SOURCE CODE AND OUTPUT

7.1 CODING

Similarity calculator (psycho graphics based):

Ca.py:

```
# Import Required Modules and library
import pandas as pd
import numpy as np
from scipy.spatial import distance
from pymongo import MongoClient
import os

# Read Dataset CSV File from the disk
df = pd.read_csv('dataset.csv')

#Arrange a CSV File in a order
rows = rows = df['name'].values.tolist()
df['min'] =  (df['o1'] + df['c1'] + df['e1'] + df['a1'] + df['n1']) /5
df['max'] =   (df['o2'] + df['c2'] + df['e2'] + df['a2'] + df['n2']) /5
df.set_index('name', inplace=True)

# Crate a new Empty datafram
newdf = pd.DataFrame(index=rows,columns=rows)

# Loop each record(min max) in dataframe and find eucledean distance of them
for index,rec in df.iterrows():
  now = [rec['min'],rec['max']]

  for sindex,srec in df.iterrows():

    sub = [srec['min'],srec['max']]
    new_value = distance.euclidean(now,sub)
    newdf.at[index,sindex] = new_value

# Store the new Dataframe in mongodb
client = MongoClient(os.getenv("DB_URI"))

db = client['ca_db']
collection = db['ca_scores']

# Convert a dataframe into JSON Format
data = newdf.to_dict()
row = newdf.index.values.tolist()
# Iterate Each row and convert into Object
for key in row:
  newobj = {
```

```
            'name':key,
            'similar':[data[key]]
            }
  # Insert the objects into MongoDB
  rec_id = collection.insert_one(newobj).inserted_id
