REVERSE IMAGE QUERYING

Submitted in partial fulfillment of the requirements of the degree of

BACHELOR OF COMPUTER ENGINEERING

by

Janhavi Anap (19102043)

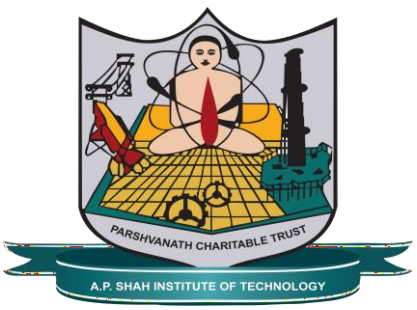
Prathamesh Hambar (19102001)

Tejas Sheth (19102026)

Het Patel (19102005)

Guide

Prof. Rushikesh Nikam



Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2021-2022)

A.P SHAH INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that the Mini Project 2B entitled "Reverse Image Querying" is a bonafide work of **"Janhavi Anap (19102043), Prathamesh Hambar (19102001), Tejas Sheth (19102026), Het Patel (19102005)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering.**

_____	_____	_____
Guide	Project Coordinator	Head Of Department
Prof. Rushikesh Nikam	Prof. Rushikesh Nikam	Prof. Sachin Malave

Date:

A.P SHAH INSTITUTE OF TECHNOLOGY

Project Report Approval

This Mini project report entitled **"Reverse Image Querying"** by **"Janhavi Anap (19102043), Prathamesh Hambar (19102001), Tejas Sheth (19102026), Het Patel (19102005)"** is approved for the degree of ***Bachelor of Engineering*** in ***Computer Engineering***, ***2021-22***.

Examiner Name Signature

1. _____

2. _____

Date:

Place:

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Janhavi Anap (19102043)

Prathamesh Hambar (19102001)

Tejas Sheth (19102026)

Het Patel (19102005)

Date:

Abstract

As days go by, the volume of graphics i.e. visual data keeps increasing exponentially. This presents a more recent and complex set of problems for handling and making use of such vast, disorganized and unregulated data. Multiple organizations have launched many commercial products to tap into this market. There are multiple methods and functions for reverse image querying but most of them are either incomplete or compute-intensive to be a viable option for smaller use cases. In this study, we have implemented and improvised a rudimentary yet efficient solution by using the cosine similarity model to extract similar images for the given input image from the user. The model used is explained comprehensively together with visual representations. We have also explored multiple use cases that can be used as commercial products. The resulting web application can be utilized to act as a reverse image search engine as a standalone application or can be embedded as a sub-module in a larger application.

CONTENTS

List of Figures

List of Tables

List of Graphs

1. Introduction

Information forms the basis of our knowledge. Over the years, we have relied on books, newspapers, and other physical media. They were all stored in libraries or any other educational institutions as limited physical copies. With digitisation, the amount of information that can be stored in a limited space keeps growing exponentially. Then came the era of the World Wide Web (WWW). This enabled a whole new set of information that was generated in huge amounts every day. But the most important aspect of this was how rapidly a lot of information was being shared.

One of the first conceptualisations of a search engine was given by Vannevar Bush in 1939 and it was elaborated on and published in July 1945 in the article "As We May Think" (Bush 2022) of The Atlantic magazine. To quote:

"Consider a future device … in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."

He envisioned Memex (a portmanteau for memory and expansion), a hypothetical electromechanical device in which individuals would compress and store all of their books, records, and communications. This concept highly influenced the development of early hypertext systems, which eventually gave rise to the World Wide Web and further complex technologies based on this. To quote:

"Wholly new forms of encyclopedias will appear, ready-made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified. The Encyclopædia Britannica could be reduced to the volume of a matchbox. A library of a million volumes could be compressed into one end of a desk."

With the coming of WWW, the surge of usage of hyperlinks and studies on link analysis, pioneered crucial components of search engines through algorithms such Hyper Search and Page Rank. Soon, web crawlers were being used for web indexing.

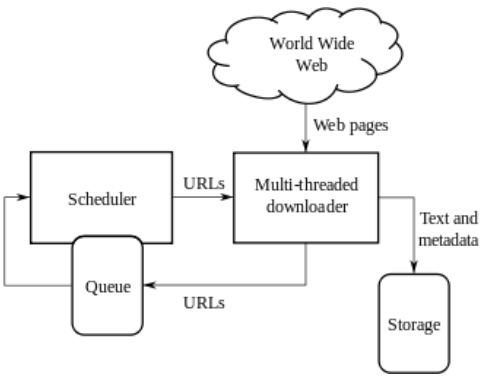


Fig. 1 Architecture of a Web Crawler

Post the DotCom bubble, multiple social media websites started to gain popularity with Facebook, Reddit, Pinterest, Instagram, etc. These websites heavily focused on media sharing more prominently on image sharing. As the internet became more and more accessible, data generation grew steadily. More than a billion photos are uploaded every day and this keeps on incessantly increasing.

This posed a challenge to the developers to make image retrieval systems to cater to such a high influx of images accurately. An image retrieval system is a system used for browsing, searching and retrieving images from a large database of digital images. Image search is a specialized data search used to find images. To search for images, a user may provide query terms such as keyword, image file/link, or click on some image, and the system will return images "similar" to the query. The similarity used for search criteria could be meta tags, colour distribution in images, region/shape attributes, etc. Its types are

- **Image Meta Search:** Usually, metadata is added such as title, captioning, keywords, description, etc. to the images to retrieve them later using annotation words.
- **Image Collection Exploration:** It is a paradigm to explore large digital image repositories.
- **Content-based Image Retrieval:** Instead of textual description, CBIR utilizes Computer Vision techniques to index images based on their contents/features such as colour, texture, shape, objects, etc.

To perform CBIR, there are multiple techniques based on the type of user queries. The prominent ones are

- **Query By Example (QBE):** This includes providing an example to the CBIR to base upon its search.
- **Semantic Retrieval:** A user usually makes an open-ended request which is difficult to process. The CBIR then extracts the features such as colour, texture, etc. and finds similar images containing those features.
- **Relevant Feedback:** In this, the user progressively refines the search results by marking images in the results as "relevant", "not relevant", or "neutral" to the search query, then repeating the search with the new information.
- **Iterative/ Machine Learning:** Convolutional Neural Networks among others are used to find similar images.

Modern-day search engines are highly complex and use parts of multiple techniques or make their own state-of-the-art algorithms to provide accurate results with the possible latency.

In this project, we will be creating a rudimentary CBIR-based QBE search engine with a dataset. Similar images will be queried using the cosine similarity model. A basic interface will be made to facilitate the search by the users.

2. Literature Survey

There are multiple products for identifying the most similar image from a given large image database. Ye Chen [1] states about multiple shortcomings of the current software, databases being disorganized, having no regulation, and the room for improvement for the similarity models. It compares the multiple similarity models in [1] such as Cosine similarity, Pearson similarity, Correlation distance, Chebyshev distance, Manhattan distance, and Euclidean distance. The paper assumes that we are looking for a similar image, not the same image; the features of vectors that are in the same direction can be recognized similarly. The specific distance is not really important. Other models have a much lower accuracy as compared to the cosine similarity model.

This proves to be quite efficient, but it also presents a critical flaw. This model is dependent on any other feature extraction model that is used before it. Thus, any errors or misinterpretation of a feature could give dramatically different results. Another limitation of such a system is that it can only be used for object-based applications. It cannot be used to do a biometric scan for a human face i.e. it cannot distinguish between slight differences between features if not heavily trained and tested. Thus, it can be improved to overcome its flaws.

