

# IMAGE BASED PRODUCT SHOPPING SEARCH ENGINE

A PROJECT REPORT

*Submitted by*

BL.EN.U4AIE19002 - AKULA DHANUSH

BL.EN.U4AIE19043 – NVS PRADYUMNA

BL.EN.U4AIE19059 - SATWIK REDDY SRIPATHI

**19AIE303 – Signal and Image Processing**

*Under guidance of*

Dr. Suja P

Assistant Professor

CSE Department



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

**BANGALORE 560 035**

AMRITA VISHWA VIDYAPEETHAM  
AMRITA SCHOOL OF ENGINEERING, BANGALORE, 560035



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “**IMAGE BASED PRODUCT SHOPPING SEARCH ENGINE**” submitted by

AKULA DHANUSH BL.EN.U4AIE19002

NADELLA VENKATA SAI PRADYUMNA BL.EN.U4AIE19043

SATWIK REDDY SRIPATHI BL.EN.U4AIE19059

in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in “**Artificial Intelligence Engineering**” is a bonafide record of the work carried out under my(our) guidance and supervision at Amrita School of Engineering, Bangalore.

NAME OF SUPERVISOR

Dr. Suja P  
Assistant Professor  
Department of Computer Science

<<Signature of the Chairperson of the Department with date>>

NAME OF CHAIRPERSON-.....

This project report was evaluated by us on ..... (Date...)

INTERNAL EXAMINER

EXTERNAL EXAMINER

## ABSTRACT

In today's world there is lot of demand for user interface to facilitate users and make it easy for them in accessing something. Similarly, there is increase in demand for image-based searching and collecting information. We come across many e-commerce websites which do not support image-based product searching. So, we are developing an application for image-based product recognition and classification with Graphical User Interface where users can search for similar product by giving an input query image. Our work is mainly to help people who have dyslexia and find it difficult to read, by providing them a simple user-friendly interface to buy and shop for the products they need similar to the product they want. We worked on an image-based search engine which will identify top ten images in the database which is similar to the input query image. The user can search for images similar to the input image, and add them to cart after entering the quantity of the product required and finally buy then once the bill is generated. This project is an application developed on Image Processing. The techniques used include feature extraction, feature matching, image distance calculation for similarity finding etc. the module used to carry out this include VGG16 available in keras.

## TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION	5
CHAPTER 2 – SYSTEM ARCHITECTURE	7
1. CONVOLUTION NEURAL NETWORKS	8
2. VGG16 MODEL	9
3. IMAGE CLASSIFICATION	11
4. GRAPHICAL USER INTERFACE (GUI)	11
5. WORKING	12
CHAPTER 3 – IMPLEMENTATION	13
CHAPTER 4 – RESULTS	21
CHAPTER 5 – CONCLUSIONS AND FUTURE ASPECT	28
CHAPTER 6 – REFERENCES	29

## LIST OF FIGURES

Fig1.1 Image Operations	6
Fig2.1 System Architecture	7
Fig2.2 Simple Neural Network	8
Fig2.3 VGG-16 Model Architecture	9
Fig2.4 Layers of VGG-16 CNN model	10
Fig2.5 Feature extraction	11
Fig3.1 Model Summary	14
Fig4.1. Login Window	21
Fig4.2 Login Window with credentials	21
Fig4.3 Product Query Window	22
Fig4.4 File explorer	22
Fig4.5 Product query window after loading product	23
Fig4.6 Product Display window	23
Fig4.7 Product Display Window	24
Fig4.8 Product Display Window	23
Fig4.9 Product Query Window	25
Fig4.10 Product Query Window	25
Fig4.11 Dataset	26
Fig4.12 Dataset Description	27

## CHAPTER 1: INTRODUCTION

Now a days people prefer something which is easy to understand and is easy to use. With increase in demand for an application which is user friendly and easy to use, we have developed an application which includes user interface and makes it easy for every user. This application is mainly to address the people with dyslexia and help them use this application with ease. We worked on an image-based product recognition system with which one can find products similar to the image of product they want. This way any user can simply take picture of the product they want and find all related and similar products to it.

In order to implement this, the input query image is used as a template and other images similar to this query image are found in database which are nothing but products. So, the similar looking product based on shape, color, size, type etc. are found and returned to the user. The user can now choose from the available products and buy them accordingly.

The key concepts included in this project include Image Processing, image feature extraction, image distance calculation for identifying similarity, user interface etc.

Let's start by briefly understanding what Image Processing is and how important role it plays in today's world.

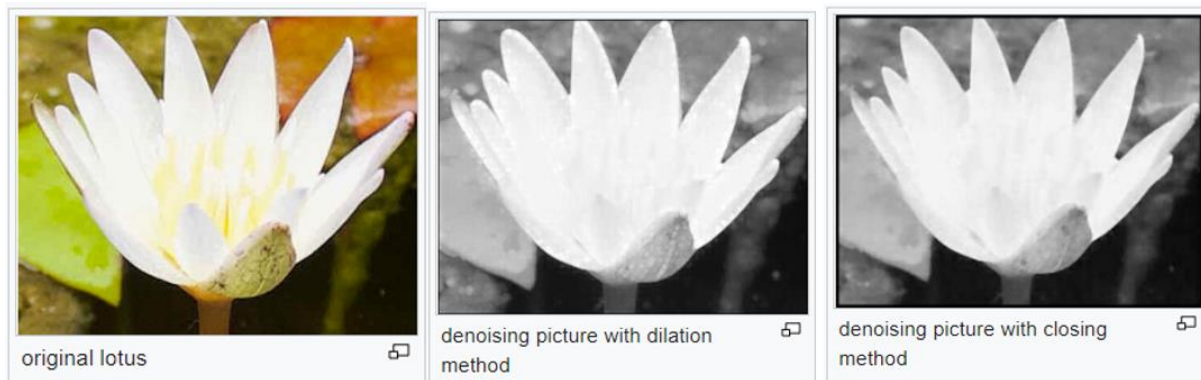
The field of signal and image processing encompasses the theory and practice of algorithms and hardware that convert signals produced by artificial or natural means into a form useful for a specific purpose. The signals might be speech, audio, images, video, sensor data, telemetry, electrocardiograms, or seismic data, among others; possible purposes include transmission, display, storage, interpretation, classification, segmentation, or diagnosis. Signal and image processing engineers design and implement techniques to process and analyze signals and images to perform useful operations such as signal enhancement, signal filtering, data compression, feature extraction, image and video understanding, and data mining.

Signal and image processing algorithms lie at the heart of almost all of today's digital technology, from the mobile phone to advanced satellite imaging. Fields such as medical imaging & monitoring, Remote Sensing & environmental monitoring, Consumer electronics, Industrial condition monitoring, Defense and Homeland Security, Robotics and Autonomous vehicles etc. adopt these processing techniques to carry out any process.

Let us understand what image processing is.

### 1.1. IMAGE PROCESSING

Image Processing is a sub category of Signal Processing. Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. There are two methods of processing, analogue and digital image processing. Image enhancement, image reconstruction, image recognition, image restoration, image feature extraction etc. are different techniques which Image Processing can do. Operation such as dilation, erosion, opening closing, filtering etc. can also be done.



**Fig1.1 Image operations**

The above image shows how the techniques can be applied on images and can be made better according to our requirement.

In this project we are implementing Image Processing Techniques. An input image is given and the task is to identify similar images present in our database. The top ten most similar images are taken from the database and are shown as output. To implement this, let us briefly know what image features are, how are they extracted and how is distance between two images calculated.

## **1.2. IMAGE FEATURE EXTRACTION AND DISTANCE CALCULATION**

Every image has so much information in it. Of all the information it has, there might be information which is sufficient enough to classify that image. This necessary and sufficient information is nothing but a feature. So, feature extraction is basically to collect necessary information from an image. In this project, features of all the images in the database are collected and stored. Features in an image can be based on shape, size, color and many more. Each feature describes each factor present in that image. When the input query image is given, its features are also calculated on the same lines. Now, how similar two images are or how distinct two images are is calculated but the distance between those two images. The more the distance is, the less is the similarity. So, the distance between query image and all images present in the dataset is found and the top ten images which have least distance are found and given back as output.

This process is carried out using VGG16 model where the features are extracted and represented as a vector. This vector is then subjected to convolution and pooling to get an optimal output. Principal Component Analysis (PCA) is also incorporated to take top few features only from all images.

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. VGG16 is developed based on CNN. It is a Keras model using for several image processing. It is developed by training ImageNet dataset which is a vast dataset of 14million images. It has a 92.7 accuracy for the top 5 categorization consisting of 16 layers.

In the following sections, more details on working of the modules used is described.