  # Print the ID of each Inserted Record
  print(rec_id)


print("Completed")
```

(Collaborative filtering based)

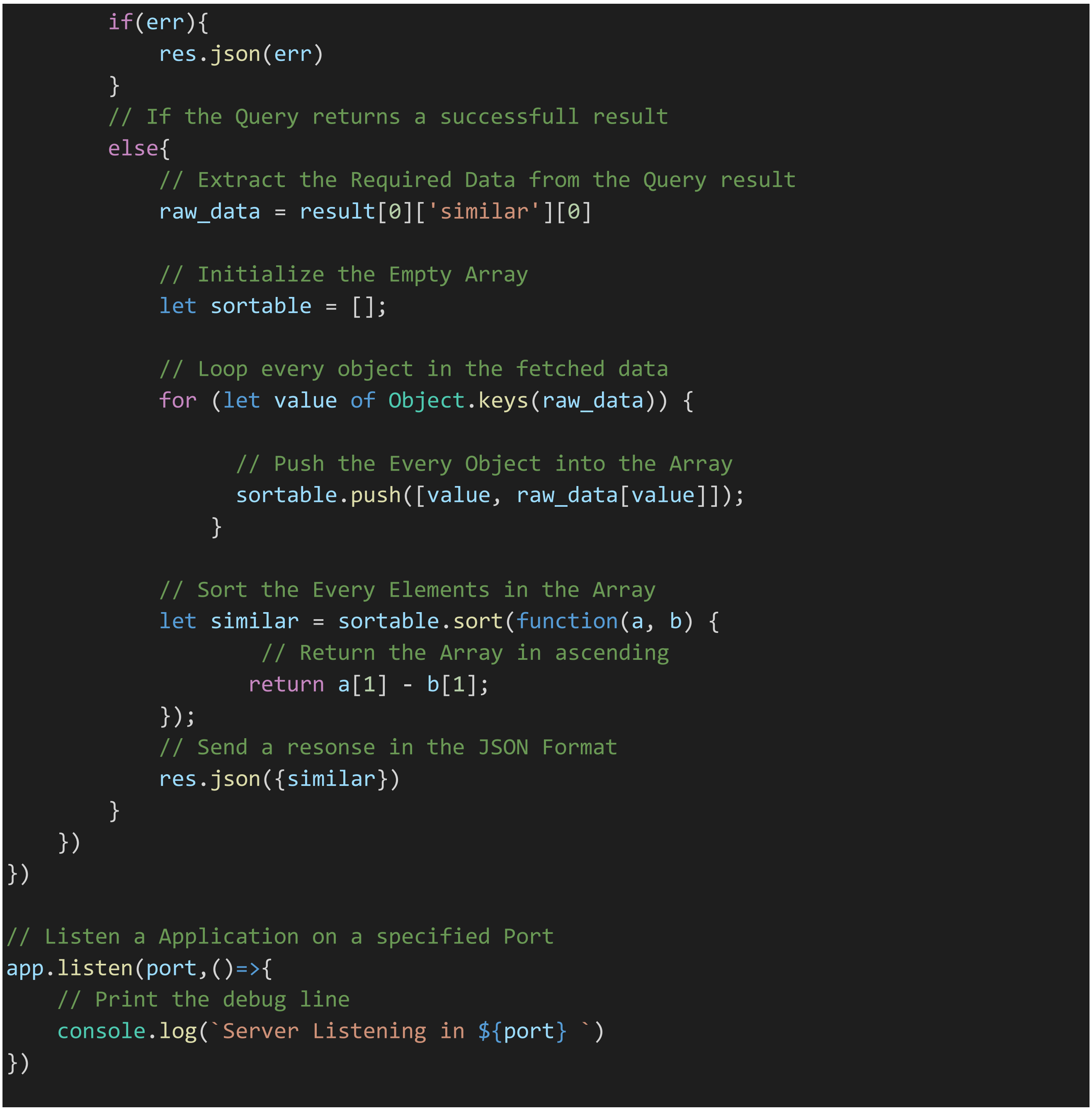
```

import pandas as pd
from scipy import sparse
from sklearn.metrics.pairwise import cosine_similarity

# read the dataset
ratings = pd.read_csv("cl-dataset.csv",index_col=0)
ratings = ratings.fillna(0)


# standardize the data
def standardize(row):
    new_row = (row - row.mean())/(row.max()-row.min())
    return new_row

ratings_std = ratings.apply(standardize)


# Find the cosine similarity
item_similarity = cosine_similarity(ratings_std.T)
print(item_similarity)

# Crete a DataFrame
item_similarity_df =
pd.DataFrame(item_similarity,index=ratings.columns,columns=ratings.columns)


# Get Similar Product
def get_similar(product_name,user_rating):
    similar_score = item_similarity_df[product_name]*(user_rating-2.5)
    similar_score = similar_score.sort_values(ascending=False)
    return similar_score
print(get_similar("Product6",1))
```

```
# User based recommendation
testuser = [("Product1",5),("Product5",1),("Product6",1)]
similar_product = pd.DataFrame()
for product,rating in testuser:
    similar_product =
similar_product.append(get_similar(product,rating),ignore_index=True)
similar_product.head()
similar_product.sum().sort_values(ascending=False)
```