Image retrieval is the technique of retrieving similar pictures from an image database. Many search engines providing image retrieval are based on text-based or keyword-based approaches. They take text as an input and give similar images as an output. They also tend to provide results based on the caption using Natural Language Processing. It is also tedious to manually annotate keywords for pictures on the net to retrieve actual and similar images. But it becomes very troublesome to interpret the user's intention only based on a few keywords; hence the search results turn out to be ambiguous and unsatisfactory.

Thus, to improve the results, Adrakatti, Arun and Wodeyar, R. and K.R, Mulla [2] used reverse image search engines. They use content-based image retrieval, which outputs images based on features like colour, texture, shape, spatial layout, etc. The reverse image search engines

classify their images based on visual characteristics like colour, texture, and form, which are extracted from the image itself. Some popular reverse search engines include Google, TinEye, and Yandex. The size of their data set is 11.94 billion, 9.14 billion, and 5.6 billion, respectively. Google has the most abundant image information and uses various algorithms considering various attributes to retrieve the output.

But the mentioned tools provide output mainly focusing on images having the same content or object and tend to mistreat their visual characteristics. In our project, we solely focus on colour feature extraction. And similar images from the data set having similar average RGB values to the input image are listed in order of increasing cosine distance as the output to the query image.

To search for similar images, right now, Google Image Search is the most popular software application. Deselaers, Thomas and Keysers, Daniel and Ney, Hermann [3] suggest a few methods clubbed together to achieve image enhancements. The use of features to represent the image content along with its average R-G-B cluster, which is invariant against translation and rotation, in the form of groups to display the images. Since the clusters contain mainly visually similar images, the results obtained are a great starting point.

The search for many words is ambiguous and results in varied output images. Whereas, we almost always obtain two groups as a result, even for words with lesser ambiguity. We have presented an approach using image processing and image retrieval methods, to facilitate the user to meet his search requirements faster.

They [3] present methods like Pearson similarity, correlation distance, etc. to improvise image-based searching in data sets for (most) similar images. To obtain more precise results they use a variety of different algorithms to compare cluster results. Using the proposed features in other applications like image retrieval and copyright infringements, we hope to gain further information on how to improve the results.

The algorithm that Kekre, Hemant and Mishra, Dhirendra and Narula, Ms and Shah, Vidhi [4] present for searching and retrieving images separates Red, Green and Blue colour planes and calculates the average of all row means and all column means of each plane. To form a feature

vector, the features of all three planes are combined. Feature vectors of all images present in the database once calculated are stored in the feature database. The Euclidean distance between the query image's feature vector and the feature database is calculated by the system. Similar images are fetched from the database on the basis of the low value of Euclidean distance. The Low value of Euclidean distance indicates higher relevance to the query image. The algorithm produced better results with the smaller data set. The size of the data set and a number of different classes affected the relevancy of retrieved images.

In various fields, large collections of digital images are being created. Usually, the only way of searching through these collections is through keywords. Veltkamp, Remco Tanase, and Mirela [5] surveyed technical aspects of various content-based image retrieval systems and found that out of 56 systems that were reviewed, 46 systems used colour feature extraction, 38 used texture, 29 used shape, and 5 used face detection. The easier the feature is to extract, the easier it is to implement it in the system, and the easier it is to use that feature class. Most of the systems use colour and texture feature extraction since they are found to be effective. But retrieval of images based on texture doesn't always yield images with similar texture unless the data set contains the majority of a particular texture.

The yielded result also depends on the kind of data set used. A general search in a biased data set would give unsatisfactory results and using a small and insufficient data set would yield irrelevant results. Hence having an equally diverse data set is also very important.

3. Problem Statement

To build a web application that retrieves similar images when given an input image based on their colour using image feature extraction.

4. Objectives and Scope

4.1 Objectives

- To overcome the limitations of text-based search -

Because of the lack of particular keywords, a suitable image may be overlooked. While there is often no relevant text accompanying the photos or videos, they are nevertheless important. In truth, there may be visuals or movies that have nothing to do with the surrounding text. In certain circumstances, the returned results may be useless and have nothing to do with the photos and videos that were requested.

One of the disadvantages of text-based image retrieval is that a word can have different meanings. This problem is best illustrated with an example, searching for the images or videos of jaguar or Apple. The system can't differentiate whether the user is looking for a jaguar car or a jaguar animal.

- Colour Based Reverse Image Search -

The web application is supposed to display almost similar images in a matter of seconds. Here, we fetch each image from the given dataset on the basis of its average Red, average Green, and average Blue values. We are expected to be looking for a similar image or images, and not the same image. That is calculated by using the Cosine Similarity Model. We preferred the Cosine Model over Pearson similarity, Correlation distance, Chebyshev distance, Manhattan distance, and Euclidean distance models as it is comparatively the best function, with greater similarity returns.

- To check the efficiency of Image Search Engines.

4.2 Scope

1. Protect Copyrights

Many photographers use reverse image search to discover whether an image of theirs has been published somewhere online without their authorization. **Reverse image search** is a great tool to discover such re-use of images.

2. Find Object Details

Many times we come across images of products, and food items that are fascinating and want to know more about them, but we don't know their names. In such cases, we rely on searching using keywords, but without proper names, searching engines fail to deliver the desired results. Reverse Image search eliminates the problem of finding data regarding unidentified products, plants, animals, flowers etc.

3. Authenticity

To realize the manipulated version of any image. It could be cropped, colour modified, revolved or adjusted.

5. Experimental Setup

5.1 Hardware requirements

- **CPU:** 1.8 GHz or faster 64-bit processor; Quad-core or better recommended.
- **RAM:** Minimum of 4GB of ram.
- **Storage:** 4GB of free hard disk space.

5.2 Software requirements

1. **Python:** General Programming Language used to code the model, includes several libraries.
2. **Python libraries:** Numpy, Matplotlib, PIL (Python Imaging Library), pandas, streamlit, os.
3. **VS Code:** IDE to run, train and test the ML model. Plenty of extensions, open-source, cross-platform support,
4. **Git & GitHub:** Version Control System used for collaboration.
5. **Streamlit:** Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.
6. **Google Workspace:** Project Management Tool to improve team coordination and file sharing.
7. **Teamgantt:** Project Management Tool to improve team coordination and file sharing using Gantt chart.

6. System Design

6.1 Class diagram

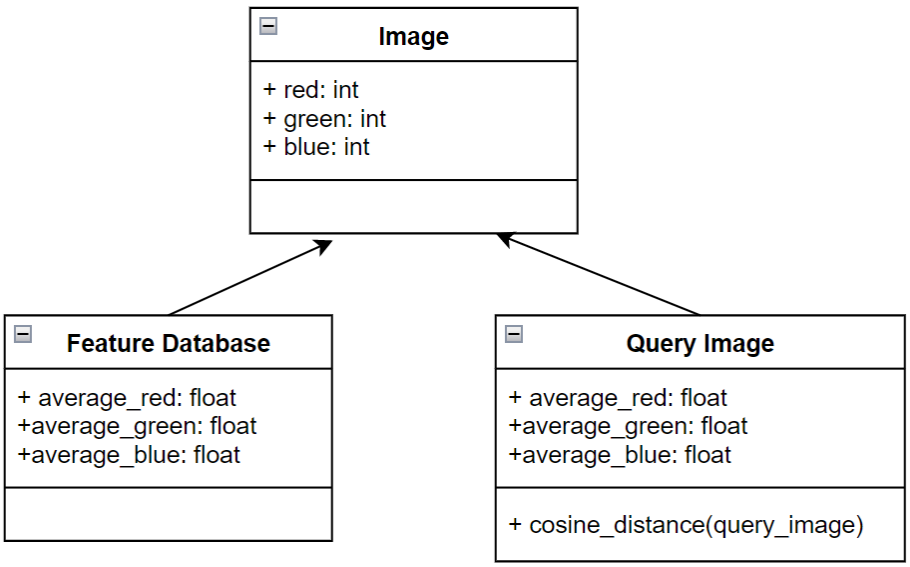


Fig 6.1: Class Diagram

An image has multiple properties out of which the colour property has three values. Red, Green, and Blue. By iterating through all the pixels of the image while collecting Red, Green and Blue values of the pixel, average_red, average_green, and average_blue values can be calculated. Similarly, average_red, average_green, and average_blue values of the query image can also be calculated. The cosine distance between these values is calculated which helps to provide the results.