## CHAPTER 2: SYSTEM ARCHITECTURE:

In this chapter, details of system architecture followed for this project and modules included in the architecture are briefed. This project revolves around Image classification, which is one of the main applications of Image processing. Following modules and their inter-connections is discussed:

1. CNN – VGG16 Model
2. Image Classification
3. GUI
4. Working

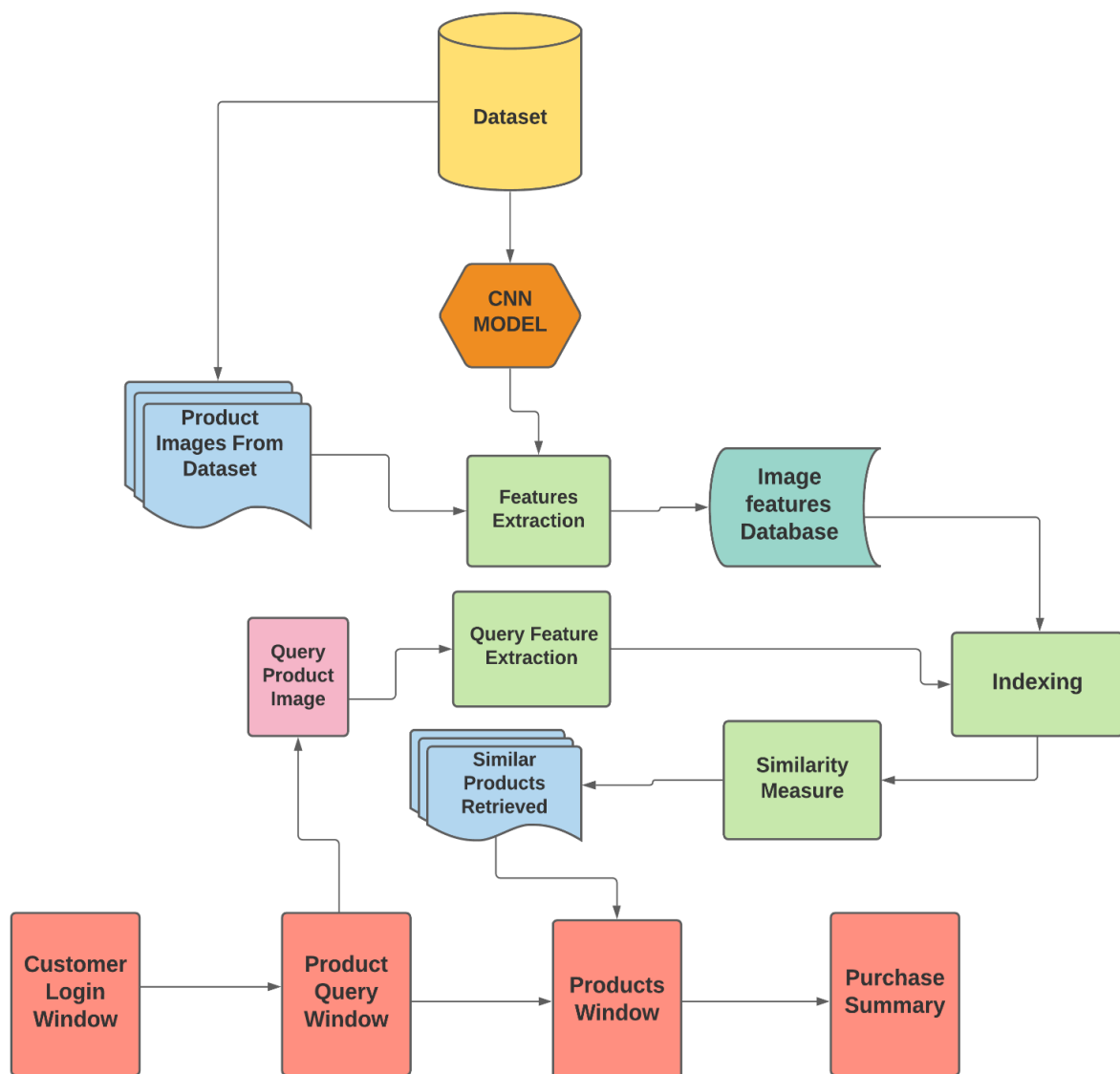
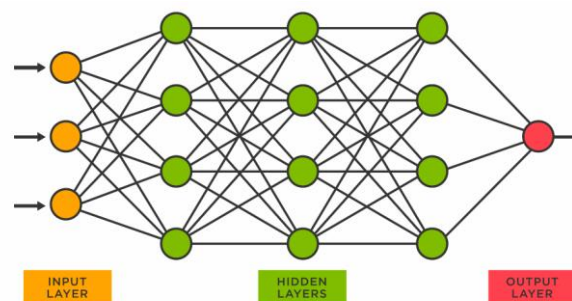


Fig2.1. System Architecture



## 2.1. CONVOLUTIONAL NEURAL NETWORKS (CNN):

Neural network is series of algorithms and tasks which tries to achieve the relation between the sets in data. These networks endeavor tasks by mimic process, just like humans learning from training and evolving and thus are also called Artificial neural networks and play a significant role in the field of Artificial Intelligence. These neural networks are composed of nodes or neurons, which are inspired from biological neural network and designed, where each network is made up of artificial nodes or neurons for solving artificial problems. These artificial networks are evolved based on self-learning. The main aim of this field of neural network is to enable machines to see world as humans and conceive it similar manner and use the knowledge to solve complex problems which are sometimes not even possible by humans. These neural networks have wide range of applications like function approximation, or regression analysis, including time series prediction and modelling, classification, including pattern and sequence recognition, novelty detection and sequential decision making, Data processing, including filtering, clustering, blind signal separation and compression.



**Fig2.2. Simple Neural Network**

Convolution neural networks (CNN) are one such artificial networks which predominantly assist Image processing, where these networks through deep learning algorithms inculcate the nature of self-learning as humans. CNN plays a key role in advancement of the computer vision field by performing tasks such as Image and video recognition, Image classification and analysis, Feature extraction, recommendation system and many other vision related tasks. In CNN, key function is convolution, in terms, it involves a linear operation of filtering, i.e., technique of multiplying the input data with a set of weights which are also called as filters or kernels. These filters are often small compared to the actual data on which CNN is working and the multiplication will be element wise multiplication of filter-sized patch of input data and the result holds information of both input data and the weights or data composed in networks. Thus, it involves in working with small filters than input which allows in multiplying same weights with input data several times at different points and systematically covers the patch of input data from left to right and top to bottom. This powerful idea of multiplying same set of weights with input data several times results in extracting detailed information or features of input data, and this can help in wide range of applications under Signal and Image processing. There are wide range of Convolution neural networks which are used as models, and these models are unique and vary from each other based on their network architecture which covers type of layers or filters and their working, their principle functioning, applications in which they are used, performance and data on which it works on for self-learning to evolve into best CNN models.

## 2.2. VGG16 Model:

VGG16 is convolutional neural network model which is designed and proposed in 2014, by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. This model is designed as a part of ImageNet challenge. ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is a program where groups come together with software programs to compete for ImageNet challenge, where challenge is to design a model which performs object detection and image classification, and endeavoured accuracy of 92.7% top-5 test accuracy in ImageNet, and won 1<sup>st</sup> and 2<sup>nd</sup> prize in object recognition and image classification. ImageNet is a huge dataset with over 14 million high resolution images and roughly belong to wide range of 20,000 classes or categories. These images are collected from several sources and are labelled by human labellers using tools like Amazon’s Mechanical Turk crowd-sourcing tool. This model is trained for several weeks using NVIDIA titan Black GPU’s. And the name VGG-16 came from the organization under which both K. Simonyan and A. Zisserman have worked for this model “Visual Geometry Group”, and thus VGG-16 is used as an object detection and image classification algorithm which is able to classify 1000 images from 1000 different classes with top 5 error rate of 7.3% and more accuracy of 92.7% and its easy transfer learning feature made this model popular for computing tasks in computer vision field.

### 2.2.a. Model Architecture:

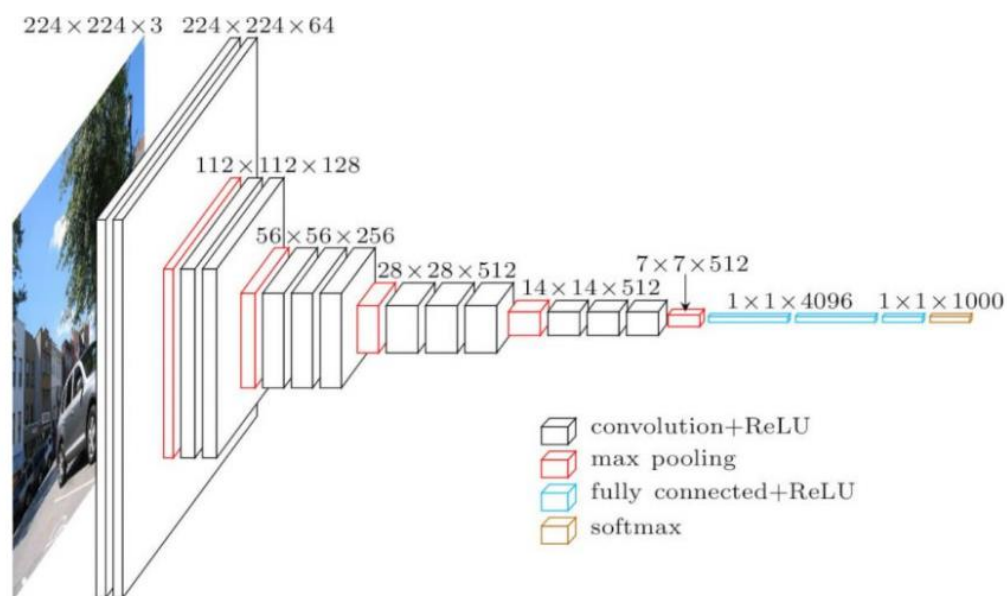


Figure 1. VGG16 Architecture

Fig2.3. VGG-16 Model Architecture

The VGG-16 as a convolutional neural network uses 16 convolutional layers which are kernels or filters, each with unique configuration and size to get the features and details of input data. During training, input to this model or Convnet is fixed  $224 \times 224 \times 3$ , where any input image size is cut down or normalized to  $224 \times 224$  size with 3 channels (RGB). After input is normalized, it is pre-processed by subtracting mean red, green and blue values from RGB channels, to prevent the effect of illumination, distortion and other discrepancies of the image on training of the model. This model is comprised of 13 convolutional layers with small  $3 \times 3$  filters and have five max pooling layers and three dense layers, adding up to 21 layers in which

