

CHAPTER 1

INTRODUCTION

Recommender systems are so commonplace now that many of us use them without even knowing it. Because we can't possibly look through all the products or content on a website, a recommendation system plays an important role in helping us have a better user experience, while also exposing us to more inventories we might not discover otherwise.

Some examples of recommender systems in action include product recommendations on Amazon, Netflix suggestions for movies and TV shows in your feed, recommended videos on YouTube, music on Spotify, the Facebook newsfeed and Google Ads. Important component of any of these systems is the recommender function, which takes information about the user and predicts the rating that user might assign to a product, for example. Predicting user ratings, even before the user has actually provided one, makes recommender systems a powerful tool.

Also the recommendation system acts as a filtering mechanism, when in situation the hundreds of products in the options list. For efficient filtering it needs to group the items in bases of relationship between them. It's optimal to find product to product score by using the Psychographics technics

Types:

- 1) Content Based Recommendation (focused in Items Objects)
- 2) Collaborative filtering Recommendation (focused in relations)

Content Based Recommendation:

Content-based filtering is one of the common methods in building recommendation systems. Systems that recommend an item to a user based upon a description of the item and a profile of the user's interests. Content-based recommendation systems may be used in a variety of domains ranging from recommending web pages, news articles, restaurants, television programs, and items for sale. Although the details of various systems differ, content-based recommendation systems share in common a means for describing the items that may be recommended, a means for creating a profile of the user that describes the types of items the user likes, and a means of comparing items to the user profile to determine what to recommend. The profile is often created and updated automatically in response to feedback on the desirability of items that have been presented to the user.

Advantages:

1. Content representations are varied and they open up the options to use different approaches like: text processing techniques, the use of semantic information, inferences, etc...
2. It is easy to make a more transparent system: we use the same content to explain the recommendations.

Disadvantages:

1. **Limited content analysis:** if the content does not contain enough information to discriminate the items precisely, the recommendation will be not precise at the end.
2. **Over-specialization:** content-based method provides a limited degree of novelty, since it has to match up the features of profile and items. A totally perfect content-based filtering may suggest nothing “surprised.”

Collaborative filtering:

Recommendation for a user A is then made by looking at the users that are most similar to A in this sense, and recommending items that these users like. The process of identifying similar users, products and recommending what similar users like is called collaborative filtering

Advantages:**No domain knowledge necessary**

The collaborative filtering is not fully dependent on the domain knowledge about products, it will learn about the products by the interaction between the users and items. It focused to find the similar data from the products

Serendipity

The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

Great starting point

To some extent, the system needs only the feedback matrix to train a matrix factorization model. In particular, the system doesn't need contextual features. In practice, this can be used as one of multiple candidate generators.

Disadvantages:

Cannot handle fresh items

The prediction of the model for a given (user, item) pair is the dot product of the corresponding embedding. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem. However, the following techniques can address the cold-start problem to some extent

High Resource conception

That process requires a lot of computation and CPU resources. whenever the application's data size increases the required computation is also increased equally, It's not suitable for startup companies capital. Most of the recommender systems built on the machine learning algorithms, these algorithms required numerous operation iterations to produce a successful recommender system.

Scalability:

In a software application, the scalability features is vital one, specifically when the software dealing with data, because whenever the enterprise gets big, they processed data size also increased, so that they used softwares also needs to handle large datasets at any time

Modularity:

Always that developed software will not be in the constant architecture, the enterprise needs business structure changing constantly, so further enhancement and feature upgrade is often time necessary in the software development to meet the evolving market, so the modularity of the application is important constraints to consider. the complex software needs to be broken into multiple pieces to work with easily. and the development time will not affect the enterprise operation. The recommender system must be in the modular architecture and easy to work with.

Complex:

The traditional recommender system mostly built on the machine learning algorithms and needs high intensive data manipulation computations to create a positive decision making system. It's very complex to build it from scratch and require a lot of man resources to bring a new feature or change something.

Recommender system based on psychographic

Big consumer data promises to be a game changer in applied and empirical marketing research. However, investigations of how big data helps inform consumers' psychological aspects have, thus far, only received scant attention. Psychographics has been shown to be a valuable market segmentation path in understanding consumer preferences. Although in the context of e-commerce, as a component of psychographic segmentation, personality has been proven to be effective for prediction of e-commerce user preferences

Psychographic segmentation in predicting users' online purchasing preferences across different product categories in e-commerce by using a data-driven approach has not been fully exploited in recent times, but in theory clearly shows we can build a recommendation system in light weight and also increase the operation when the data increases. The Big Five Factor (BFF) also known as OCEAN model consist of personality traits extraversion (also often spelled extroversion), agreeableness, openness, conscientiousness, and neuroticism, these five thing enough to identify a user's personality

In the context of the recommender system, we can't always predict the user's personality all time, with a small amount of known data, but we can find the product that can match with the people who have a particular personality level. using that we can make a group of product bases of personality type. the similarity of a product chosen from that

Combining other market segmentation techniques like demographic and geographics based models bring even more positive decision making systems, also that system needs to support the existing recommender system like collaborative based filtering techniques whenever it has been required.