6.2 Architecture Diagram

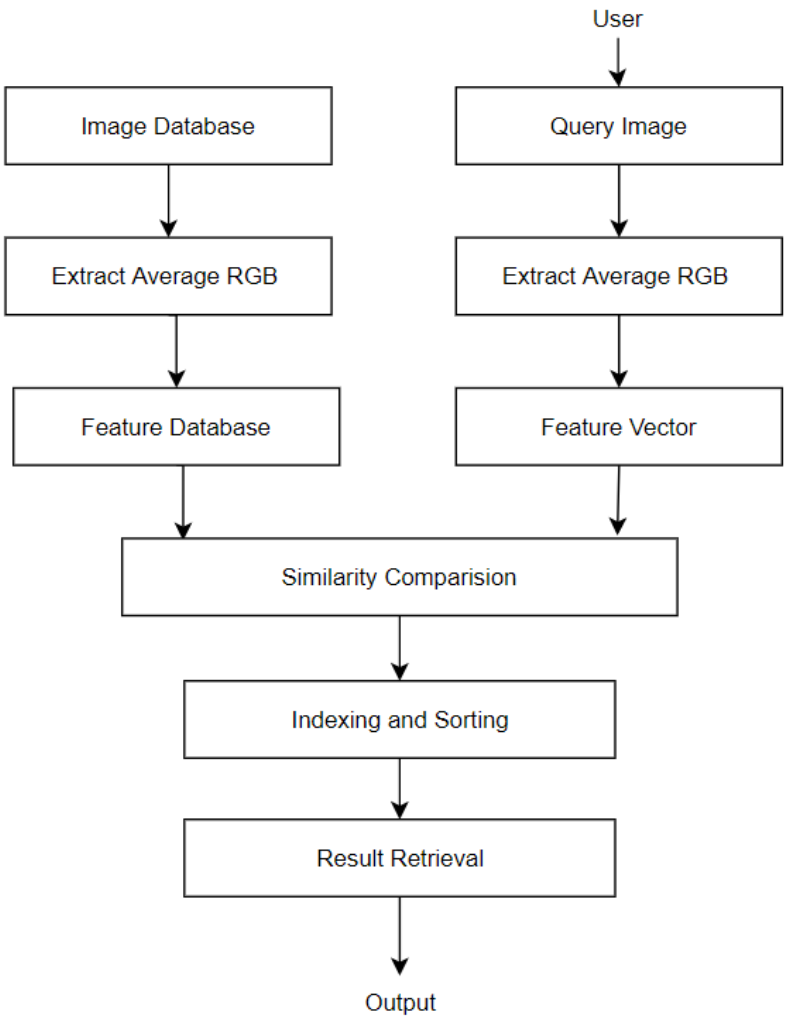


Fig 6.2: Architecture Diagram

The architecture of the project includes a database that stores images as well as its features. The images that are used in this project are in jpg format. Its average Red, Green and Blue values are stored as separate attributes in the database.

The cosine distance calculated between every feature of every image in the database and the features of the query image is stored along with the index of the image. This stored cosine-distance data is then sorted in ascending order and the results are displayed.

6.3 Activity Diagram

6.3.1 Pre-processing

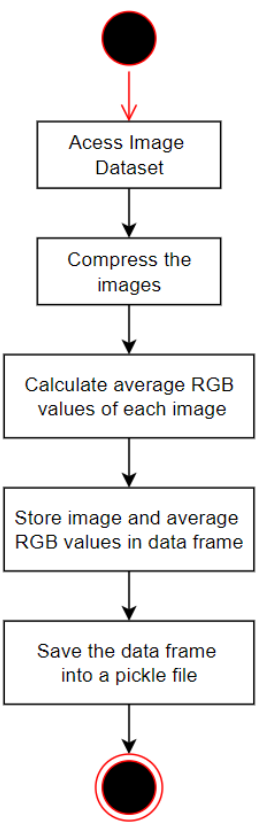


Fig 6.3.1: Activity Diagram - Preprocessing

Before creating the feature database, all the images in the dataset are compressed to improve the performance time. The compression of the image is done by resizing the images with a fixed base width of 300 pixels. After the compression by iterating through all the pixels of the

image while collecting Red, Green and Blue values of the pixel, average_red, average_green, and average_blue values are calculated and stored. Since the program is implemented in python the above data is stored in a data frame. This data frame is then stored as a pickle file for easy access to the data.

6.3.2 Retrieving Similar Images

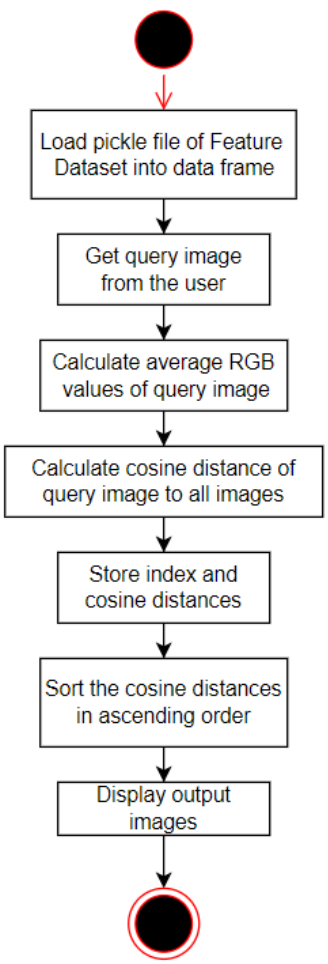


Fig 6.3.2: Activity Diagram - Retrieving Similar Images

For retrieving similar images, the pickle file is first loaded into a data frame. The query image is then fetched from the user. By iterating through all the pixels of the query image while collecting Red, Green and Blue values of the pixel, average_red, average_green, and average_blue values are calculated. The distance between average_red, average_green, and average_blue values of the query image and from the feature database are compared. This comparison is done by calculating the cosine distance between them. The index of the image from the dataset and the cosine distance between the image and the query image is stored and sorted. The sorted images are then displayed as a result of the query image