only 16 layers are weight layers which are learnable parameter layers. Once image is ready after pre-processing for training, it is first passed to stack of convolutional layers which have small 3x3 receptive filters as discussed, and this is the most unique part of the design of the model, as they have used these number of layers with small 3x3 filters with stride of 1 and same padding which are deeper with more non-linearities and less parameters, instead of using layers with hyper parameters. These layers are followed by max-pooling layers with 2x2 filters and stride of 2. All these convolution layers and max-pooling layers are arranged consistently throughout the architecture of the model. The number of filters each convolution layer has increases consistently, where 1<sup>st</sup> two convolutional layers has 64 filters and then followed by a max pooling layer, 2<sup>nd</sup> two convolutional layers has 128 filters and then followed by a pooling layer, 3<sup>rd</sup> three convolutional layers has 256 filters and followed by a pooling layer, and 4<sup>th</sup> and 5<sup>th</sup> three convolutional layers have 512 filters. All these convolutional layers and max pooling layers are followed by three fully connected hidden layers, in which first two are comprised of 4096 neurons and last layer is comprised of 1000 neurons.

### 2.2.b. Model Working:

When image of size 224x224 with RGB 3 channel is passed to first stack of 2 convolutional layers with 64 filters each, the convolution stride is fixed at 1 pixel and padding is 1 pixel. This aids in storing spatial resolution and this results in size of the output activation map is same as the input dimensions. The activation maps are then passed through spatial max pooling over a 2 x 2-pixel window, with a stride of 2 pixels. This halves the size of the activations. Thus, the size of the activations at the end of the first stack is 112 x 112 x 64. The activations then flow through a similar second stack, but with 128 filters as against 64 in the first one. Consequently, the size after the second stack becomes 56 x 56 x 128. This is followed by the third stack with three convolutional layers and a max pool layer. The no. of filters applied here are 256, making the output size of the stack 28 x 28 x 256. This is followed by two stacks of three convolutional layers, with each containing 512 filters. The output at the end of both these stacks will be 7 x 7 x 512. So, the dimension of these activations keeps on reducing after convolution with every stack of layers, results to get converge into extracting out main features or aspects of the input data. The stacks of convolutional layers are followed by three fully connected layers with a flattening layer in-between. The first two have 4,096 neurons each, and the last fully connected layer serves as the output layer and has 1,000 neurons corresponding to the 1,000 possible classes for the ImageNet dataset. The output layer is followed by the SoftMax activation layer used for categorical classification.

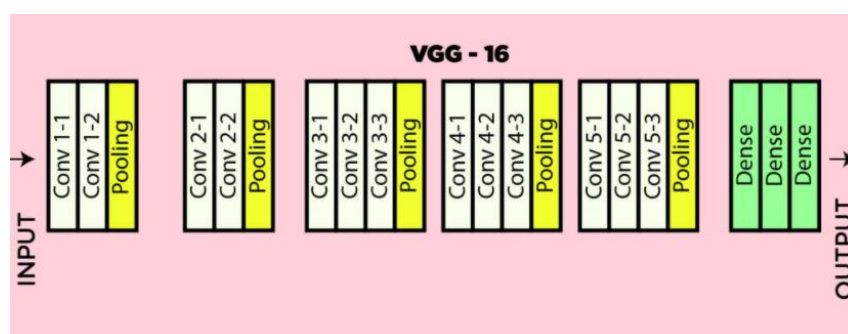


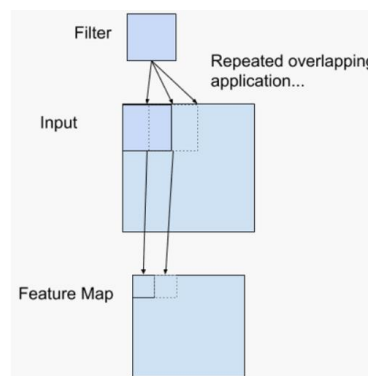
Fig2.4. Layers of VGG-16 CNN model

### 2.3. IMAGE CLASSIFICATION:

Image classification is a complex task of categorizing images and labelling them to a specific class to which they are belonged to. These criteria of image classification help in many tasks related to computer vision such as Pattern matching, pattern recognition, image and object recognition, and many more. This is achieved using deep learning, in which a neural network model is trained to perform tasks such as feature extraction and feature comparison to predict or to classify images. In this project with the help of VGG-16 model, image classification is achieved and Image based product shopping Search engine application is made. To achieve image classification using any deep learning model following tasks are to be performed given an input image:

1. Image is pre-processed
2. Feature Extraction of Image
3. Comparing Features of the image with features stored in the model, in the form of weights.
4. Finally, predicting the class of image.
5. Similar images are also retrieved by performing similarity measure.

Image is pre-processed using mean subtraction and other techniques to remove distortions and illuminations from the image. Features are details or important aspects of image or data of the image without which image can't be depicted or presented. As discussed in the working of VGG-16 model, features of the images are extracted and are used for prediction. But in this project instead of predicting the class of the image, similar images from the database are retrieved with the help of similarity measure, where features of each image in the dataset are compared with query image and images with most similar features are retrieved.



**Fig2.5. Feature extraction.**

### 2.4 GUI:

To make application interactive and user-friendly Graphical User Interface (GUI). In this project GUI is built with the help of Pyqt5 designer. Qwidgets are used which help in taking input and displaying output and selecting action, where LineEdit widget are used as input widgets, labels come under output widgets and push buttons are used to select an action.

## 2.5. WORKING:

This project is about an application for shoppers to search products based on image of a product rather than tags or names. A dataset is made with 183 high resolution images which are products. All these product images cover furniture like chairs, tables, wardrobes, beds and clothing like shirts, t-shirts, shorts, and tops. And a excel file is made where all the details and attributes like product name, type of the product, color of the product, brand name to which product belongs to, rating of the product, and price of the product. In this project four windows are designed which are login form, product query image, product window and purchase summary.

First, user is logged in using a username and a password. And in second window image of query product is loaded and this image becomes input image to CNN model. In this project Vgg-16 model is taken and a custom model is made, where last layer of Vgg-16 which is used to categorize the input image to a class in 1000 classes of ImageNet dataset. So, the input layer of the custom CNN model is same input layer which is of VGG-16 model and the output layer of the custom CNN model is last fc2 dense layer, where features of all the images in the dataset are acquired. Now as shown in system architecture, all the images of the products are taken and with the help of custom model all the features are extracted and preserved. And an indexing is made according to the list of products and according to product index, product's features are stored as a space of features. Alone, these activations or features of dimension nearly 4096 provide a good representation, but it is a good idea to do one more step before using these as our feature vectors, which is to do a principal component analysis (PCA).

PCA is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. We apply PCA for because the 4096-bit feature vector may have some redundancy in it, such that multiple elements in the vector are highly correlated or similar. This would skew similarity comparisons towards those over-represented features. And other reason is, operating over 4096 elements is inefficient both in terms of space/memory requirements and processor speed, and it would be better for us if we can reduce the length of these vectors but maintain the same effective representation. PCA allows us to do this by reducing the dimensionality down of the feature vectors from 4096 to much less, but maintain a representation which is still faithful to the original data, by preserving the relative inter-point distance. Thus, PCA reduces the amount of redundancy in our features (from duplicate or highly-correlated features), speeds up computation over them, and reduces the amount of memory they take up.

Next, query product's image is taken from the second window and features of this image are extracted. And these new features are fit into features vector space, which is made after the performing PCA. Next, these new features are compared with features of each image stored in the space. This comparison is made by calculating the distance of similarity between the features of the query image and features of other images in the dataset. Top images with less similarity measure is retrieved which are nothing but products which are similar to query image product loaded in second window. Now these retrieved images of products are displayed in third window named products window. Now from this window, user can add products to cart and also given an option to choose quantity. And in the next and last window, purchase or bill of the user is displayed in the form of table which includes products he added to the cart, type of products, colour and brand of the products. And according to the quantities selected and individual prices of the products total bill is shown.