PROJECT OBJECTIVES:

The major objective of the project is to develop a recommender system that can be easy to scalable from small operations to enterprise needs. We focused on building applications that favor micro service architecture by separating the all functional units in a distributed manner, for example, the high intensive computation required functions are deployed and managed in the separate machine with a required resource, and low-level computation required multiple components reside in a single system. For resource allocation between the components has been assigned by the requirement loads. The communication between the client and server has been done by REST (Representational State Transfer) APIs, and also the Web Sockets when the specific requirements scenarios.

The major problem of existing recommender systems is highly dependent on a large amount of previous data about the product and user. The lack of data produces inefficient results in the recommendation system.

In our project, the initial data has been calculated by using a method called Psychographics segmentation. Psychographics is a qualitative methodology used to study human psychological attributes. Psychographics have been applied to the study of a person's personalities, values, opinions, attitudes, interests, and lifestyles.

Each personality score has been calculated by the experts in that product marketing department, after that the scores have been reviewed and validated by a group of experts from the various specialized teams by conducting multiple brainstorming sessions. Finally, the application calculates the Psychographics score for each product in the system, and stores it on databases. When it requests to find similar items, it just compares the scores between all other products, and returns the list of products sorted by most close relational scores.

In a situation, if the product recommendation decision did not perform well in real-time, the score has been recalculated again by a group of experts. That operation continuously performed until it shows the positive results

CHAPTER 2

LITERATURE SURVEY

Title: *Content-based filtering for recommendation systems using multi attributes networks - 2017*

Authors: *JieunSon, Seoung BumKim*

Idea: Content based recommendation fully relied on the product attributes. It grouped the items by those attributes. Create similarity models by using the Term frequency and Inverse Document frequency (TF IDF) algorithm

Disadvantage:

1. Requires a lot of domain knowledge. The null attributes causes the unbalanced groupings, hard to scalable when a new attributes added causes a scalability issue
2. Model has limited ability to expand, and TF IDF Algorithm is resource intensive

Title: *Collaborative Filtering Recommendation Algorithm Based on Knowledge Graph - 2018*

Authors: *Ruihui Mu, Xiaoqin Zeng*

Idea: Obtain the similarity model by using filtering the similar items, and grouping in together by the past information. And also predict the null ratings by comparing another similar data

Disadvantages:

1. Past information plays the crucial role in the model. The lack of data predicts the inefficiency results
2. It has a cold start problem. When the new item entered into the system it takes some time to gather enough data

Title:A Collaborative Recommendation System for Online Courses Recommendations

Authors: Raghad Obeidat ; Rehab Duwairi ; Ahmad Al-Aiad

Idea: It recommends an online course for students based on similarities of students' course history. The system employs data mining techniques to discover patterns between courses to pick similar courses that match user's history

Disadvantages:

1. The recommendation system requires a user to enroll in some courses and need to show signs of positive engagements.
2. Based on the user's engagement in a particular course type, the recommendation system makes a decision.
3. The insufficient data and engagement durations will produce insufficient results

Title: Movie Recommendation System Using Semi-Supervised Learning

Authors: Sushmita Roy ; Mahendra Sharma ; Santosh Kumar Singh

Idea: Recommendation by using machine learning algorithms such as a content-based filtering approach, a collaborative-based filtering approach.

Disadvantages: Highly computation required and hard to scalable

Title: A Comparative Study of Psychographic Segmentation Based on an Online Review Enhanced Recommender System

Authors Hui Liu,Yinghui Huang, Zichao Wang, Kai Liu, Xiangen Hu,and Weijun Wang

Idea Identify the psychographics personality variables and use cases in the recommender system

Disadvantages: Absence of Practical Application and no information about the issues

2.1 EXISTING SYSTEM:

The explosive growth of the world-wide-web and the emergence of e-commerce have led to the development of recommender systems. Recommender systems are personalized information filtering used to identify a set of items that will be of interest to a certain user.

A general recommendation system study aims to increase the predictive accuracy of the recommendation system. However, a study suggests that even if the recommendation system is accurate, when customers were recommended with the same item every time, their satisfaction or reliability is likely to decrease

The quantification of the information provides the opportunity to build a decision making system, that system can be used to make a recommender system more effectively. The existing approaches like content based and collaborative based are dependent on the past information about the product, and the user.

The content based recommendation system consists of Item Representation that Extract some feature attributes for each item, that is, the description operation of structured items. The corresponding processing procedure is called: Content Analyzer, and Profile Learning Use the item feature data that a user likes (dislikes) in the past to learn the user's favorite profile and Recommendation Generation: By comparing the user profile obtained in the previous step to the feature of the feature item, the user is recommended for a set of the most relevant items; the corresponding process is called: Filtering Component (filter component)

collaborative filtering models today which can be generally split into two classes: user- and item-based collaborative filtering. In either scenario, one builds a similarity matrix. For user-based collaborative filtering, the user-similarity matrix will consist of some distance metric that measures the similarity between any two pairs of users. Likewise, the item-similarity matrix will measure the similarity between any two pairs of items A common distance metric is cosine similarity. The metric can be thought of geometrically if one treats a given user's (item's) row (column) of the ratings matrix as a vector. For user-based collaborative filtering, two users' similarity is measured as the cosine of the angle between the two users' vectors.

when it has enough data to predict the recommendation, it tries to fill the blank values in the matrix. that process of predication not a one time operation it continually do that until it shows the positive feedback, the system gets more and more accurate when the requires extra data about the user and product.