6.4 Sequence Diagram

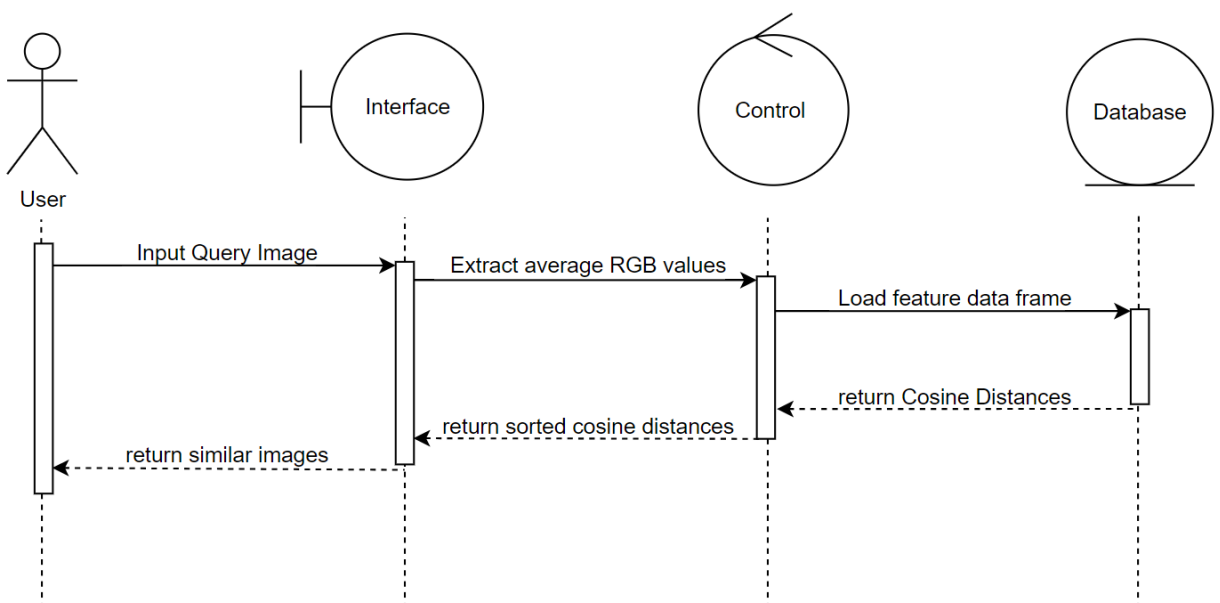


Fig 6.4: Sequence Diagram

The sequence diagram with respect to the user can be explained as follows. First, the user inputs a query image through the user interface implemented on the web browser. In the

backend, average_red, average_green, and average_blue values are calculated. From the database, the feature database is extracted. Then the cosine distance between the average_red, average_green, and average_blue values from the data frame and the query image is calculated. The distances are then indexed and sorted in ascending order. The images with minimum cosine distance are displayed as a result of the user interface on the web browser.

6.5 UML Diagrams and Program Flow

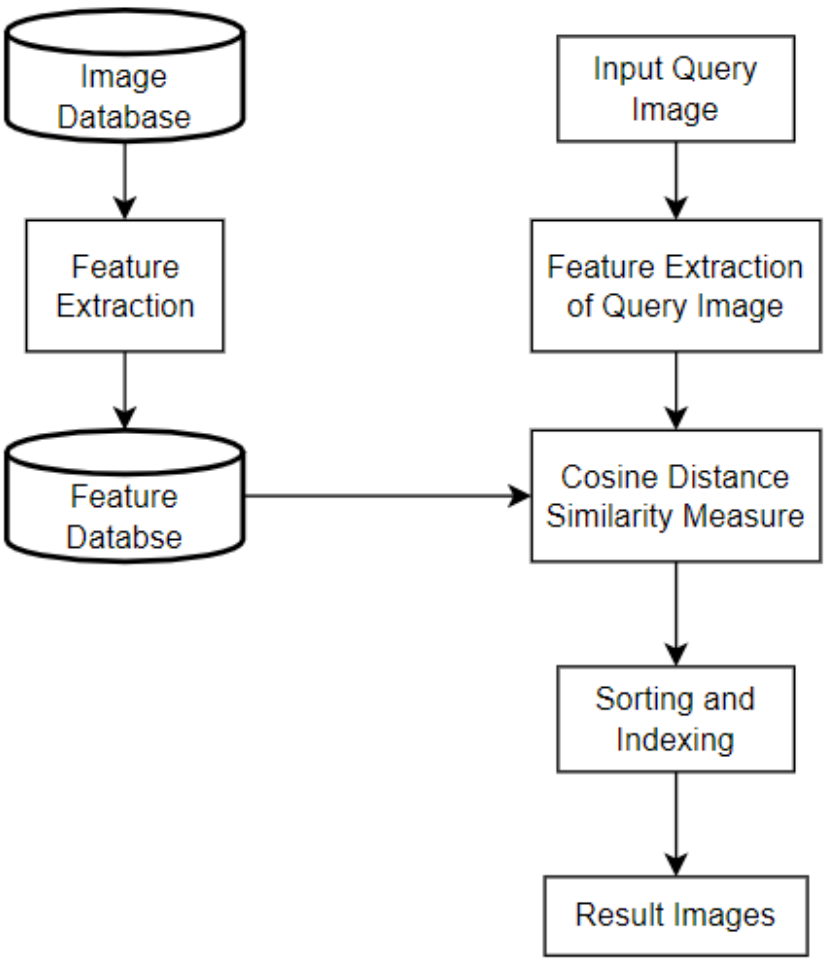


Fig 6.5: UML Diagram and Program Flow

As the general flow of the program, upon the input of the query image the average Red, Green, and Blue values are extracted as features. These features are then compared with the features of all the images present in the feature database. The distance between these images is sorted in ascending order and indexed. The images having the least distance are displayed as the result.