## CHAPTER 3: IMPLEMENTATION

In this chapter, the details of code is discussed and explained.

```
#import libraries
import os
import keras
from keras.preprocessing import image
from keras.applications.imagenet_utils import decode_predictions, preprocess_input
from keras.applications import vgg16
from keras.models import Model
import numpy as np
import matplotlib.pyplot as plt
```

Necessary python libraries are imported. In this Keras is imported which is a package related to deep learning and has applications and CNN models. For our project to vgg-16 model is imported from this Keras module. And from Keras module Preprocess\_input is imported for pre-processing of image and taking the image into array for further processing. And Numpy is also imported for array related computations. Matplotlib is also imported for plotting.

```
#model
model = keras.applications.vgg16.VGG16(weights='imagenet', include_top=True)
```

Here, from Keras module VGG16 model is created with the name of model and with weights as ImageNet dataset to get more accuracy.

```
def load_image(path):
    img = image.load_img(path, target_size=model.input_shape[1:3])
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return img, x
```

This is a function named load\_image, which helps to transform or normalize the image to the standard size of 224x224x3 when path of the image is provided. And in this function, we are using preprocess\_input function imported from Keras module to remove distortions and illuminations of the image. Thus, this function returns clean image with standard size which any input image for CNN should be.

```
#Model for feature extraction
feat_extractor = Model(inputs=model.input, outputs=model.get_layer("fc2").output)
```

What we have in the model variable is a highly effective image classifier trained on the ImageNet database. We expect that the classifier must form a very effective representation of the image in order to be able to classify it with such high accuracy. We can use this to our advantage by re-purposing this for another task.

What we do is we copy the model, but remove the last layer (the classification layer), so that the final layer of the new network, called `feat_extractor` is the second 4096-neuron fully-connected layer, "fc2 (Dense)".

The way we do this is by instantiating a new model called `feat_extractor` which takes a reference to the desired input and output layers in our VGG16 model. Thus, `feat_extractor`'s output is the layer just before the classification, the last 4096-neuron fully connected layer. It looks like a copy, but internally, all Keras is doing is making a pointer to each of these layers and not actually copying anything. Thus, the output "prediction" from `feat_extractor` will just be the layer `fc2` from model.

If we run the `summary()` function again, we see that the architecture of `feat_extractor` is identical to the original model, except the last layer has been removed. We also know that not just the architecture is the same, but the two have the same weights as well. Below given is the summary of our model.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
Total params: 134,260,544		
Trainable params: 134,260,544		
Non-trainable params: 0		

**Fig3.1. Model Summary**



```
#recursively crawl the folder specified by image_path looking for images of the extensions inside of image_extensions
#and then limiting them to a random subset of maximum max_num_images images.

images_path = 'dataset'
image_extensions = ['.jpg', '.png', '.jpeg'] # case-insensitive (upper/lower doesn't matter)
max_num_images = 1000

images = [os.path.join(dp, f) for dp, dn, filenames in os.walk(images_path) for f in filenames if os.path.splitext(f)[1].lower() in image_extensions]
if max_num_images < len(images):
    images = [images[i] for i in sorted(random.sample(xrange(len(images)), max_num_images))]

print("keeping %d images to analyze" % len(images))
```

In this code, we are making ready the images dataset for feature extraction. In this above snippet, images which are in the format .jpg, .png, .jpeg are considered and all the paths of images are stored in a variable named image.

```
import time
tic = time.perf_counter()

features = []
for i, image_path in enumerate(images):
    if i % 500 == 0:
        toc = time.perf_counter()
        elap = toc-tic;
        tic = time.perf_counter()
    # print("analyzing image %d / %d. Time: %4.4f seconds." % (i, len(images), elap))
    img, x = load_image(image_path);
    feat = feat_extractor.predict(x)[0]
    features.append(feat)

print('finished extracting features for %d images' % len(images))
```

Here in this code, a loop is made to take every image from the dataset and extract its features and store it in variable named features. Here time counters are used just to understand how much time does model take to extract all the features from every image.

```
from sklearn.decomposition import PCA
features = np.array(features)
pca = PCA(n_components=30)
pca.fit(features)
```

In this above code, PCA module is imported for lowering the dimensions of the features. This is done by importing it from Sklearn package. Here a features space is made with dimension as 30 and all the features extracted before are fit into pca object, which is a space using pca.fit function. This results in dimension reduction without any loss of information.

```
pca_features = pca.transform(features)
```

The pca object stores the actual transformation matrix which was fit in the previous cell. We can now use it to transform any original feature vector (of length 4096) into a reduced 300-dimensional feature vector in the principal component space found by the PCA.



```

from scipy.spatial import distance
def get_closest_images(query_image_idx, num_results=10):
    distances = [ distance.cosine(pca_features[query_image_idx], feat) for feat in pca_features ]
    idx_closest = sorted(range(len(distances)), key=lambda k: distances[k])[1:num_results+1]
    return idx_closest

def get_concatenated_images(indexes, thumb_height):
    thumbs = []
    paths = []
    for idx in indexes:
        img = image.load_img(images[idx])
        img = img.resize((int(img.width * thumb_height / img.height), thumb_height))
        paths.append(images[idx])
        thumbs.append(img)
    concat_image = np.concatenate([np.asarray(t) for t in thumbs], axis=1)
    return concat_image, paths

```

Here, in this code distance function is imported from package named scipy, which helps us in calculating the distance between features of different images. The list similar\_idx contains the image's similarity to every other one. We can sort that list and find the indexes of the most similar images. The next cell will sort them, and then find the most similar items, and return the indexes 5 most similar images. Notice we take from indexes 1:6 rather than 0:5 because the most similar image to the query image, will trivially be the query image itself, since it is included in the distance calculation. So we just skip it. And thus, function get\_closest\_images takes index of the image and returns indexes of closest images. And the function get\_concatenated\_images takes input as indexes of the images and returns paths of those images which can be used further.

## LOGIN WINDOW

```

: from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QDialog, QWidget
from PyQt5.QtGui import QPixmap

class Ui_login_Form(object):

    def openSearchwindow(self):
        user_id = self.username_lineEdit.text()
        password = self.password_lineEdit.text()

        users = ['Dhanush', 'Pradyumna', 'Satwik']
        passwords = ['AIE19002', 'AIE9043', 'AIE19059']

        if user_id == users[0] and passwords[0] or users[1] and passwords[1] or users[2] and passwords[2]:
            self.window = QtWidgets.QMainWindow()
            self.ui = Ui_SearchWindow()
            self.ui.setupUi(self.window)
            self.window.show()
        else:
            self.caution_label.setText("Invalid Credentials")

```

The above code is a part of code which is retrieved using ui files which are designed to generate GUI. This above code is part of Login Window, where it will limit certain users to access. All the configurations of widgets like labels, lineEdits, pushbuttons comprises of the code.

## Search Window

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QWidget
from PyQt5.QtGui import QPixmap

class Ui_SearchWindow(object):

    def setupUi(self, SearchWindow):
        SearchWindow.setObjectName("SearchWindow")
        SearchWindow.resize(865, 711)
        self.centralwidget = QtWidgets.QWidget(SearchWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.img_label = QtWidgets.QLabel(self.centralwidget)
        self.img_label.setGeometry(QtCore.QRect(10, 10, 431, 661))
        self.img_label.setStyleSheet("border-image: url(shopping.png);\n"
                                     "border-top-left-radius :50px;")
        self.img_label.setText("")
        self.img_label.setObjectName("img_label")

        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(440, 10, 401, 651))
        self.label_2.setStyleSheet("background-color: rgb(255, 255, 255);\n"
                                     "border-bottom-right-radius :50px;")
        self.label_2.setText("")
        self.label_2.setObjectName("label_2")
```

Similarly, a part of code for Search Window GUI, which is product query window.

```
def loadimage(self):
    file_name = QtWidgets.QFileDialog.getOpenFileName(None, 'Open file', 'F:\\', "Image files (*.jpg *.png *.jpeg)")
    self.pixmap = QPixmap(file_name[0])
    self.product_display_label.setPixmap(self.pixmap)
    f = open("query_imagepath.txt", "w")
    f.write(str(file_name[0]))
    f.close()

    with open('query_imagepath.txt') as f:
        contents = f.read()

    new_image, x = load_image(str(contents))
    new_features = feat_extractor.predict(x)

    # project it into pca space
    new_pca_features = pca.transform(new_features)[0]

    # calculate its distance to all the other images pca feature vectors
    distances = [ distance.cosine(new_pca_features, feat) for feat in pca_features ]
    idx_closest = sorted(range(len(distances)), key=lambda k: distances[k])[0:10]
    results_image, paths_final = get_concatenated_images(idx_closest, 200)

    textfile = open("temp.txt", "w")
    for ele in paths_final:
        textfile.write(ele+"\n")
    textfile.close()
```

This loadimage function is an important function and is a part of search window, where this function is an action to a button. This function is comprised of all the steps or lines of code needed to get similar images. And in this project text files are used to aid the flow of the data between window to window. Here after loading the query image it's path is written to a file named query\_imagepath and as well as set that query image to display label. And again this path is taken to retrieve similar images and after getting the paths of similar images, all the paths are again stored or written in text file named temp.txt for further purposes.

Next comes important aspects of product display window.