Recommendation System Api

App.js

```
// Load the required Modules
const express = require('express')
const mongoose = require('mongoose')
const dotenv = require('dotenv').config()

// Initialize the Express App Instance
const app = express()

// Assigns a PORT to Web Application
const port = process.env.PORT || 5000

// Get the DataBase URI From the System Environment Variables
let url = process.env.DBURI;

// Establish a Connection to MongoDB
mongoose.connect(url,
    {
        dbName:'ca_db',
        useNewUrlParser: true,
        useUnifiedTopology: true
    }
);

mongoose.set('useCreateIndex', true);

const db = mongoose.connection;
db.on('error', ()=>console.log("DB Connection Error"));
db.once('open',()=>console.log('Connction DB Done'));

// initialize a Schema for MongoDB Collections
const caschema = mongoose.Schema({
    name:{
        type:String,
        required:true,
    },

    similar:{
        type:Array,
        required:true,
    }
});

// Bind the schema to MongoDB Collection
model = mongoose.model('ca_scores', caschema);


// Create Index Route, that send a response about the instance
app.get('/',(req,res)=>{
    res.json({
        Name: "Cognitive-Analytica",
        InstanceId : "7E3AL83Z",
        status : "Active"
    })
})

// Get Route for all product simlilarity
app.get("/all",(req,res)=>{
    // select the all record
    model.find({},(err,result)=>{
        if(err){
            // send a errorr esponse
            res.json({
                status:"fail",
                data:err
            })
        }
        else{
            // Get a success response
            res.json({
                status:"success",
                data:result
            })
        }
    })
})
// Initiallize a End point that taking a product name on the query
app.get('/item/:productname',(req,res)=>{
    // fetch the Product name parameter from request object
    let query = req.params.productname

    // Query the Database and get the result
    model.find({"name":query},(err,result)=>{
        // If the Query returns error
```



```
        if(err){
            res.json(err)
        }
        // If the Query returns a successfull result
        else{
            // Extract the Required Data from the Query result
            raw_data = result[0]['similar'][0]

            // Initialize the Empty Array
            let sortable = [];

            // Loop every object in the fetched data
            for (let value of Object.keys(raw_data)) {

                  // Push the Every Object into the Array
                  sortable.push([value, raw_data[value]]);
                }

            // Sort the Every Elements in the Array
            let similar = sortable.sort(function(a, b) {
                    // Return the Array in ascending
                   return a[1] - b[1];
            });
            // Send a resonse in the JSON Format
            res.json({similar})
        }
    })
})

// Listen a Application on a specified Port
app.listen(port,()=>{
    // Print the debug line
    console.log(`Server Listening in ${port} `)
})
```