DRAW BACKS:

1. It works well, when it have a enough data to predict, when a new product or a user enters into a system, the system can't serve the precised recommendation because it needs more information to positive results
2. Lack of information produce inefficient results, at the starting time it just randomly provides the content to the users, there is a chance the user just leaves the system if they experience the unpleasant experience on the system
3. Some Algorithms require a lot of domain knowledge. The null attributes causes the unbalanced groupings, hard to scalable when a new attributes added causes a scalability issue
4. The cold-start problem arises when items added to the catalogue have either none or very little interactions. This constitutes a problem mainly for collaborative filtering algorithms due to the fact that they rely on the item's interactions to make recommendations
5. It comes with numerous components to make recommendation system, when the input data level increased it start to show performance degradation, and hard to scalable
6. The multiple mathematical operations has been performed multiple times on the huge amount of data that huge Computation Intensive Algorithms requires more computational resources
7. The traditional recommender system mostly built on the machine learning algorithms and needs high intensive date manipulation computations to create a positive decision making system.
8. because of it requires more computation resources it will be in high cost, it may not be suitable for starter company's capital
9. Hard to architect the application and upgrade the specific components, lack of modularity, requires a lot of human resources
10. It can't predict the changing user preferences, the humans are always changing depends upon the nature, but the existing can't aware of immediate changes,

2.2 PROPOSED SYSTEM:

By doing product to product based collaborative filtering and combining them with **Psychographics Market segmentation** techniques fix the problems and existing system, and also the proposed system following micro service architecture with less computation requirements and easy to scalable.

it plays a promising role in predicting online user preferences, There are two main reasons. First, data collection for such research typically requires online customers to complete lengthy surveys and. Second, acquiring the abstract psychographic characteristics of online products through the mandatory selection of questionnaires. Psychology and marketing research indicate that human language reflects psychological characteristics. Several researchers have found that variations in word use embedded in writing such as blogs, essays, and tweets can predict aspects of personality, thinking style, social connections, and purchase decisions

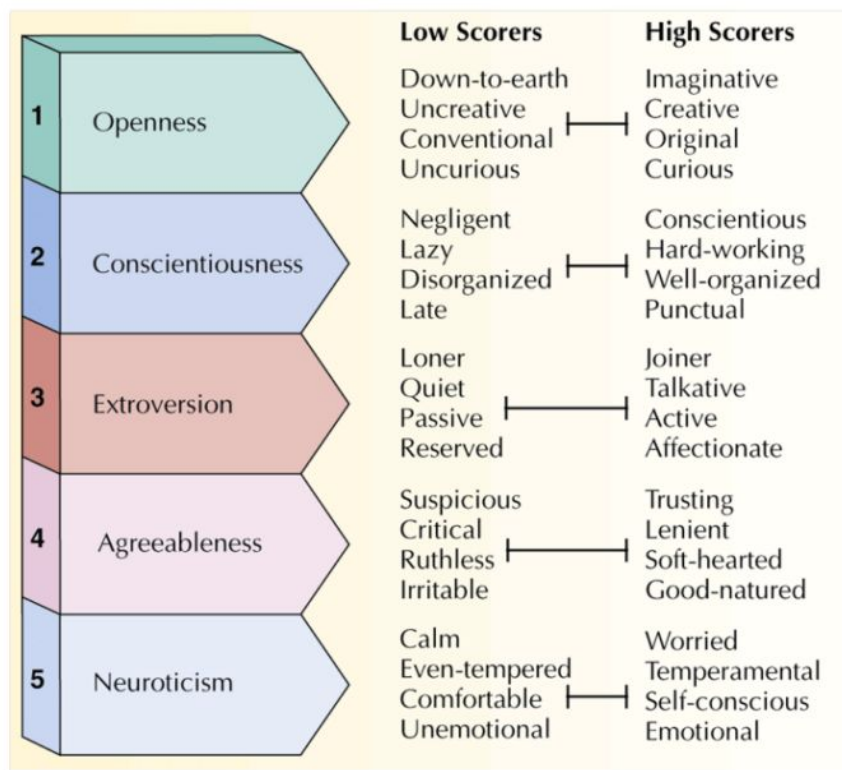
ADVANTAGES:

1. Doesn't depend on lots of data, when a product initially enters into system the group of experts calculated the score based on the psychographics personality traits
2. The Dependent of the Initial data has been reduced the cold start problem can't arise, at the start
3. Minimal computation is enough to build a recommender system, the computation only performed when a new product enters into system,
4. It's easy to Scalable, whenever the input data is huge the only that time computation required a lot
5. it's possible to provide a recommendation as s as system like a Software as a service from the cloud by the their party vendors
6. When a Negative decision making found it is easy to recompute the score of the product
7. Easy to troubleshooting and improving the performance
8. Compatibility to all the systems, and extensibility when its required

Psychographics:

Psychographics is a qualitative methodology used to describe traits of humans on psychological attributes. Psychographics have been applied to the study of personality, values, opinions, attitudes, interests, and lifestyles.