```
import pandas as pd
fileName = 'Images Dataset Product Description.xlsx'
data_only = pd.read_excel(fileName, index_col=False)
data = pd.DataFrame(data_only)
data.drop(data.columns[[0,2]], axis=1, inplace=True)
```

Here pandas is imported to read the attributes of the products which are saved in a xlsx format file. And a dataframe is made with the data in the file.

```
def getLabelData(queryIMAGE):
    fileName = 'Images Dataset Product Description.xlsx'
    data_only = pd.read_excel(fileName, index_col=False)
    data = pd.DataFrame(data_only)
    data.drop(data.columns[[0,2]], axis=1, inplace=True)

    columns = ['PATH : ', 'PRODUCT : ', 'COLOR : ', 'TYPE : ', 'BRAND : ', 'RATING : ', 'PRICE : ']
    datalist = data.values.tolist()
    ansList = []
    outString = ''
    for rowList in datalist:
        if queryIMAGE in rowList:
            ansList = rowList

    return ansList
```

Here a function named getLabelData is written, in which if we fed the image path, attributes related to that image are collected from the dataframe created before. This helps us in showing the details of the product beside the product in GUI window.

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
from PyQt5.QtCore import *

class Ui_ProductWindow(QtWidgets.QWidget):

    global items_list
    items_list = []
    def opendialog(self):
        sender_button = self.sender()
        product_index = str(sender_button.objectName()).split('_')[1]
        quantities = ['1','2','3','4','5']
        quantity, response = QtWidgets.QInputDialog.getItem(self, 'Add to cart', 'Quantity', quantities)
        if response == False:
            pass
        else:
            items_list.append([product_index, quantity])
        f = open("cartdetails.txt", "w")
        for l in items_list:
            for ele in l:
                f.write(str(ele)+'\t')
            f.write('\n')
        f.close()
```

This is some part of code for product display GUI, and opendialog is a function which helps to choose quantity of the product for the user and also writes user activity of adding product to the cart and how many portions of product he added to a text file named cartdetails.txt.

```

import pandas as pd
global cartDFFinal
def readProductDetails(cartDetailsFile, finalDetailsFile):
    cartdetailsLines = []
    with open(cartDetailsFile, 'r') as file:
        cartdetailsLines = file.read()
    cartdetailsLineStrings = cartdetailsLines.split('\n')
    cartdetailsProductStrings = []
    for strDetails in cartdetailsLineStrings:
        cartdetailsProductStrings.append(strDetails.split('\t'))
    cartdetailsProductStrings.pop()
    cartdetails = []
    for st in cartdetailsProductStrings:
        cartdetails.append(list(st[0:2]))
    finals = []
    with open(finalDetailsFile, 'r') as file:
        for row in file.readlines():
            row = row.split('\n')
            finals.append(row[1][:-1])
    return cartdetails, finals

```

In the above code to display the details or purchase summary from the text file which we preserved as a user activity is retrieved.

```

def getBillDetails(cartDetailsFile, finalDetailsFile):
    cartDETAILS, finalsLIST = readProductDetails(cartDetailsFile, finalDetailsFile)
    fileName = 'Images Dataset Product Description.xlsx'
    data_only = pd.read_excel(fileName, index_col=False)
    data = pd.DataFrame(data_only)
    data.drop(data.columns[[0]], axis=1, inplace=True)
    dataLISTS = data.values.tolist()
    billDetails = []
    for l in cartDETAILS:
        finalsIndex = int(l[0]) - 1
        queryIMAGE = finalsLIST[finalsIndex]
        ansList = []
        for rowList in dataLISTS:
            if queryIMAGE in rowList:
                ansList = rowList
        smallList = ansList[2:6]
        smallList.append(ansList[7])
        quantity = int(l[1])
        smallList.append(quantity)
        amt = int(ansList[7].split(' ')[1])
        totalPrice = '₹ ' + str(quantity*amt) + ' /-'
        smallList.append(totalPrice)
        billDetails.append(smallList)
    return billDetails

```

And this function helps to get details of the products in the form of dataframe to show in the form of table.

```

cartFile = 'cartdetails.txt'
finalFile = 'temp.txt'
bill = getBillDetails(cartFile, finalFile)
cartDF = pd.DataFrame(bill, columns = ['PRODUCT', 'COLOR', 'TYPE', 'BRAND', 'PRICE', 'QUANTITY', 'TOTAL PRICE'])
totalBillQuantityListValues = cartDF['QUANTITY'].to_list()
totalBillQuantity = sum(totalBillQuantityListValues)
totalBillQuantityString = str(totalBillQuantity)
totalBillAmountList = cartDF['TOTAL PRICE'].to_list()
totalBillAmountListValues = [int(a.split(' ')[1]) for a in totalBillAmountList]
totalBillAmount = sum(totalBillAmountListValues)
totalBillAmountString = '₹'+str(totalBillAmount)+' /-'
withFinalPrice = [' ', ' ', ' ', ' ', ' ', 'TOTAL : ', totalBillQuantityString, totalBillAmountString]
bill.append(withFinalPrice)
cartDFFinal = pd.DataFrame(bill, columns = ['PRODUCT', 'COLOR', 'TYPE', 'BRAND', 'PRICE', 'QUANTITY', 'TOTAL PRICE'])

```

Final dataframe is created which consists of all the information of the purchase summary to show in the form of table in cart window.

```

import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *

class cartWindow(QWidget):

    def __init__(self):
        QWidget.__init__(self)

        tablemodel = pandasModel(cartDFFinal)
        tableview = QTableView()
        tableview.setModel(tablemodel)

        layout = QVBoxLayout(self)
        layout.addWidget(tableview)
        self.setLayout(layout)

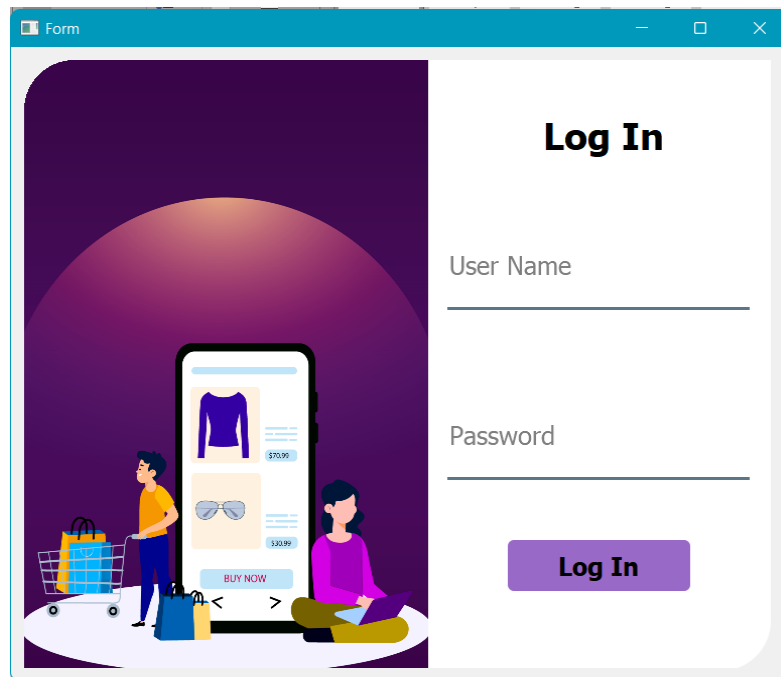
    def main_window(self):
        self.show()

```

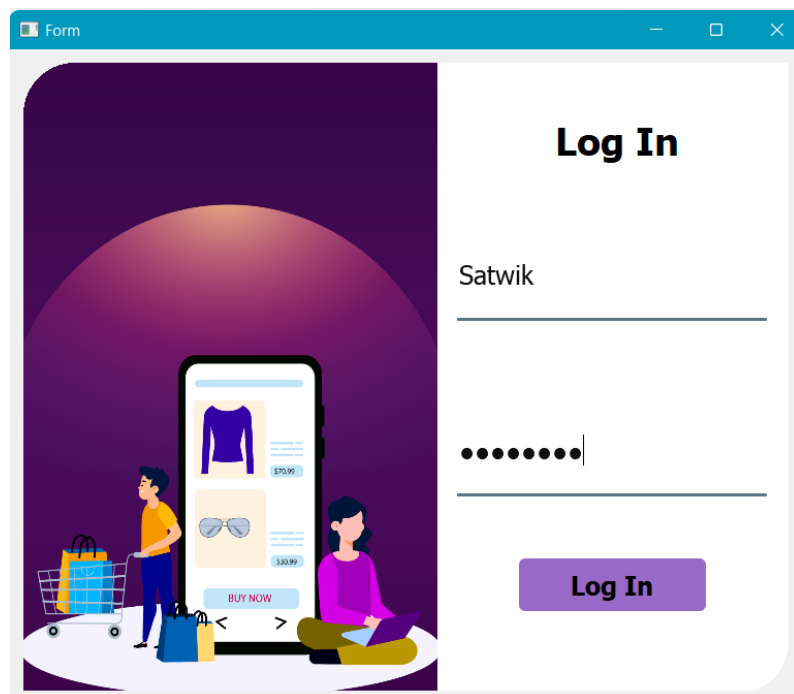
Some part of lines for making Cart Window GUI.