Sample Data Set:

name,o1,o2,c1,c2,e1,e2,a1,a2,n1,n2

product0,4,7,3,7,2,7,2,5,3,8

product1,3,5,2,2,1,7,1,2,3,9

product2,3,5,2,2,1,6,4,8,4,6

product3,4,5,1,3,4,6,2,3,3,3

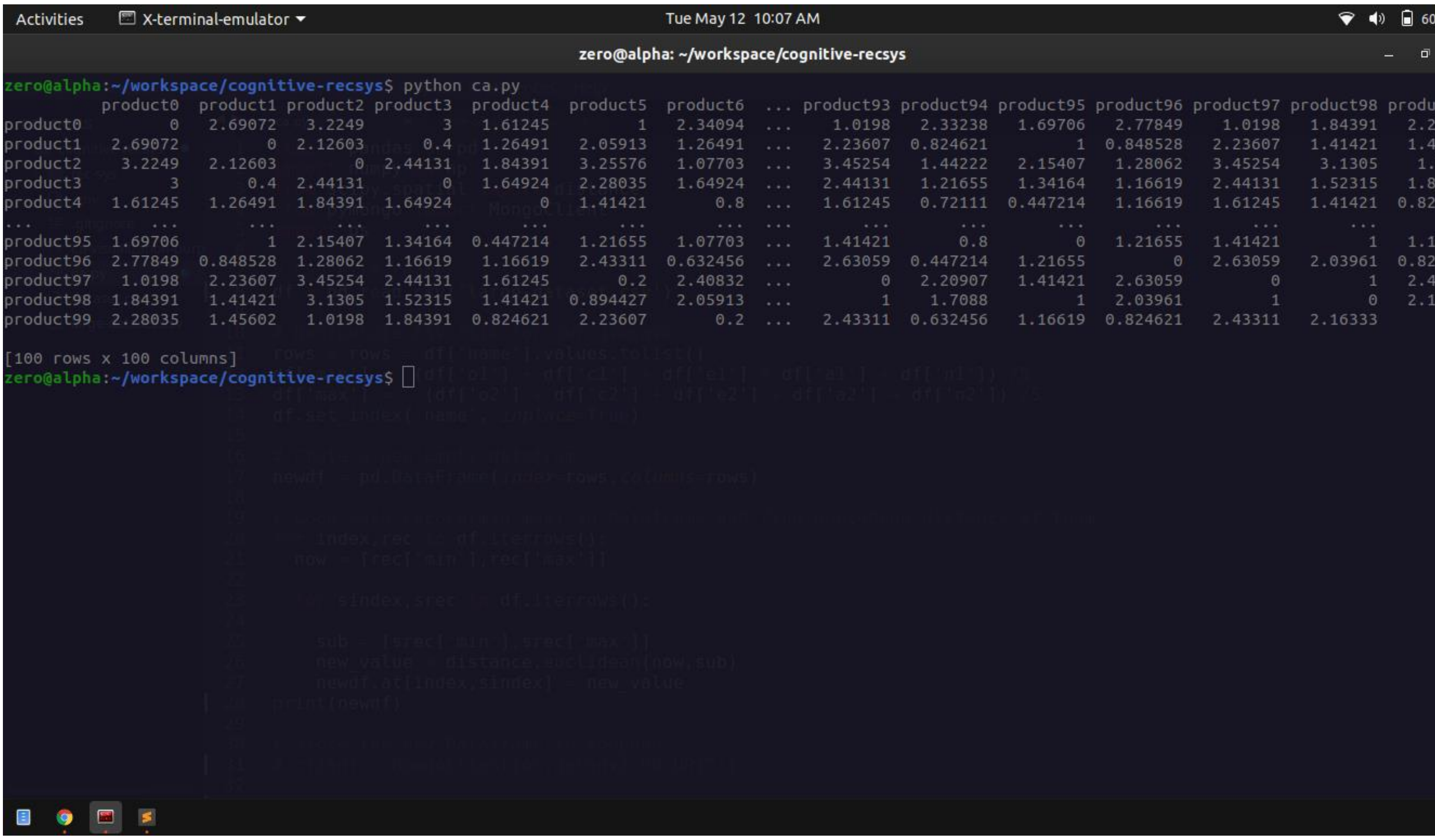
product4,4,5,2,7,1,5,2,3,3,5

Dependencies And Run Scripts:

```
      {

  "name": "cognitive-analytica-recommendation-system",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  },
  "author": "Naveen Micheal",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "mongodb": "^3.5.5",
    "mongoose": "^5.9.7"
  }
}
```

