**Appendix - I**

**Source Code**

#Algorithm

#Take the query image and convert it to grayscale.

#Now Initialize the ORB detector and detect the keypoints in query image and scene.

#Compute the descriptors belonging to both the images.

#Match the keypoints using Brute Force Matcher.

#Show the matched images.

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

# Read the query image as query\_img

# and traing image This query image

# is what you need to find in train image

# Save it in the same directory # with the name image.jpg

query\_img=cv2.imread('/content/drive/MyDrive/Images\_for\_ORB/query\_img\_bw.jpg')

train\_img=cv2.imread('/content/drive/MyDrive/Images\_for\_ORB/train\_image\_bw.jpg')

# Convert it to grayscale

query\_img\_bw = cv2.cvtColor(query\_img,cv2.COLOR\_BGR2GRAY)

train\_img\_bw = cv2.cvtColor(train\_img, cv2.COLOR\_BGR2GRAY)

# Initialize the ORB detector algorithm

orb = cv2.ORB\_create()

# Now detect the keypoints and compute

# the descriptors for the query image

# and train image

queryKeypoints, queryDescriptors=orb.detectAndCompute(query\_img\_bw,None)

trainKeypoints, trainDescriptors = orb.detectAndCompute(train\_img\_bw,None)

# Initialize the Matcher for matching

# the keypoints and then match the

# keypoints

matcher = cv2.BFMatcher()

matches = matcher.match(queryDescriptors,trainDescriptors)

# draw the matches to the final image

# containing both the images the drawMatches()

# function takes both images and keypoints

# and outputs the matched query image with # its train image

final\_img = cv2.drawMatches(query\_img, queryKeypoints,

train\_img, trainKeypoints, matches[:20],None)

final\_img = cv2.resize(final\_img, (1000,650))

# Show the final image

cv2\_imshow(final\_img)

cv2.waitKey(3000)

**RECOMMNDATION ENGINE**

import pandas as pd

from scipy import sparse

from sklearn.metrics.pairwise import cosine\_similarity

ratings=pd.read\_csv("dataset.csv",index\_col=0)

ratings.fillna(0, inplace=True)

Ratings

def standardize(row):

new\_row = (row - row.mean())/(row.max()-row.min())

return new\_row

df\_std = ratings.apply(standardize).

T print(df\_std)

sparse\_df=sparse.csr\_matrix(df\_std.values)

corrMatrix=pd.DataFrame(cosine\_similarity(sparse\_df),index=ratings.columns,columns=ratings.

CorrMatrix

corrMatrix = ratings.corr(method='pearson')

corrMatrix.head(6)

items = [("Milk",5),(" Rice flour ",1),("Jagger",1)]

similar\_scores = pd.DataFrame()

for products,rating in items:

similar\_scores=similar\_scores.append(get\_similar(products,rating),ignore\_index = True)

similar\_scores.head(10)

similar\_scores.sum().sort\_values(ascending=False)

**Appendix-II**

**Datasheets**

**Details of Paper Publication(Along with Paper)**