## CHAPTER 4: RESULTS

In this chapter, the results of above implementation is discussed.

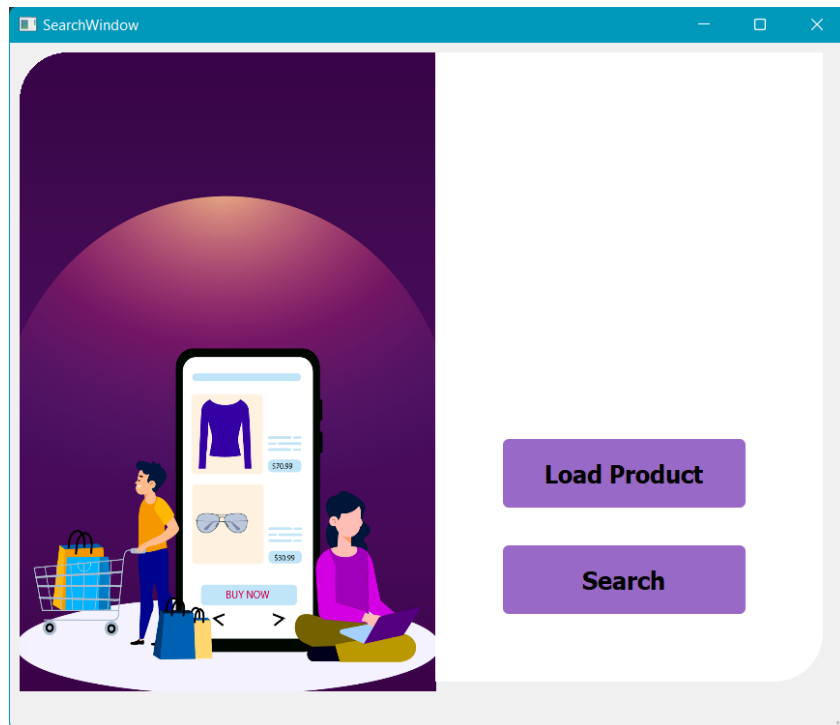


**Fig4.1. Login Window**



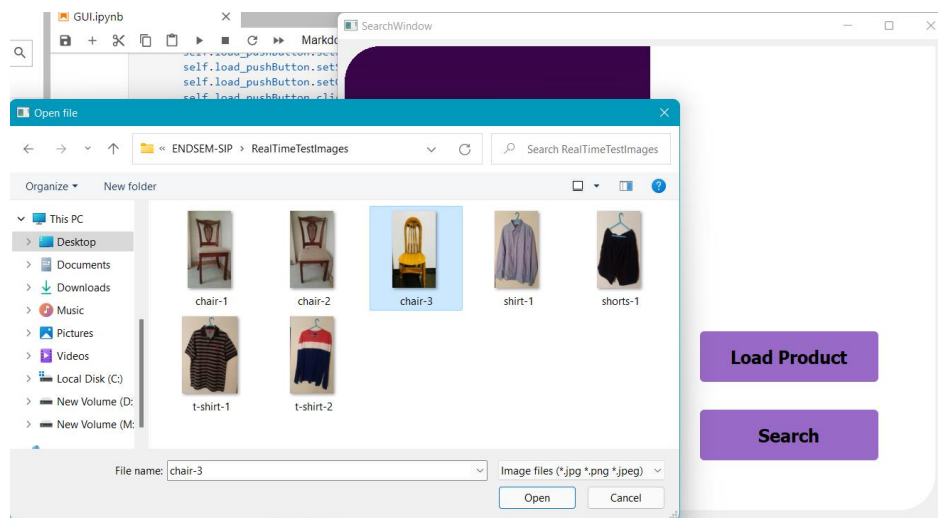
**Fig4.2. Login Window with credentials**

This are Login Window GUI, where user is logged to access the application of Image based product shopping search engine.



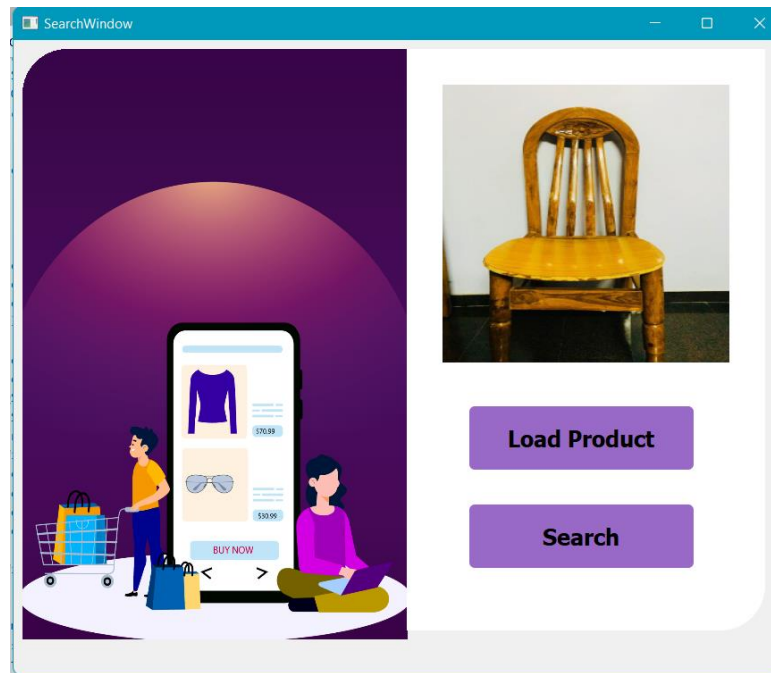
**Fig4.3. Product Query Window**

This is the image of Query Product Load window where product is loaded and searched for getting similar products.



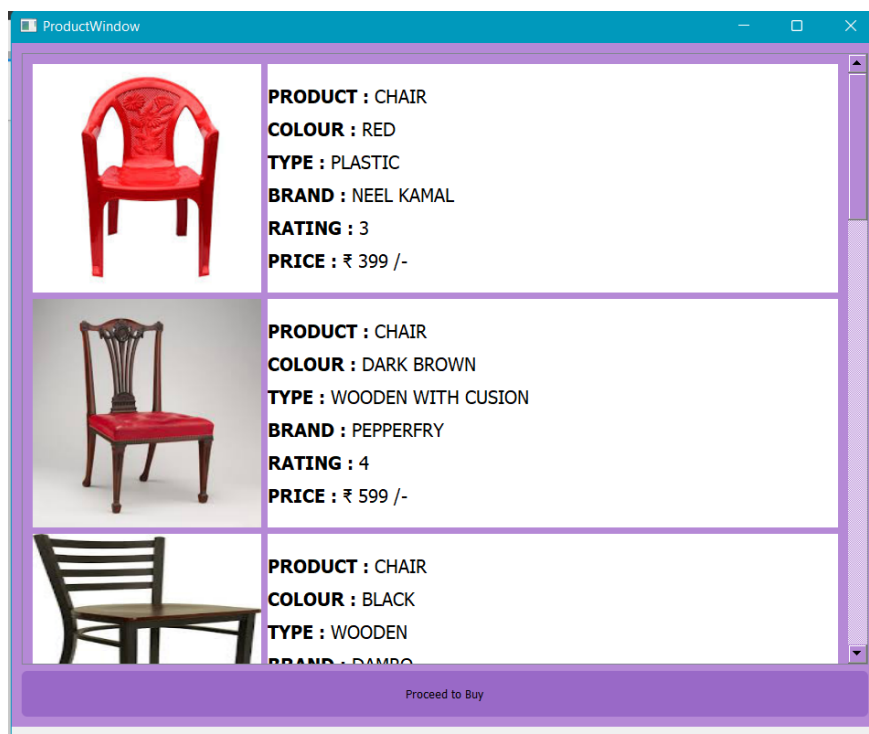
**Fig4.4. File explorer**

This is the image when load product button is clicked and a file explorer is popped to choose the product which we want to search and shop similar products.



**Fig4.5. Product query window after loading product**

This is the image after loading the product. Here product is a chair. And when button search is clicked Product display window is appeared where all the similar products are shown to buy.



**Fig4.6. Product Display window**

Above as we searched for chair all the products related to chair are show in the Product Display window. Here all the details of the product are also shown aside to the product.