Demographic information includes gender, age, income, marital status. It explains “who” your buyer is, while psychographics explain “why” they buy.



Big Five personality traits:

In psychological trait theory, the Big Five personality traits, also known as the five-factor model (FFM) and the OCEAN model.

The theory identifies five factors:

- Openness
Curious, Imaginative, Original
- Conscientiousness
Hard-working, Punctual, well organized

- Extroversion
Talkative, Joiner, Active, Affectionate
- Agreeableness
Trusting, Self-heated good natured
- Neuroticism
Worried, Emotional, Self-conscious

1. Openness

Openness is a general appreciation for art, emotion, adventure, unusual ideas, imagination, curiosity, and variety of experience. People who are open to experience are intellectually curious, open to emotion, sensitive to beauty and willing to try new things.

People who achieve a high score in the Openness trait tend to be creative or unconventional and open to new experiences and ideas. Imagination and insight are fundamental to Open people who are willing to be vulnerable, try new things – and who will almost always think outside the box.

If someone achieves a low Openness score they are more likely to be focused on routine, as opposed to variety, and to find abstract thought challenging.

2. Conscientiousness

Conscientiousness is a tendency to display self-discipline, act dutifully, and strive for achievement against measures or outside expectations. It is related to the way in which people control, regulate, and direct their impulses

Procrastinators and those who prefer an unstructured approach to life don't score highly for Conscientiousness. Instead, this is a person who has a tendency to control impulses, demonstrate impressive self-discipline and who prefers to follow a plan or schedule.

A high Conscientiousness score indicates a methodical individual who is persistent, thorough, predictable and a planner.

3. Extroversion

Extroversion is characterized by breadth of activities (as opposed to depth), urgency from external activity/situations, and energy creation from external means. The trait is marked by pronounced engagement with the external world.

Most extroverts draw their energy from being around other people while introverts need to spend time alone to be able to reboot and recharge. People who score highly for Extroversion seek out social contact, are assertive and talkative, energetic and outgoing. They favour action over contemplation and are often keen to be the centre of attention.

A low score in Extroversion is usually associated with someone who is more introspective, who prefers their own company and who may experience social anxiety.

4. Agreeableness

The agreeableness trait reflects individual differences in general concern for social harmony. Agreeable individuals value getting along with others. They are generally considerate, kind, generous, trusting and trustworthy, helpful, and willing to compromise their interests with others.

Empathy and compassion define those who have a high Agreeableness score.

Getting along well with others and being caring, cooperative and sympathetic are also usually traits that are found in people who are high scoring for Agreeableness. Individuals who are strongly defined by this trait exhibit kindness, loyalty, patience and trust.

5. Neuroticism

Neuroticism is the tendency to experience negative emotions, such as anger, anxiety, or depression.[46] It is sometimes called emotional instability, or is reversed and referred to as emotional stability. According to Eysenck's (1967) theory of personality, neuroticism is interlinked with low tolerance for stress or aversive stimuli. All of us have some degree of emotional sensitivity and the Neuroticism trait is designed to identify how much someone worries.

Those who score low for Neuroticism tend to be more relaxed, secure and emotionally grounded.

High scorers will regularly experience awkwardness, pessimism, jealousy, fear, anxiety, self criticism and a lack of confidence.

CHAPTER 3

PROJECT DESCRIPTION

3.1 PROBLEM DEFINITION

Today, several recommender systems have been developed for different domains however, these are not precise enough to fulfil the information needs of users. Therefore, it is necessary to build high quality recommender systems. In designing such recommenders, designers face several issues and challenges that need proper attention. It's necessary to investigate and report the current trends, issues, challenges, and research opportunities in developing high-quality recommender systems. If properly followed, these issues and challenges will introduce new research avenues and the goal towards fine-tuned and high-quality recommender systems can be achieved.

3.2 METHODOLOGIES

3.2.1 MODULE NAME

1. Obtain and Optimize the dataset for efficient processing
2. Find the similarity matrix
3. Recommend that item to user
4. Feedback collector

3.2.2 MODULE DESCRIPTION

3.2.2.1 OPTIMIZE THE DATASET FOR EFFICIENT PROCESSING

Always the collected Raw datasets needs to be formalized by the process of multiple data normalization techniques. In our project the initial dataset has been made manually by the group of experts who all are expertise in their belonging product marketing and use cases.

The Datasets containing the product and its attribute score in the form of CSV file format. It's necessary to clean or rearrange the dataset for efficient processing and remove the unnecessary values.

To create a psychographic dataset to the product, **by conducting a Brainstorming session** in each Cognitive attribute in Big Five personality traits.

3.2.2.2 FIND THE SIMILARITY MATRIX

Find out the similarity by using two methods

1. Item to item based collaborative filtering

Collaborative filtering based on item-item is much more reliable than content based, because in example the Horror movie reminds a Horror movie after 10 years.