7.2 OUTPUT

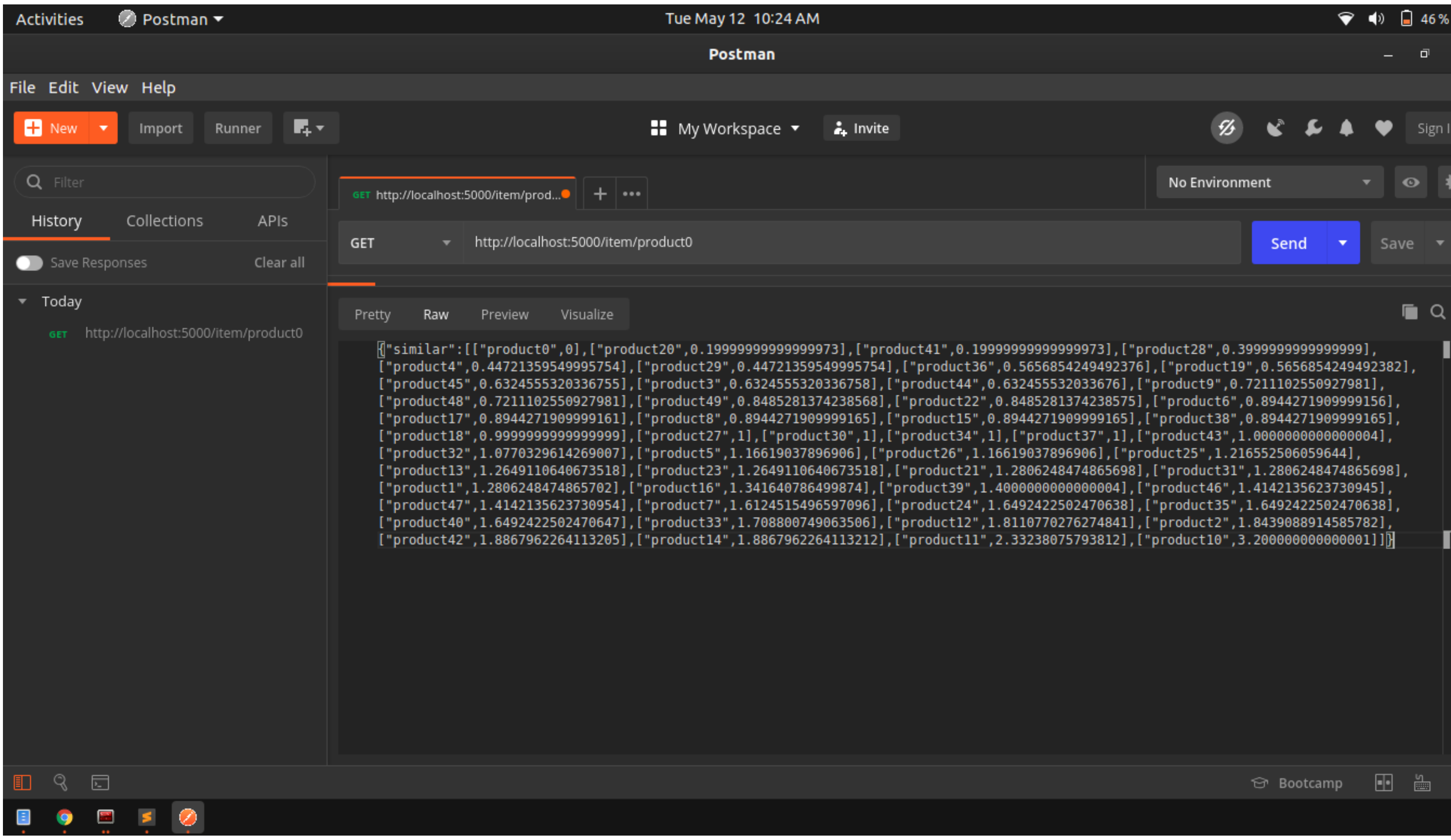
50 products psychographics score calculation:



```
zero@alpha:~/workspace/cognitive-recsys$ python ca.py
           product0  product1 product2 product3  product4  product5  product6  ... product93 product94 product95 product96 product97 product98 produ
product0          0   2.69072   3.2249        3   1.61245         1   2.34094  ...    1.0198   2.33238   1.69706   2.77849    1.0198   1.84391   2.2
product1    2.69072         0  2.12603      0.4   1.26491   2.05913   1.26491  ...   2.23607  0.824621         1  0.848528   2.23607   1.41421   1.4
product2     3.2249   2.12603        0  2.44131   1.84391   3.25576   1.07703  ...   3.45254   1.44222   2.15407   1.28062   3.45254    3.1305   1.
product3          3       0.4  2.44131        0   1.64924   2.28035   1.64924  ...   2.44131   1.21655   1.34164   1.16619   2.44131   1.52315   1.8
product4    1.61245   1.26491  1.84391  1.64924         0   1.41421       0.8  ...   1.61245   0.72111  0.447214   1.16619   1.61245   1.41421  0.82
...             ...       ...      ...      ...       ...       ...       ...  ...       ...       ...       ...       ...       ...       ...
product95   1.69706         1  2.15407  1.34164  0.447214   1.21655   1.07703  ...   1.41421       0.8         0   1.21655   1.41421         1   1.1
product96   2.77849  0.848528  1.28062  1.16619   1.16619   2.43311  0.632456  ...   2.63059  0.447214   1.21655         0   2.63059   2.03961  0.82
product97    1.0198   2.23607  3.45254  2.44131   1.61245       0.2   2.40832  ...         0   2.20907   1.41421   2.63059         0         1   2.4
product98   1.84391   1.41421   3.1305  1.52315   1.41421  0.894427   2.05913  ...         1    1.7088         1   2.03961         1         0   2.1
product99   2.28035   1.45602   1.0198  1.84391  0.824621   2.23607       0.2  ...   2.43311  0.632456   1.16619  0.824621   2.43311   2.16333

[100 rows x 100 columns]
zero@alpha:~/workspace/cognitive-recsys$ 
```