7. Project Planning and Scheduling

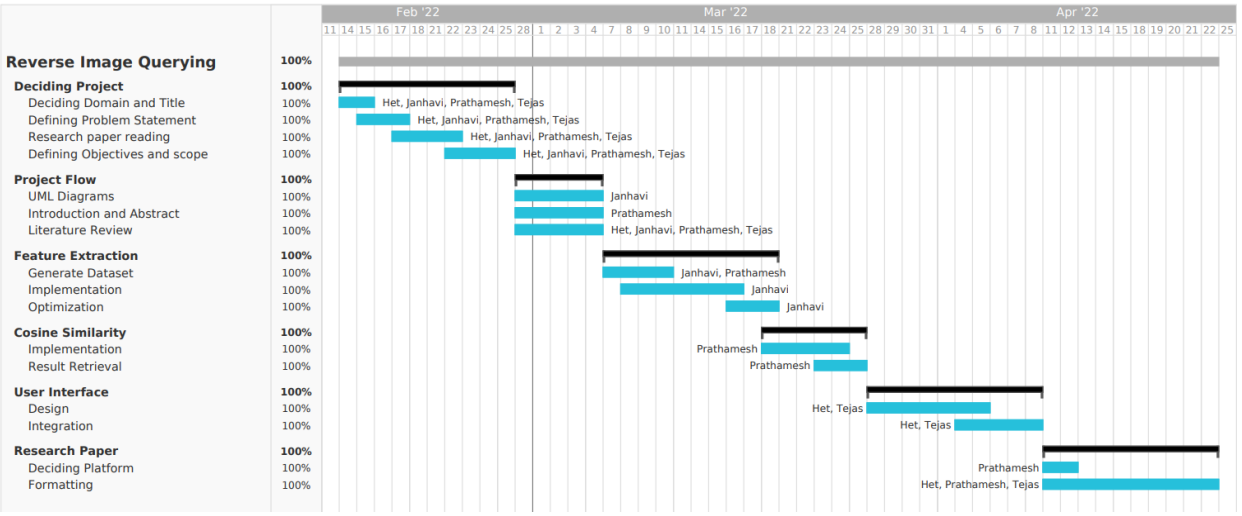


Fig 7: Gantt Chart

8. System Design/Code and Implementation

8.1 System Design and Code

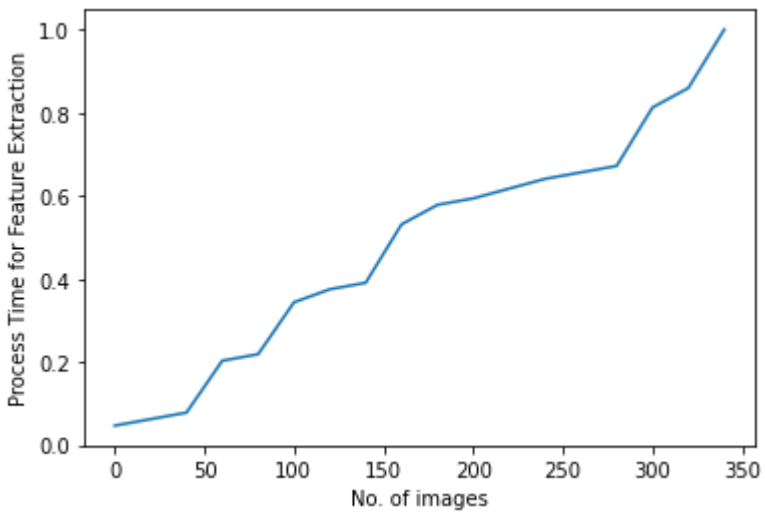
To resize-compress the image:

With basewidth already defined, as 300 pixels, the image is compressed with respect to the base_width dimensions.

```
width_percent = base_width/float(image.size[0])
hsize = int(float(image.size[1])*float(width_percent))
image = image.resize((basewidth, hsize), Image.ANTIALIAS)
image.save(dir + "/" + img)
```

Compression of images is necessary since the time for extracting the features increased with the increasing size of the data set

Processing Time for Feature Extraction.



Graph 1: Processing Time for Feature Extraction

To extract features of the image:

Images consist of pixels and each pixel has Red, Green and Blue as its attributes. By iterating through all the pixels of the image while collecting Red, Green and Blue values of the pixel, average_red, average_green, and average_blue values are calculated.

```
image = Image.open(dir + "/" + img)
img_array = np.array(image)
# calculate average rgb values
red = np.average(img_array[:, :, 0])
green= np.average(img_array[:, :, 1])
blue= np.average(img_array[:, :, 2])
```

To calculate cosine distances from query image:

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Cosine Similarity =

$$S_c(A,B) := cos(\theta) = \frac{A.B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Here the vectors correspond to the feature vector of average_red, average_green and average_blue values of the query image and the image from the database. Considering the image database and its features stored in dataframe ‘df’ and the cosine distances are stored in a list ‘cosine_distance’.

```
# calculate cosine distance from query image to all images
cosine_distance = []
idx = 0
for i in range(len(df)):
    temp_data = df.iloc[i, -3:]
    temp_data = np.array(temp_data).reshape(1, -1)
    dist = cosine(query_feature[-3:], temp_data)
    cosine_distance.append([dist, idx])
    idx += 1
```

To get result images:

The cosine distances that are calculated and stored in a list should be sorted in ascending order. The images having least cosine distance or most similar to the input query image will be stored at the beginning and the images having maximum cosine distance or least similarity will be stored in the end.

```
# sorting the cosine distances
cosine_distance.sort()
# storing the images and their respective cosine distances
result = []
for dist, idx in cosine_distance:
    result.append(idx)
```

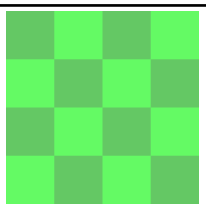
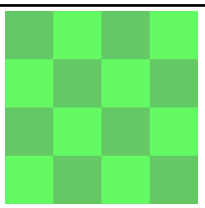
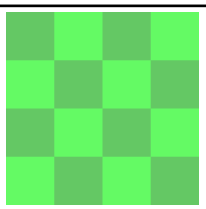
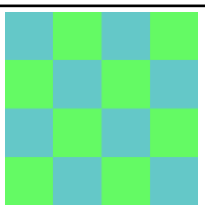
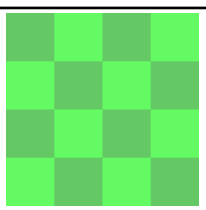
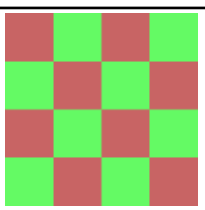
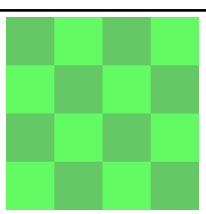
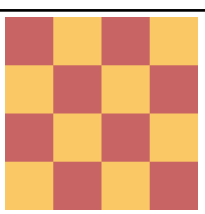
Query Image	Data frame Image	Cosine Distance(cos_dist)
 A = [100,225,100]	 B = [100,225,100]	A.B = 100*100+225*225+100*100 = 70625 |A| = √((100)^2+(225)^2+(100)^2) = 265.75 |B| = √((100)^2+(225)^2+(100)^2) = 265.75 cos(A,B) = 70625/(265.75*265.75) = 1 cos_dist = 1-cos(A,B) = 1-1 = 0
 A = [100,225,100]	 B = [100,225,150]	A.B = 100*100+225*225+100*150 = 75625 |A| = √((100)^2+(225)^2+(100)^2) = 265.75 |B| = √((100)^2+(225)^2+(150)^2) = 288.31 cos(A,B) = 75625/(265.75*288.31) = 0.987 cos_dist = 1-cos(A,B) = 1-0.987 = 0.013
 A = [100,225,100]	 B = [150,175,100]	A.B = 100*150+225*175+100*100 = 64375 |A| = √((100)^2+(225)^2+(100)^2) = 265.75 |B| = √((150)^2+(175)^2+(100)^2) = 251.25 cos(A,B) = 64375/(265.75*251.25) = 0.964 cos_dist = 1-cos(A,B) = 1-0.964 = 0.036
 A = [100,225,100]	 B = [225,150,100]	A.B = 100*225+225*150+100*100 = 66250 |A| = √((100)^2+(225)^2+(100)^2) = 265.75 |B| = √((225)^2+(150)^2+(100)^2) = 288.31 cos(A,B) = 66250/(265.75*288.31) = 0.865 cos_dist = 1-cos(A,B) = 1-0.834 = 0.135

Table 1: Example

Using the formula the cosine distances between the query image and sample images are calculated. Here the cosine distance determines how similar the two images are. As it can be seen from the TABLE 1 as we increase the deviation of the Red, Green, and Blue values of the query image, its cosine distance from the original image increases.