2. Psychographics market segmentation

Psychographics based models focus on the Big Five personality traits. The five factors are represented [by whom?] using the acronyms OCEAN Beneath each proposed global factor, there are a number of correlated and more specific primary factors.

The score has been calculated by the Euclidean distance between the two points. That score has been calculated and stores it on permanent databases the calculation operation needs executed in every time the new product enters into a system

3.2.2.3 RECOMMEND THAT ITEM TO USER

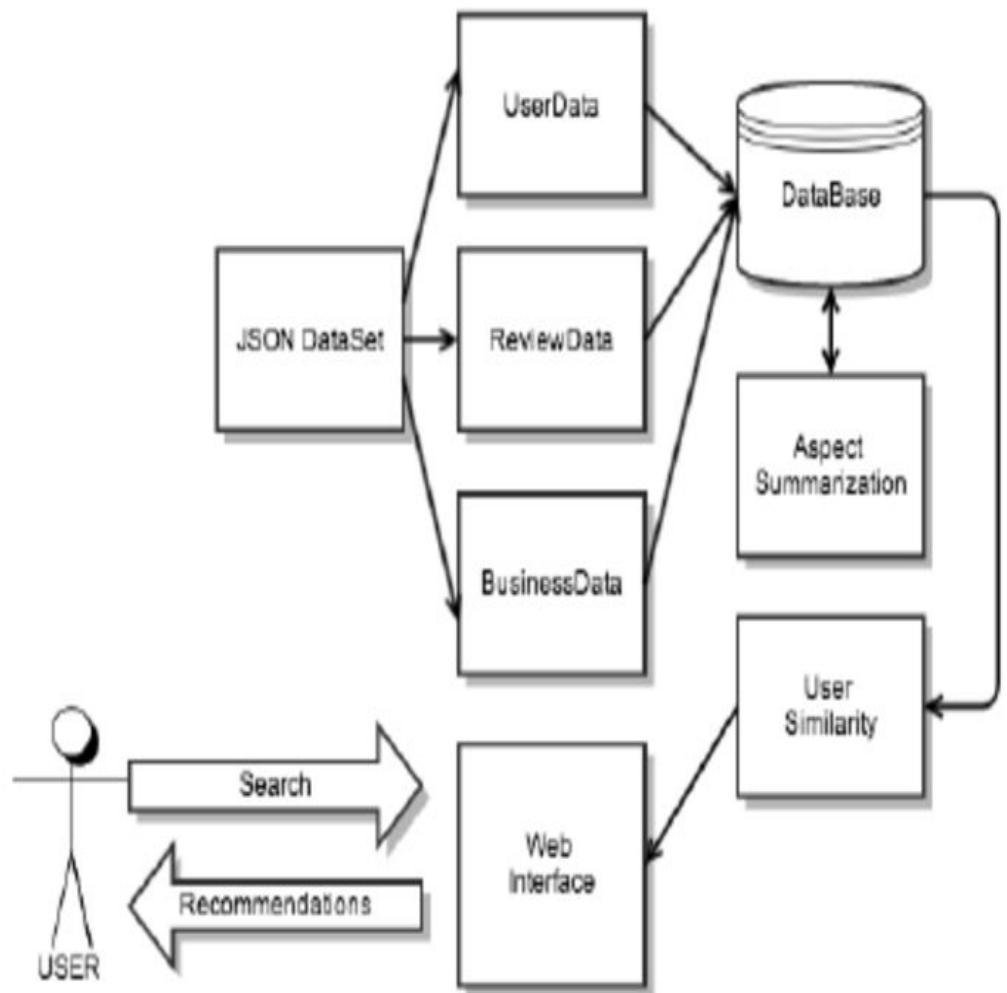
When it comes to recommendation the the other product recommendation scores has been sorted from more similar to higher, and send it to the user

A similar product has been recommended to the user, on the basis of the difference between the score. Lower scores got a high priority. In collaborative approach used a cosine similarity based score

3.2.2.4 FEEDBACK COLLECTOR

The feedback of recommendation has been collected and analyses the accuracy rate. It is responsible to tell the performance of the recommendation system. It gives the opportunity to upgrade or patch the way the approach has been used.

The collaborative based filtering approach has been used in situation when psychographics approach calculated inefficient score