Description:

Here the python application reads the data sets and produce the similarity matrix.



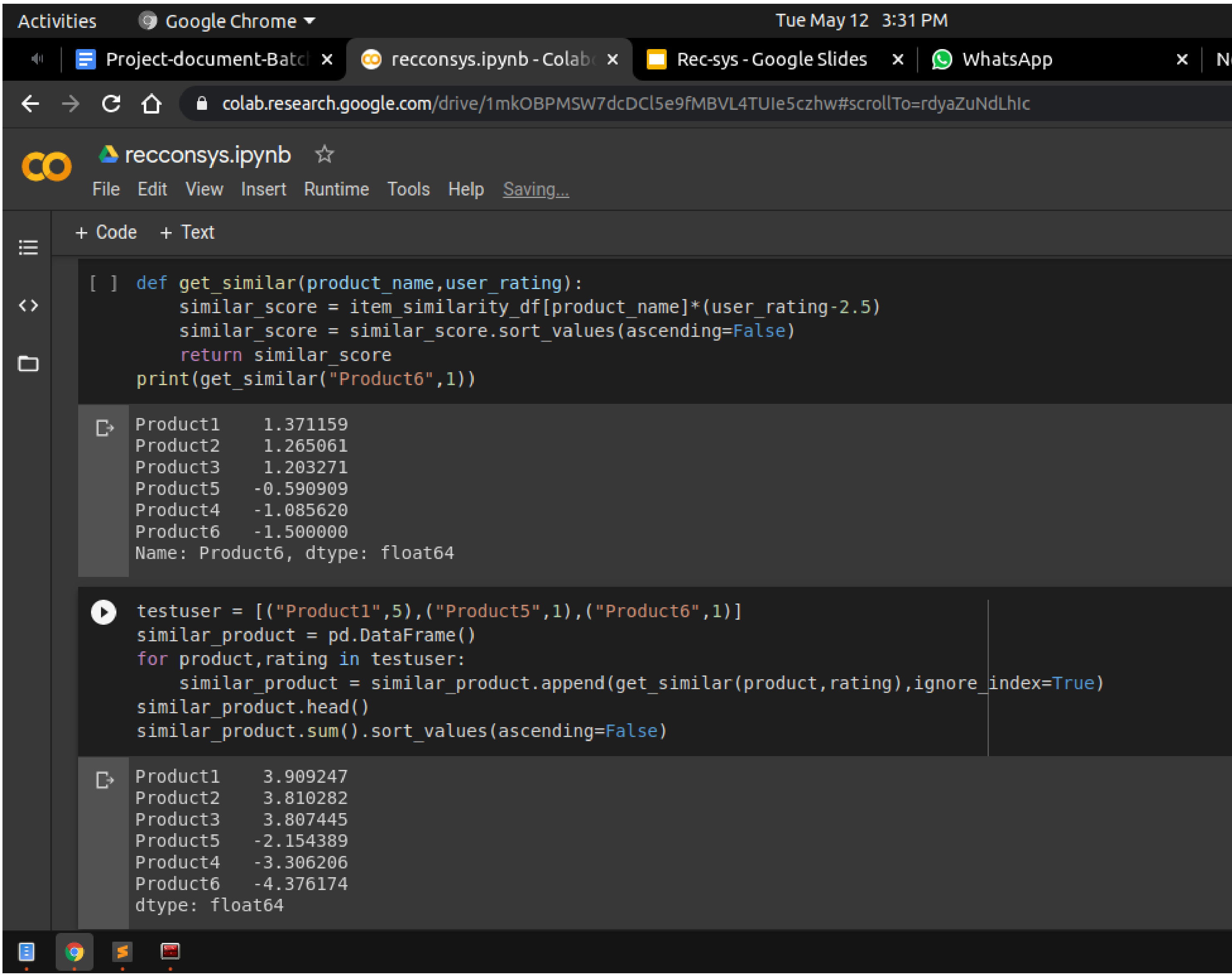
GET http://localhost:5000/item/product0

```
{"similar":[["product0",0],["product20",0.19999999999999973],["product41",0.19999999999999973],["product28",0.3999999999999999],
["product4",0.44721359549995754],["product29",0.44721359549995754],["product36",0.5656854249492376],["product19",0.5656854249492382],
["product45",0.6324555320336755],["product3",0.6324555320336758],["product44",0.632455532033676],["product9",0.7211102550927981],
["product48",0.7211102550927981],["product49",0.8485281374238568],["product22",0.8485281374238575],["product6",0.8944271909999156],
["product17",0.8944271909999161],["product8",0.8944271909999165],["product15",0.8944271909999165],["product38",0.8944271909999165],
["product18",0.9999999999999999],["product27",1],["product30",1],["product34",1],["product37",1],["product43",1.0000000000000004],
["product32",1.0770329614269007],["product5",1.16619037896906],["product26",1.16619037896906],["product25",1.2165525060596444],
["product13",1.2649110640673518],["product23",1.2649110640673518],["product21",1.2806248474865698],["product31",1.2806248474865698],
["product1",1.2806248474865702],["product16",1.341640786499874],["product39",1.4000000000000004],["product46",1.4142135623730945],
["product47",1.4142135623730954],["product7",1.6124515496597096],["product24",1.6492422502470638],["product35",1.6492422502470638],
["product40",1.6492422502470647],["product33",1.708800749063506],["product12",1.8110770276274841],["product2",1.8439088914585782],
["product42",1.8867962264113205],["product14",1.8867962264113212],["product11",2.33238075793812],["product10",3.2000000000000001]]}
```

Description:

Here the API will fetch the similar products of the given product which is passed in URL parameters or input. The fetched results are converted to JSON array of objects then return to the user

Collaborative filtering output:



```
def get_similar(product_name,user_rating):
    similar_score = item_similarity_df[product_name]*(user_rating-2.5)
    similar_score = similar_score.sort_values(ascending=False)
    return similar_score
print(get_similar("Product6",1))
```