8.2 System Implementation

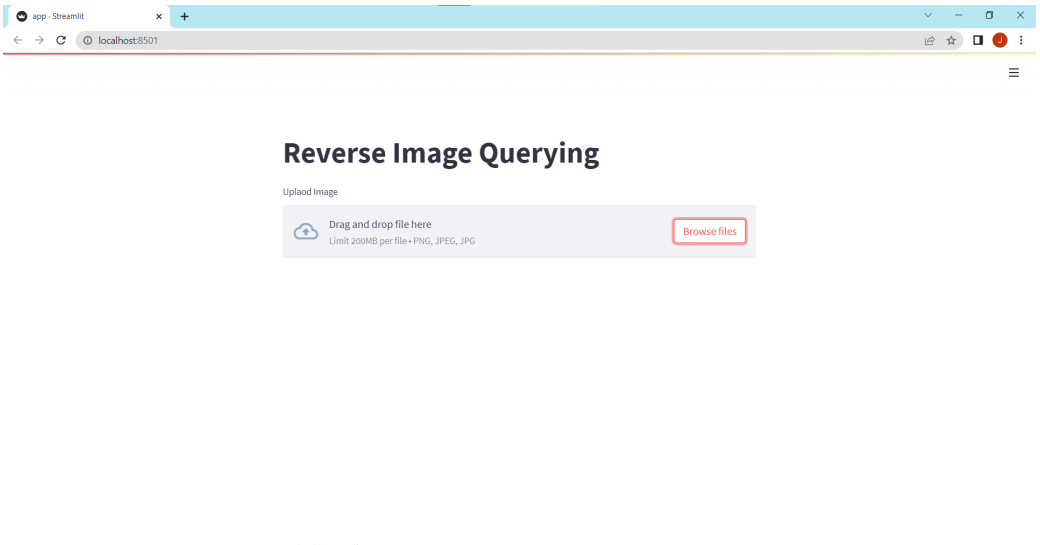


Fig 8.1: User interface

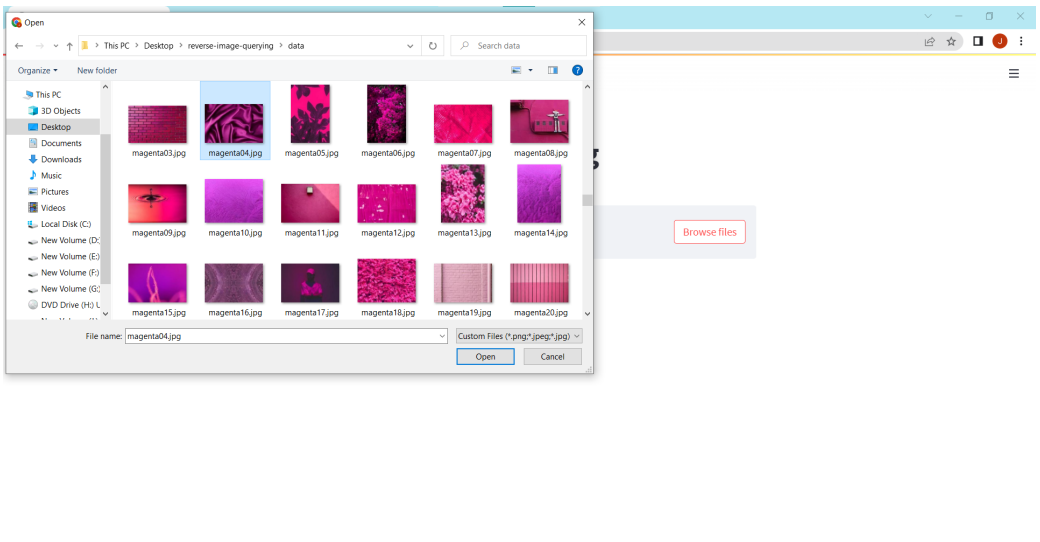


Fig 8.2: Selection of a magenta image

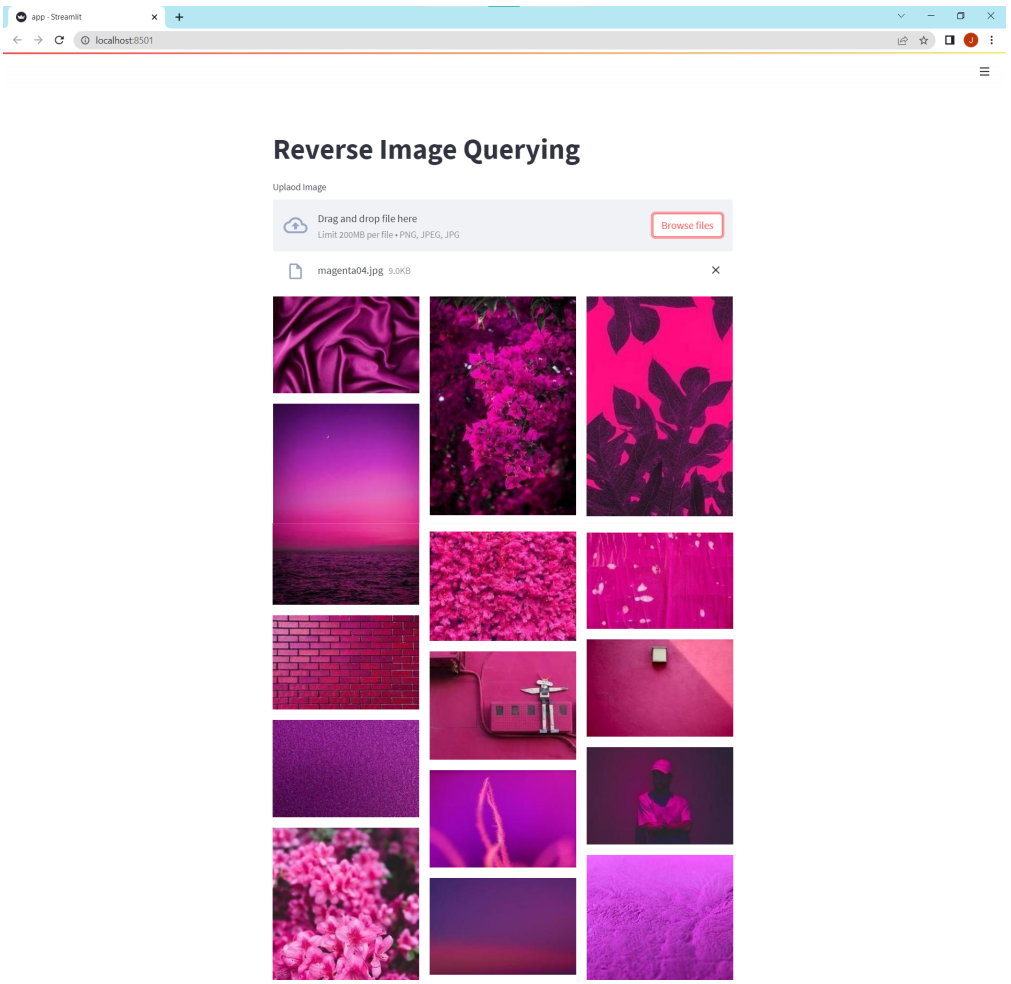


Fig 8.3: Output of similar magenta images

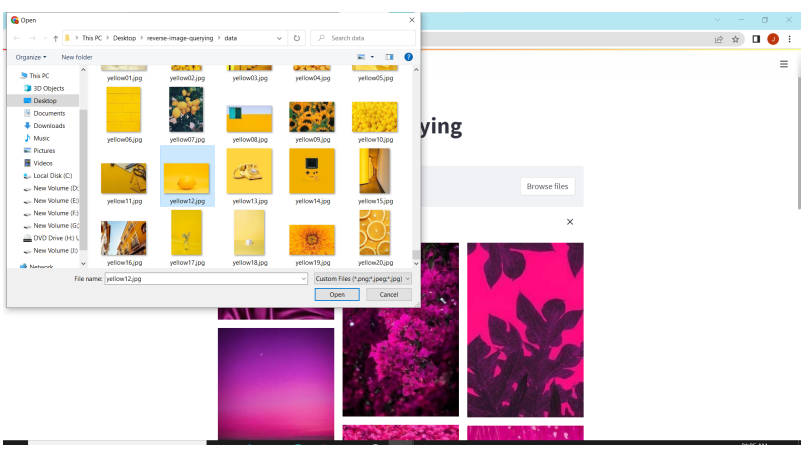


Fig 8.4: Selection of a yellow image

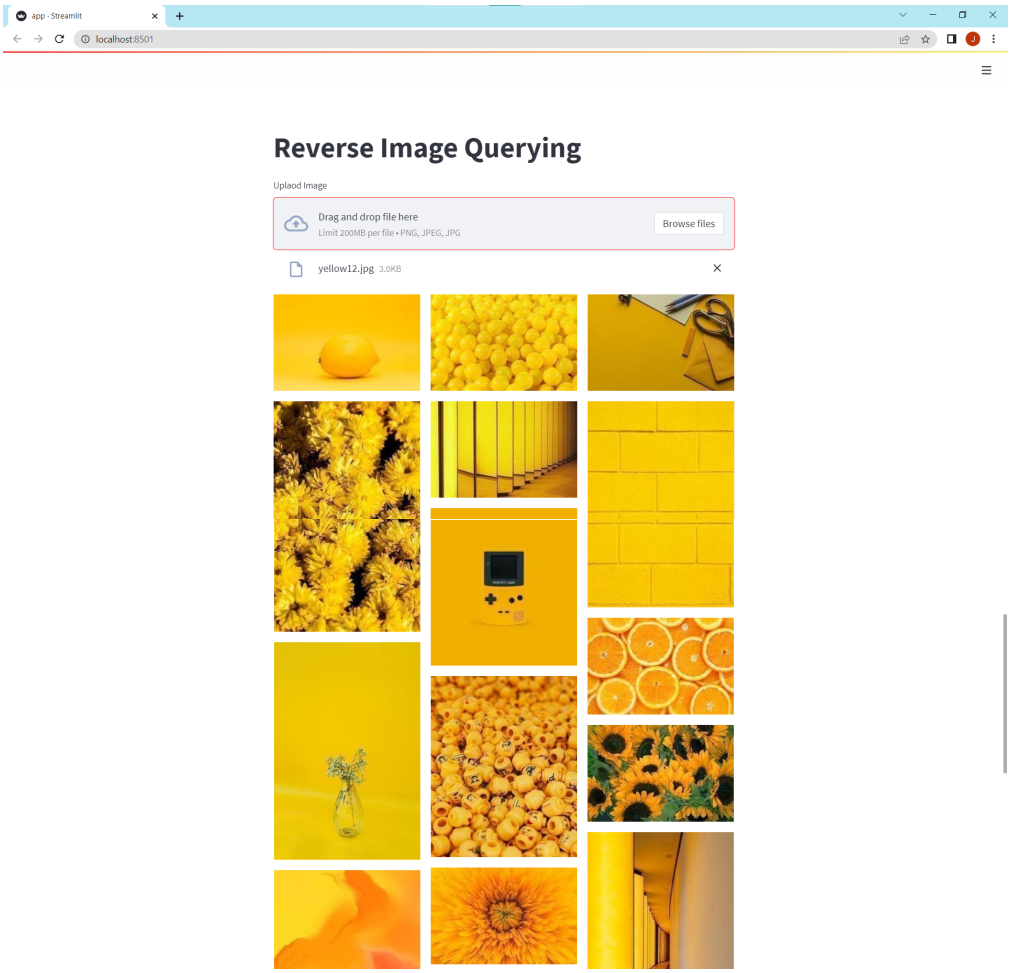


Fig 8.5: Output of similar yellow images

9. Results

Our final decision was to use the cosine similarity model, we came to the consensus after thorough research on the cosine similarity model and found out that it was, by far, the most effective model to fetch the most similar images from the given database. We focused on retrieving similar images based on their average R-G-B color attributes. We managed to achieve great results, as seen in Fig. 3 using simple yet highly effective algorithms. One of the outstanding problems was the ability to say definitively that if the two products were not considered to be similar. This is a difficult problem to solve because there will be times when we need to search for an image that we do not necessarily have any similarities with, but still want to try and find the closest match. It can be seen from Table. As we increase the deviation of the Red, Green, and Blue values of the query image, its cosine distance from the original image increases. Hence, the cosine distances between the query and sample image determines how similar the two images are.

10. Conclusion

According to our results, using the best similarity model we choose, there has been a significant increase in accuracy, based on the colour attributes. We managed to achieve great results using simple yet highly effective algorithms. We can further improve the results of the model with the help of better