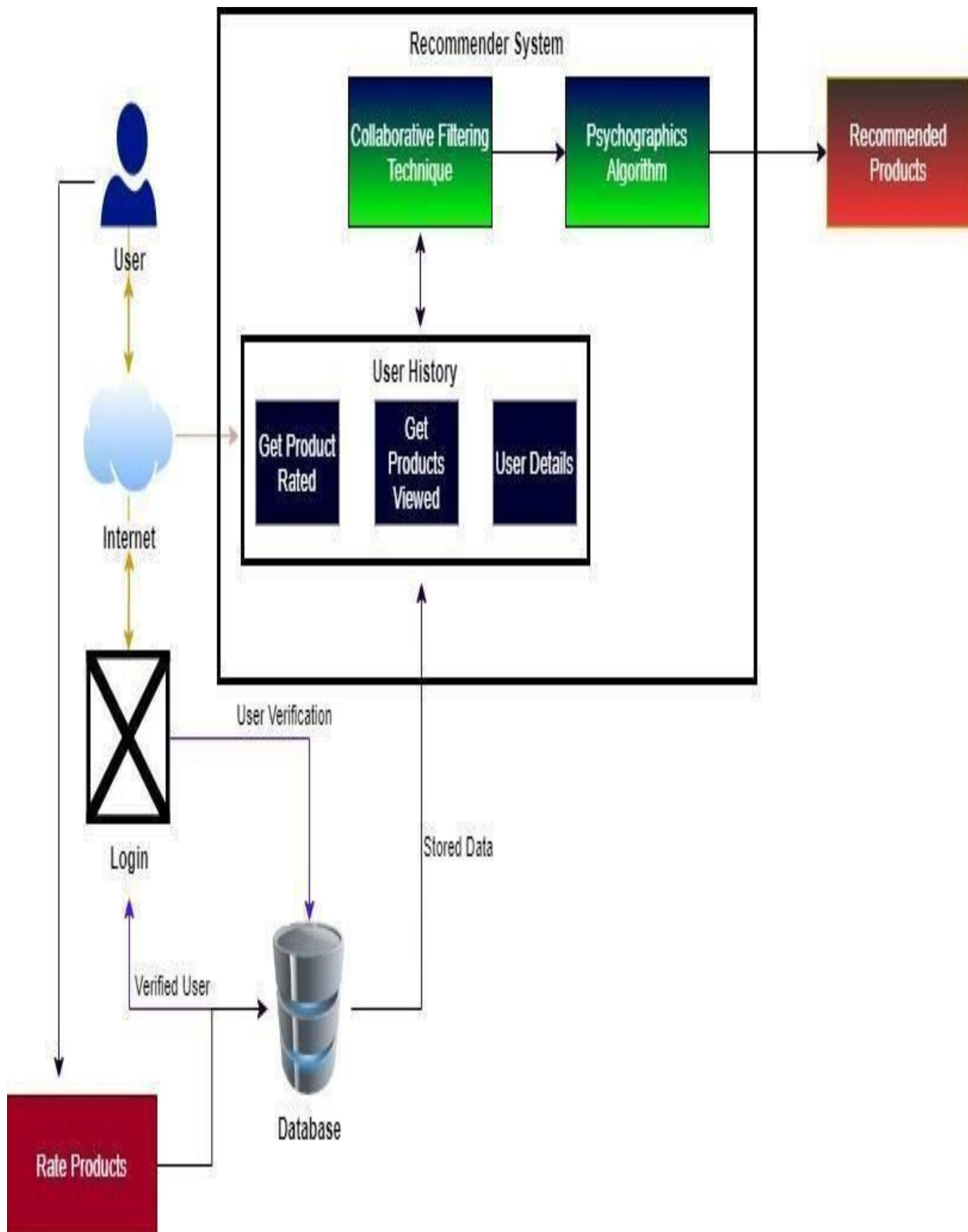


3.3 OVERALL SYSTEM ARCHITECTURE

System design is the process or art of defining the architecture, components, modules, interface, and data for the system to satisfy specified requirements. This chapter deals with various design and function of the system.

