com. Available at: https://www.kaggle.com/code/blurredmachine/lenet-architecture-a-complete-guide.