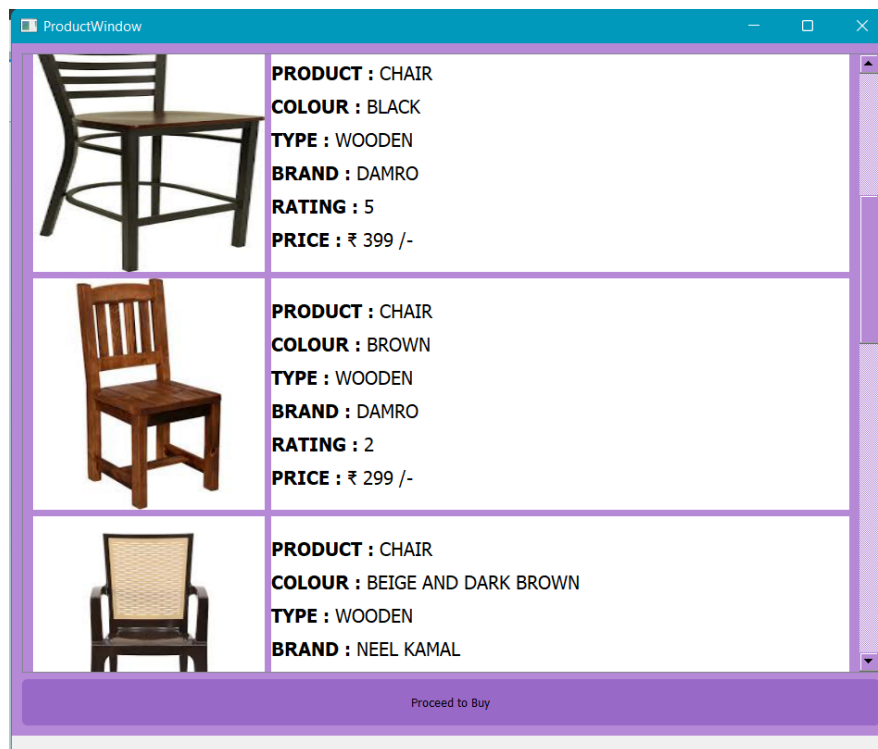


Fig4.7. Product Display Window

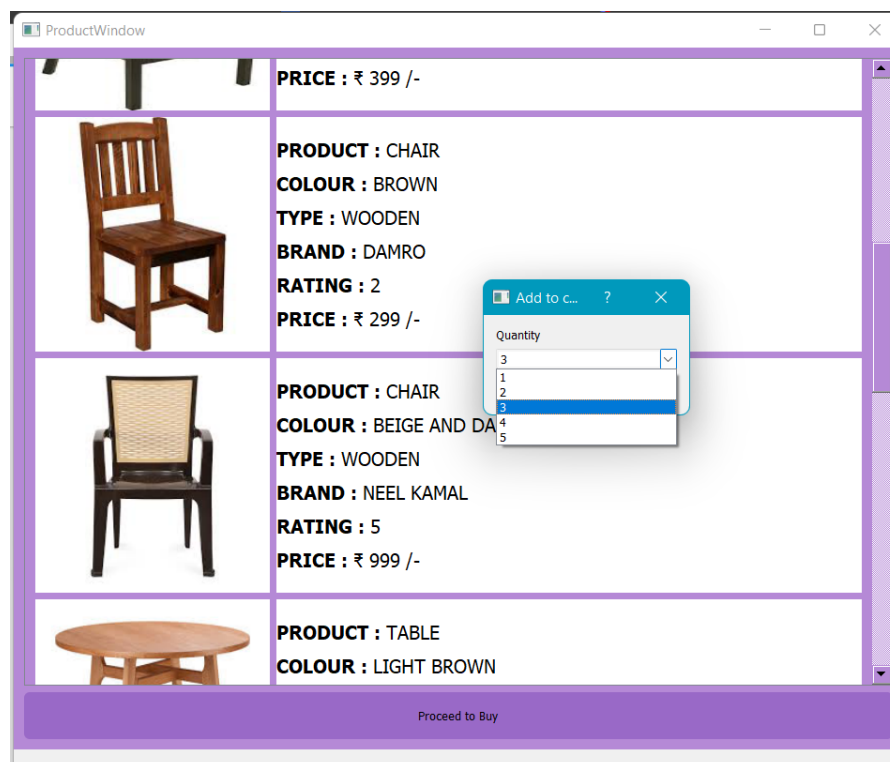


Fig4.8. Product Display Window

Here when product is clicked to add to the cart, user is given a option to choose the quantity.