hardware or accelerated computing or high-performance computing techniques like multi-threading, and many more. This model can be implemented in a number of sectors including fashion, design, automobile, and Fast-moving consumer goods (FMCG). Moreover, it can be utilized to realize manipulated versions of any image, and protect against unauthorized reuse of published images.

References

[1] Chen, Ye. (2020). Exploring the Impact of Similarity Model to Identify the Most Similar Image from a Large Image Database. Journal of Physics: Conference Series. 1693. 012139. 10.1088/1742-6596/1693/1/012139.

[2] Adrakatti, Arun and Wodeyar, R. and K.R, Mulla. (2016). Search by Image: A Novel Approach to Content Based Image Retrieval System. International Journal of Library Science. 14. 41-47.

[3] Deselaers, Thomas and Keysers, Daniel and Ney, Hermann. (2003). Clustering visually similar images to improve image search engines.

[4] Kekre, Hemant and Mishra, Dhirendra and Narula, Ms and Shah, Vidhi. (2011). COLOUR FEATURE EXTRACTION FOR CBIR. International Journal of Engineering Science and Technology.

[5] Veltkamp, Remco and Tanase, Mirela. (2000). Content-Based Image Retrieval Systems: A Survey. Technical report, Utrecht University.

[6] Luo, Bo and Wang, Xiaogang and Tang, Xiaoou. (2003). World Wide Web Based Image Search Engine Using Text and Image Content Features. Proc SPIE. 10.1117/12.476329.

[7] Brin, S., and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30(1-7), 107–117.

[8] Kekre, Hemant and Mishra, Dhirendra. (2010). CBIR using Upper SixFFT Sectors of Colour Images for Feature Vector Generation. International Journal of Engineering and Technology. 2.

[9] Gudivada, V. N., and Raghavan, V. V. (1995). Content based image retrieval systems. Computer, 28(9), 18–22. doi:10.1109/2.410145