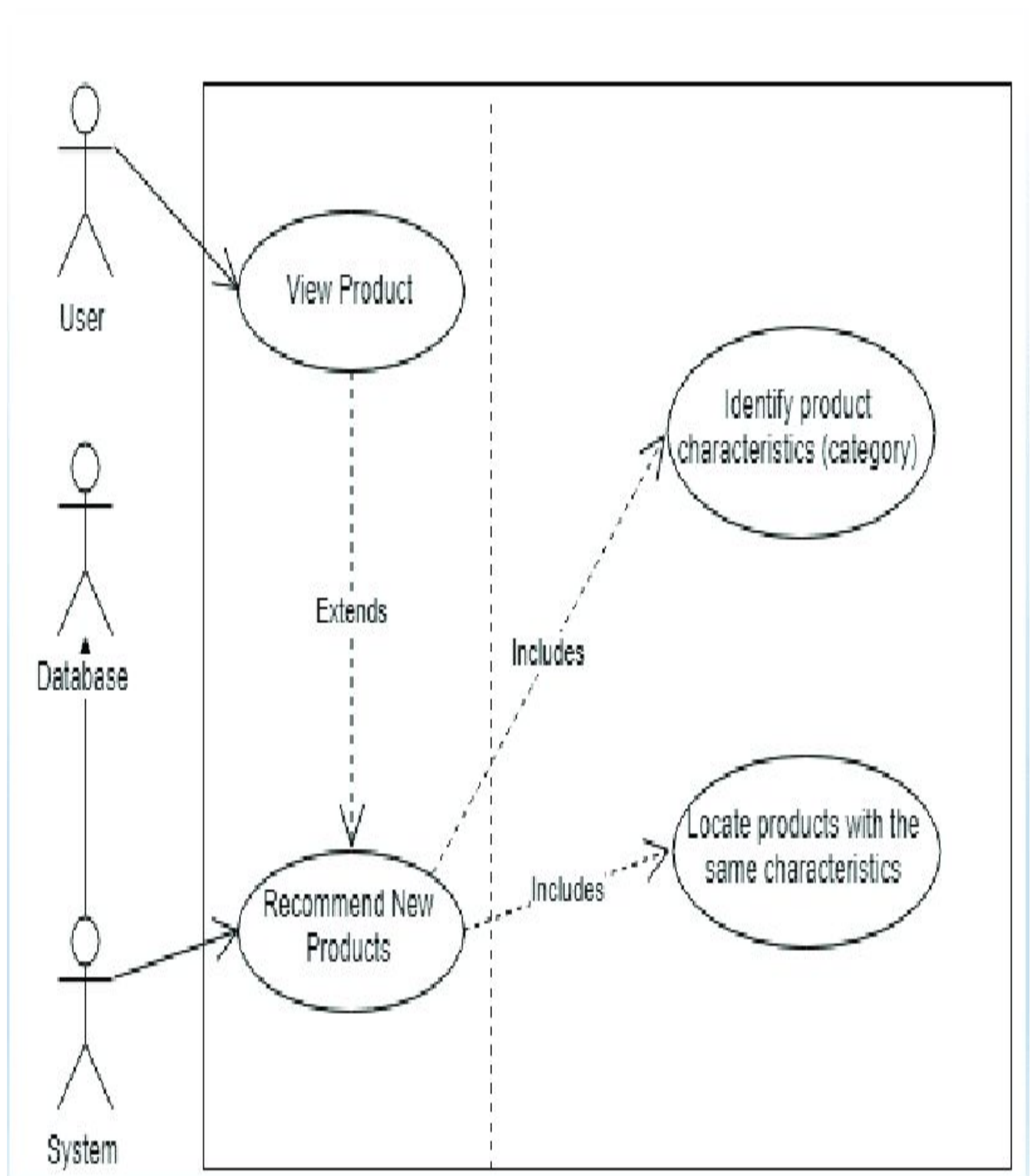
Our System follows the micro service Architecture. So the Score calculation application has a responsible calculate and store the data to the Database

The recommendation program has a responsible to pull back the score from the dB, and recommending to the user