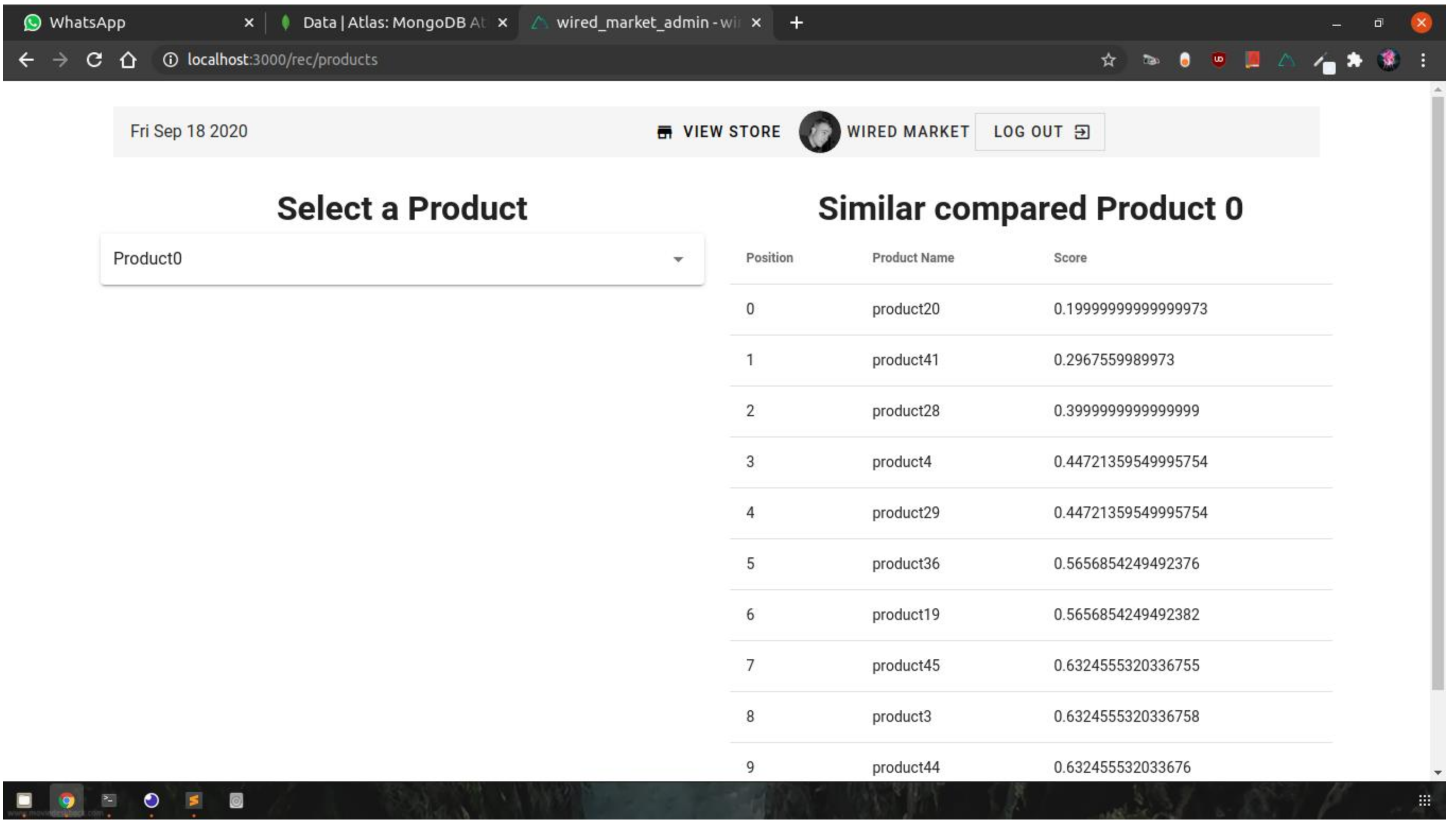
```
Product1    1.371159
Product2    1.265061
Product3    1.203271
Product5   -0.590909
Product4   -1.085620
Product6   -1.500000
Name: Product6, dtype: float64
```

```
testuser = [("Product1",5),("Product5",1),("Product6",1)]
similar_product = pd.DataFrame()
for product,rating in testuser:
    similar_product = similar_product.append(get_similar(product,rating),ignore_index=True)
similar_product.head()
similar_product.sum().sort_values(ascending=False)
```

```
Product1    3.909247
Product2    3.810282
Product3    3.807445
Product5   -2.154389
Product4   -3.306206
Product6   -4.376174
dtype: float64
```

Description:

Here the similarity score has been calculated by using collaborative based filtering algorithm.