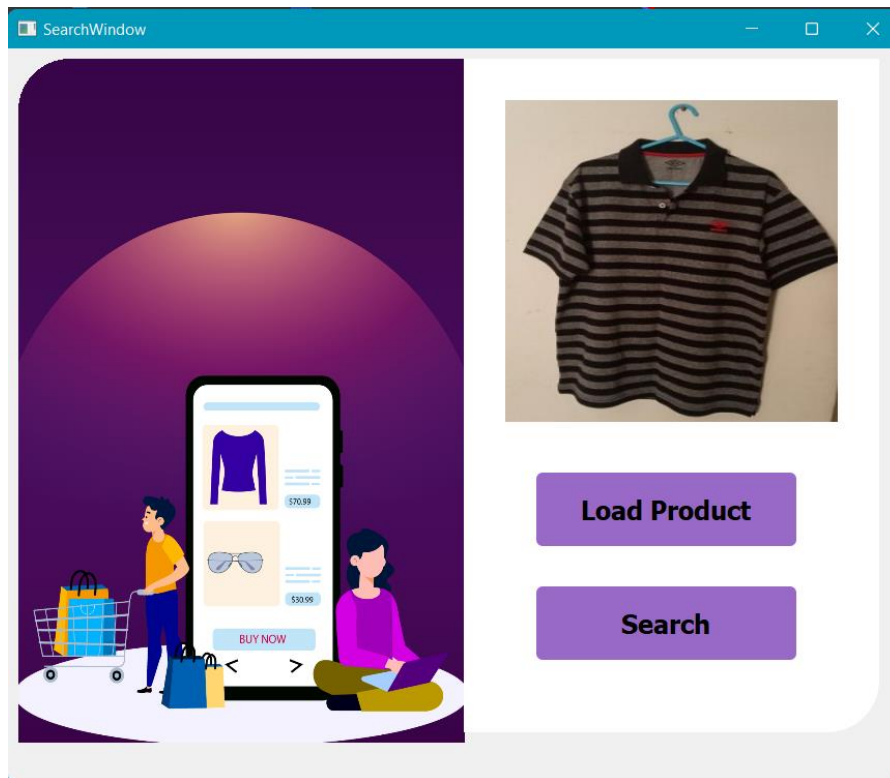


Fig4.9. Product Query Window

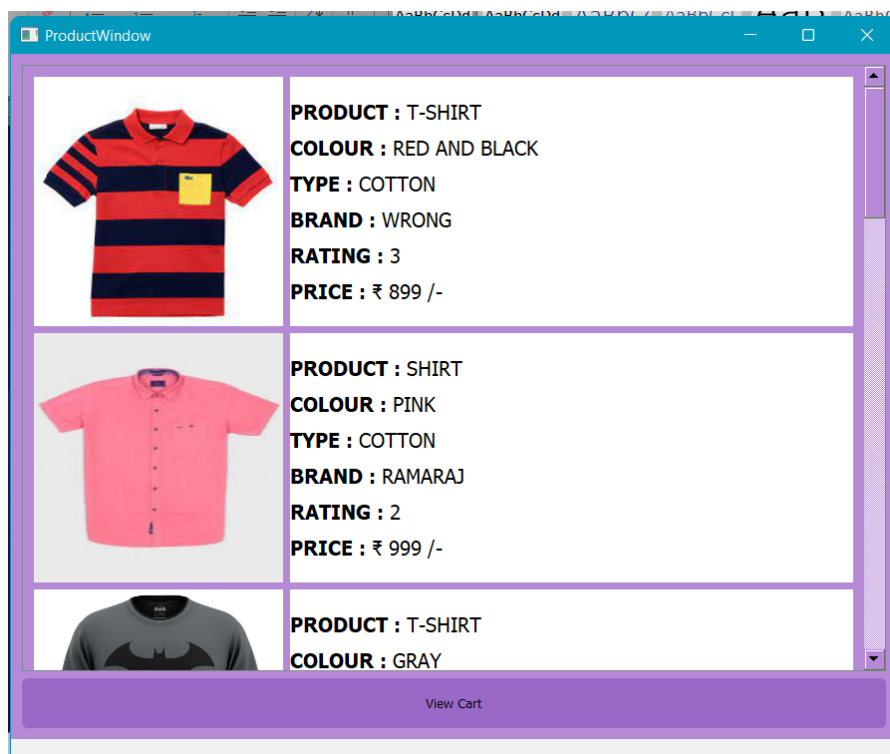
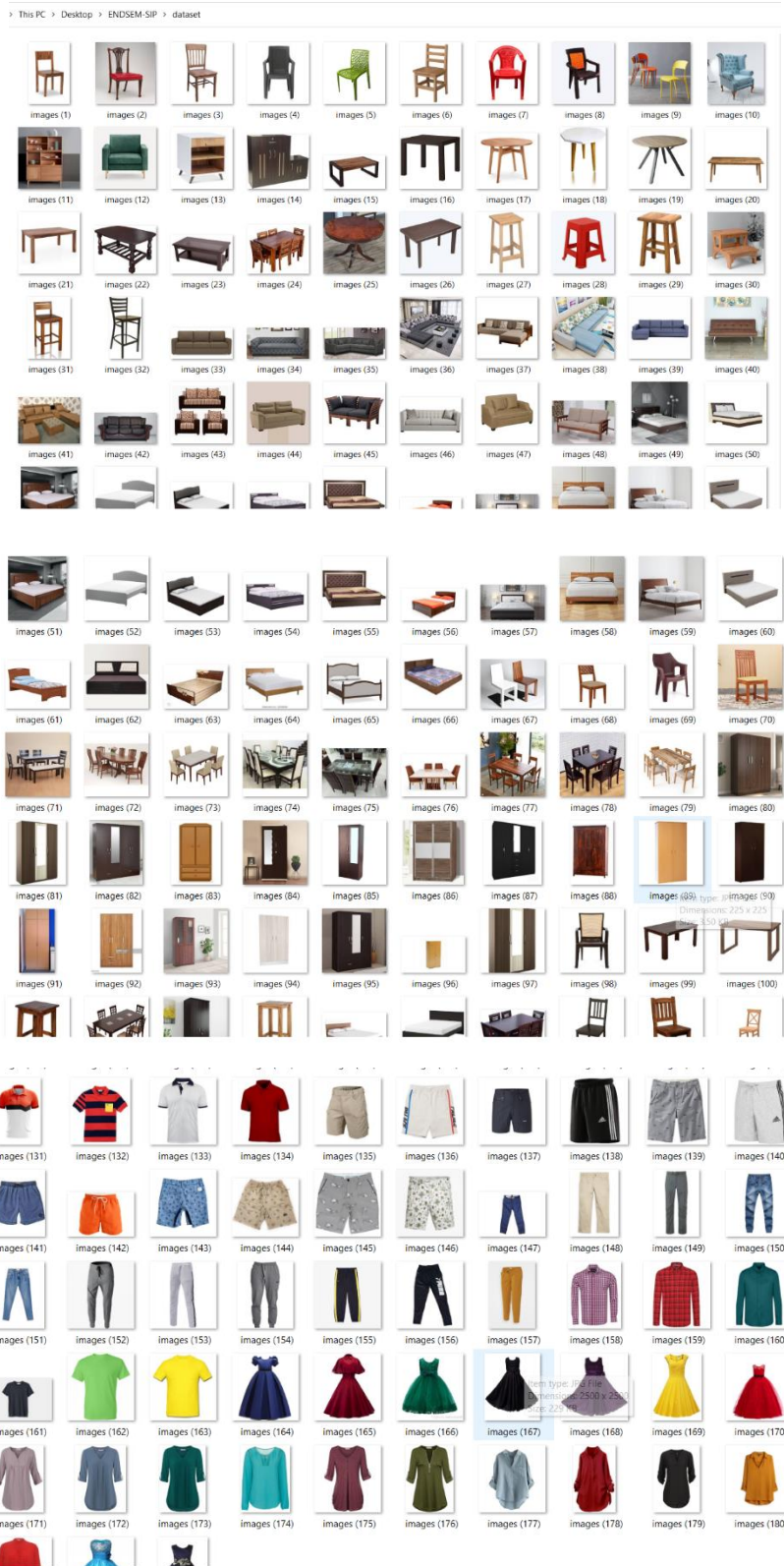


Fig4.10. Product Query Window

Similarly, when product is t-shirt and how product display window shows all the similar products to shop.



**Fig4.11. Dataset**

These are 183 products which is a dataset for our project.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	IMAGE ID	PATH	IMAGE NUMBER	PRODUCT	COLOR	TYPE	BRAND	RATING	PRICE				
1		images (1).jpeg	1	CHAIR	BROWN	WOODEN	IKEA	4	₹ 499 /-				
2		images (2).jpeg	2	CHAIR	DARK BROWN	WOODEN WITH CLUSION	PEPPERFRY	4	₹ 599 /-				
3		images (3).jpeg	3	CHAIR	BROWN	WOODEN	IKEA	3	₹ 499 /-				
4		images (4).jpeg	4	CHAIR	BLACK	PLASTIC	GODREJ	5	₹ 499 /-				
5		images (5).jpeg	5	CHAIR	GREEN	PLASTIC	PEPPERFRY	4	₹ 299 /-				
6		images (6).jpeg	6	CHAIR	LIGHT BROWN	WOODEN	IKEA	3	₹ 499 /-				
7		images (7).jpeg	7	CHAIR	RED	PLASTIC	NEEL KAMAL	3	₹ 399 /-				
8		images (8).jpeg	8	CHAIR	BLACK AND ORANGE	PLASTIC	NEEL KAMAL	4	₹ 399 /-				
9		images (9).jpeg	9	CHAIR	MULTI COLOR	PLASTIC	NEEL KAMAL	4	₹ 399 /-				
10		images (10).jpeg	10	CHAIR	SKY BLUE	SOFA CHAIR	IKEA	5	₹ 7999 /-				
11		images (11).jpeg	11	CLIPBOARD	LIGHT BROWN	WOODEN	IKEA	4	₹ 4999 /-				
12		images (12).jpeg	12	CHAIR	SOFA CHAIR	DARK GREEN	IKEA	4	₹ 7999 /-				
13		images (13).jpeg	13	TABLE WITH SHELF	WHITE AND LIGHT BROWN	WOODEN	IKEA	3	₹ 899 /-				
14		images (14).jpeg	14	CUPBOARD	DARK BROWN	WOODEN	PEPPERFRY	3	₹ 8499 /-				
15		images (15).jpeg	15	TABLE	BROWN	WOODEN	GODREJ	4	₹ 699 /-				
16		images (16).jpeg	16	TABLE	BLACK	WOODEN	GODREJ	4	₹ 899 /-				
17		images (17).jpeg	17	TABLE	LIGHT BROWN	WOODEN	IKEA	5	₹ 399 /-				
18		images (18).jpeg	18	TABLE	WHITE AND LIGHT BROWN	WOODEN	PEPPERFRY	4	₹ 299 /-				
19		images (19).jpeg	19	TABLE	BEDGE	WOODEN	IKEA	4	₹ 299 /-				
20		images (20).jpeg	20	TABLE	BROWN	WOODEN	PEPPERFRY	4	₹ 899 /-				
21		images (21).jpeg	21	TABLE	BROWN	WOODEN	PEPPERFRY	4	₹ 499 /-				
22		images (22).jpeg	22	TABLE	BLACK	WOODEN WITH SHELF	DAMRO	4	₹ 999 /-				
23		images (23).jpeg	23	TABLE	DARK BROWN	WOODEN WITH SHELF	DAMRO	4	₹ 949 /-				
24		images (24).jpeg	24	TABLE	ROSE WOOD	DINING TABLE	NEEL KAMAL	4	₹ 8999 /-				
25		images (25).jpeg	25	TABLE	ROSE WOOD	WOODEN	DAMRO	4	₹ 499 /-				
26		images (26).jpeg	26	TABLE	DARK BROWN	WOODEN	PEPPERFRY	5	₹ 599 /-				

**Fig4.12. Dataset Description**

Details of all the products are in this format in csv file.

## **CHAPTER 5: CONCLUSION AND FUTURE ASPECT**

An image-based search engine for product recognition and classification is developed for searching and collecting information of products similar to it. We also included Graphical User Interface where users can search for similar product by giving an input query image. Our work will help people who have dyslexia and find it difficult to read, by providing them a simple user-friendly interface to buy and shop for the products they need similar to the product they want. This project will identify top ten images in the database which is similar to the input query image. The user can search for images similar to the input image, and add them to cart after entering the quantity of the product required and finally buy then once the bill is generated. This project is implemented with help of VGG16 module available in keras.

Based on our conclusion we wish to further develop our project by adding image augmentation techniques so that the application can also recognize images even if they are in different orientation. We also wish to improve our model to be able identify images even if there are in different light conditions. The speed and accuracy of classification can be improved and database can be enhanced. We also wish to improve our work by developing an APP and make it fully functional on any device.

## CHAPTER 6: REFERENCES:

<https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>

<https://neurohive.io/en/popular-networks/vgg16/>

[https://www.researchgate.net/publication/265385906\\_Very\\_Deep\\_Convolutional\\_Networks\\_for\\_Large-Scale\\_Image\\_Recognition](https://www.researchgate.net/publication/265385906_Very_Deep_Convolutional_Networks_for_Large-Scale_Image_Recognition)