Appendix

Appendix A

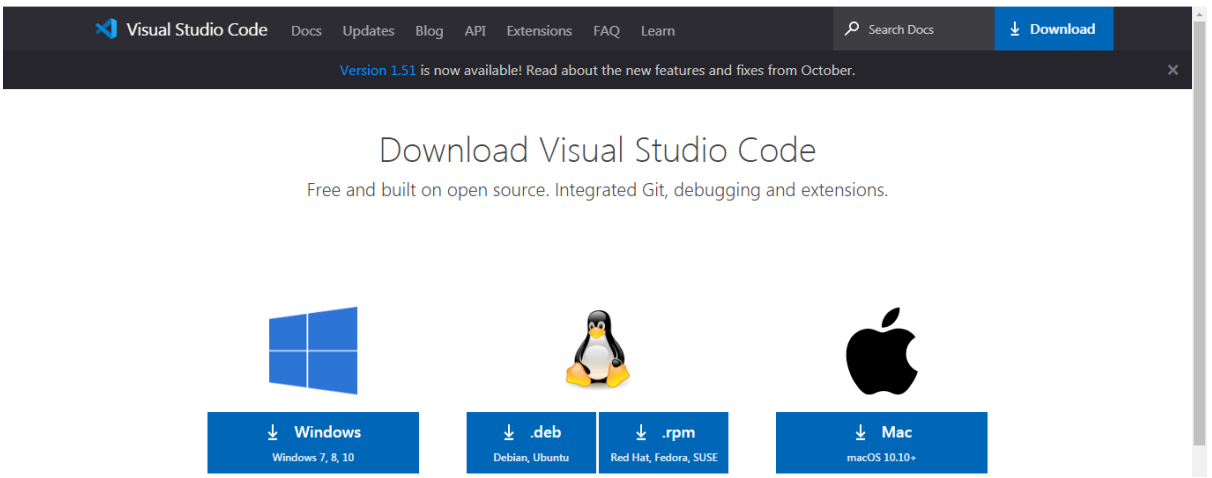
Python Download and Installation

1. Visit the official website and go to https://www.python.org/downloads/. Click the Download button.
2. Once we click the download button, it might ask for a location to save the file. Select an appropriate location and then proceed towards the installation.
3. Double Click the downloaded .exe file and select the Add Python to PATH checkbox below to ensure it is automatically added to the Windows Environment variable. Else we have to do it later on manually. Once the box is checked, click on Install Now.
4. At the time of installation of python, the pop-up will show that the installation is in progress here.
5. Once the setup is complete, we will get a message like this. Click on the Close button to finish the installation of python.
6. Once Python is installed, go to the Windows search bar and type Python, and we will find a desktop app called Python 3.7 (32-bit). Click on that and a command prompt will open.

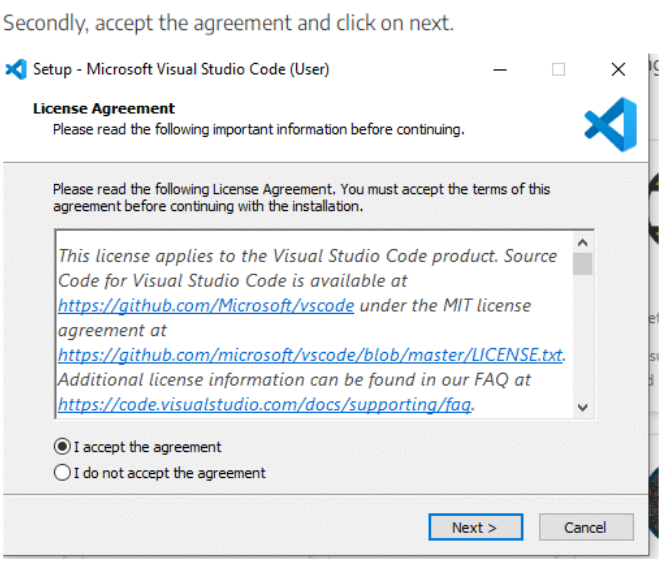
Appendix B

VS Code Download and Installation

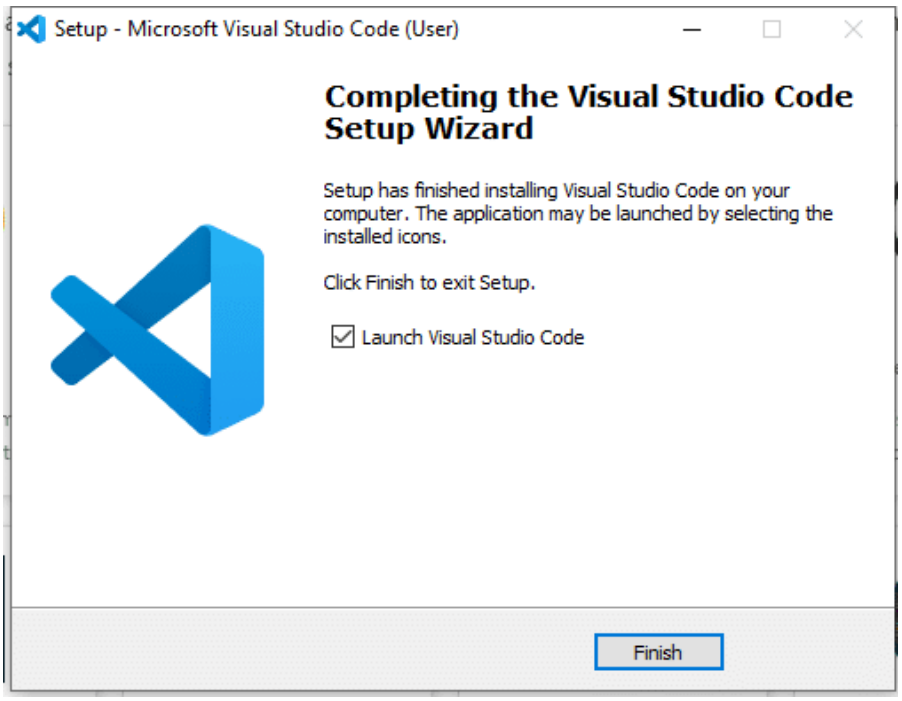
1. Download VS code from https://code.visualstudio.com/download
2. Download the Visual Studio Code installer for Windows. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). Then, run the file



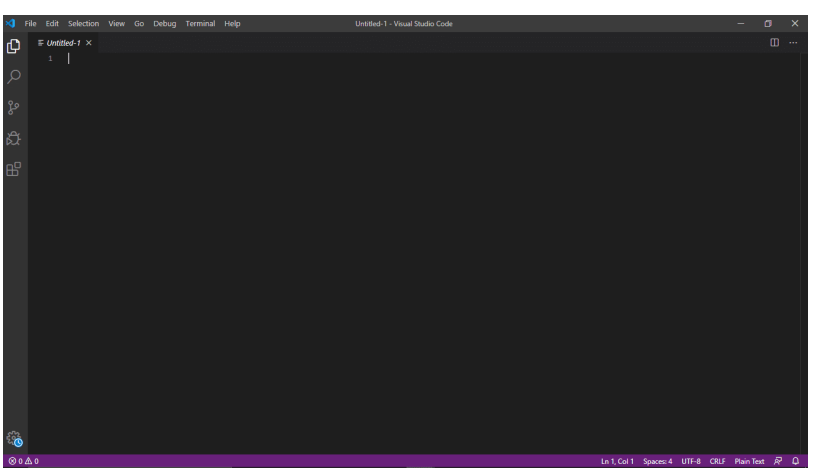
3. Accept the agreement and click “next.”



4. After accepting all the requests press the finish button. By default, VS Code installs under: **“C:\users{username}\AppData\Local\Programs\Microsoft VS Code.”**



5. If the installation is successful, you will see the following



Acknowledgement

We have great pleasure in presenting the mini project report on **"Reverse Image Querying".** We take this opportunity to express our sincere thanks towards our guide **Prof. Rushikesh Nikam,** Department of Computer Engineering, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of the project.

We thank **Prof. Sachin Malave, Head of Department**, Computer Engineering, APSIT for his encouragement during the progress meeting and providing guidelines to write this report.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Student Name1: Janhavi Anap
Student ID1: 19102043

Student Name2: Prathamesh Hambar
Student ID2: 19102001

Student Name3: Tejas Sheth
Student ID3: 19102026

Student Name4: Het Patel
Student ID4: 19102005