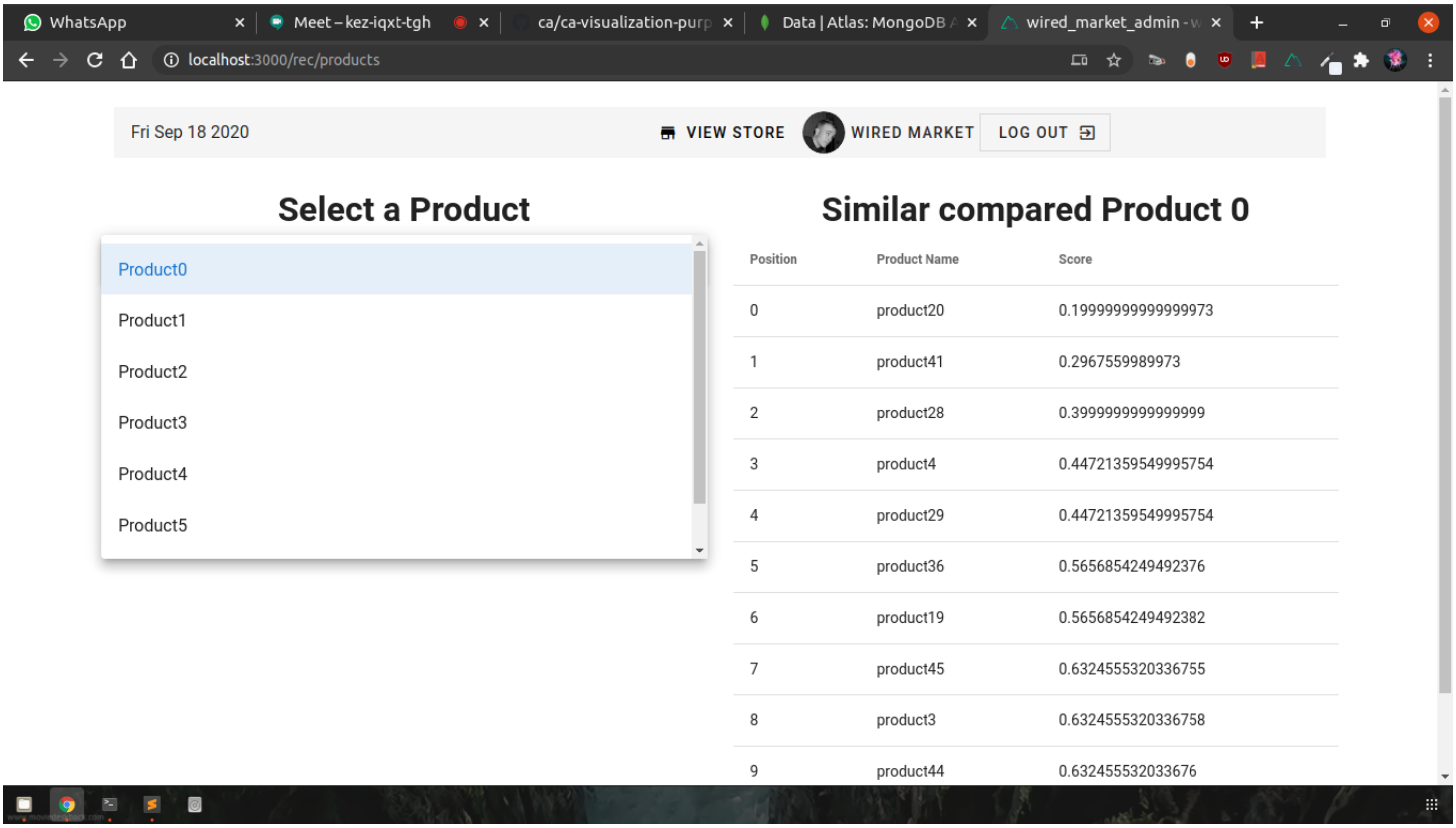
Fri Sep 18 2020 VIEW STORE WIRED MARKET LOG OUT

Select a Product

Product0

Similar compared Product 0

Position	Product Name	Score
0	product20	0.19999999999999973
1	product41	0.2967559989973
2	product28	0.3999999999999999
3	product4	0.44721359549995754
4	product29	0.44721359549995754
5	product36	0.5656854249492376
6	product19	0.5656854249492382
7	product45	0.6324555320336755
8	product3	0.6324555320336758
9	product44	0.632455532033676



Fri Sep 18 2020 VIEW STORE WIRED MARKET LOG OUT

Select a Product

- Product0
- Product1
- Product2
- Product3
- Product4
- Product5

Similar compared Product 0

Position	Product Name	Score
0	product20	0.19999999999999973
1	product41	0.2967559989973
2	product28	0.3999999999999999
3	product4	0.44721359549995754
4	product29	0.44721359549995754
5	product36	0.5656854249492376
6	product19	0.5656854249492382
7	product45	0.6324555320336755
8	product3	0.6324555320336758
9	product44	0.632455532033676