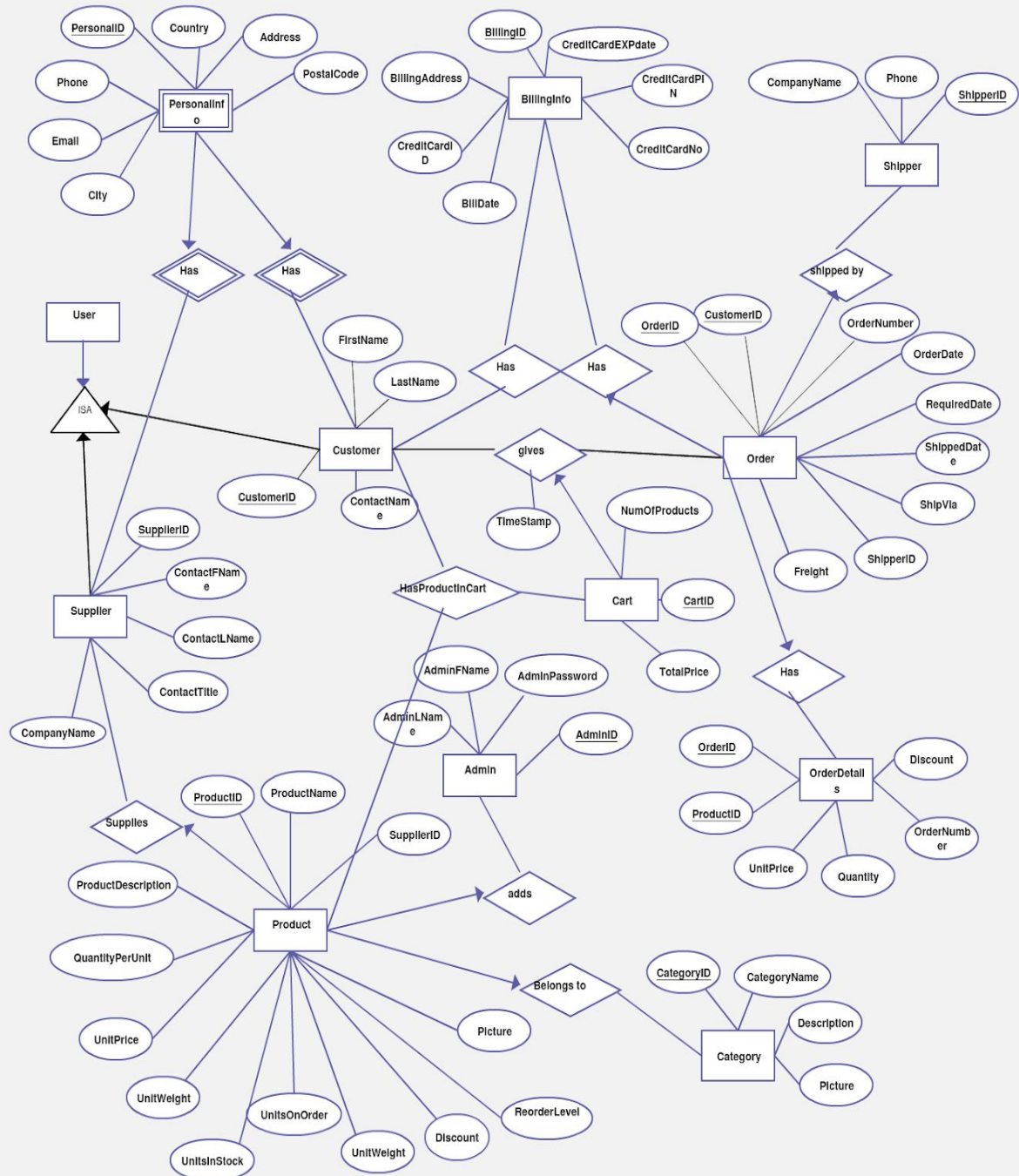


3.4 USE CASE DIAGRAM

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions that some system should or can perform in collaboration with one or more external users of the system.

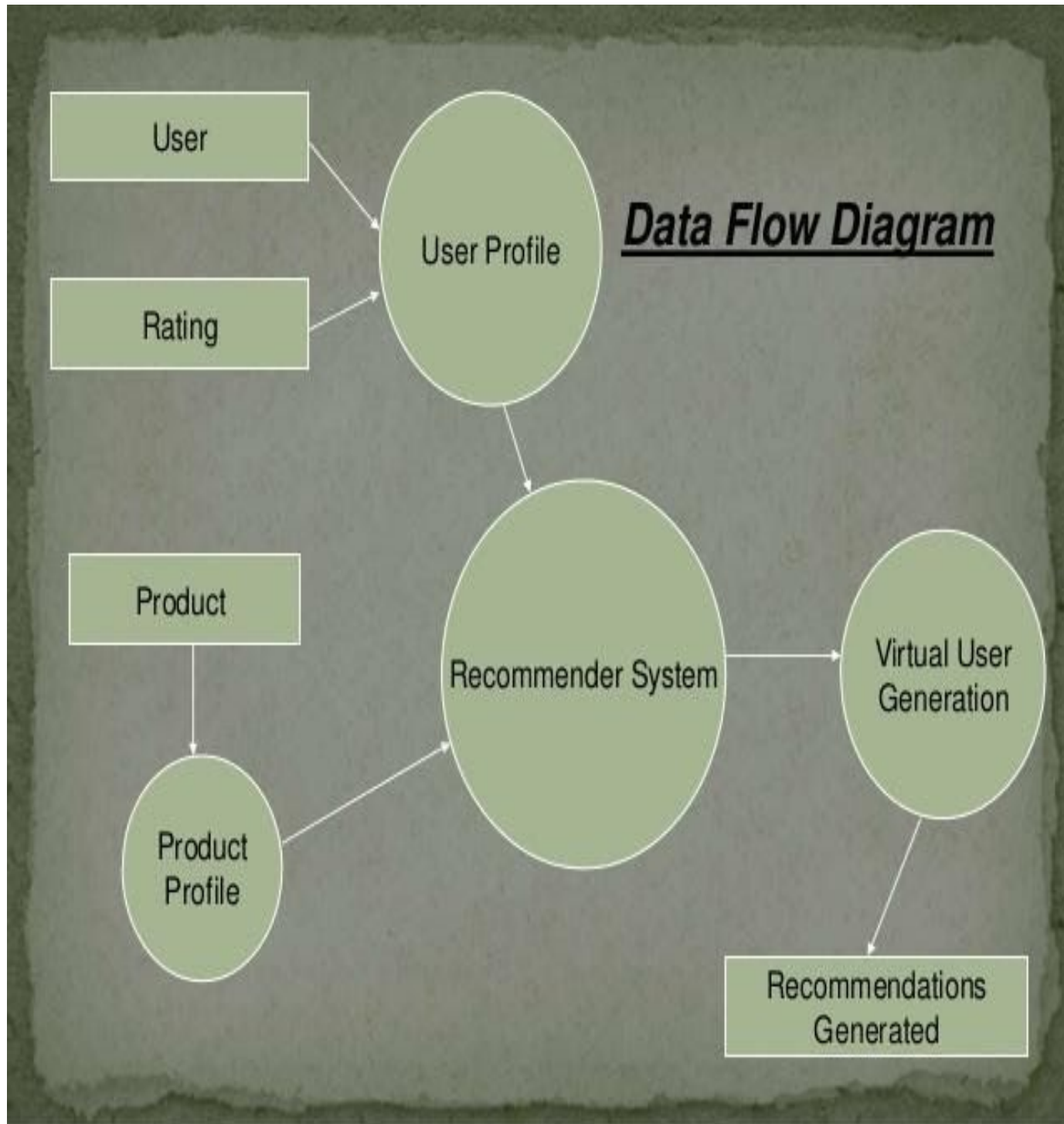


3.5 ER-DIAGRAM



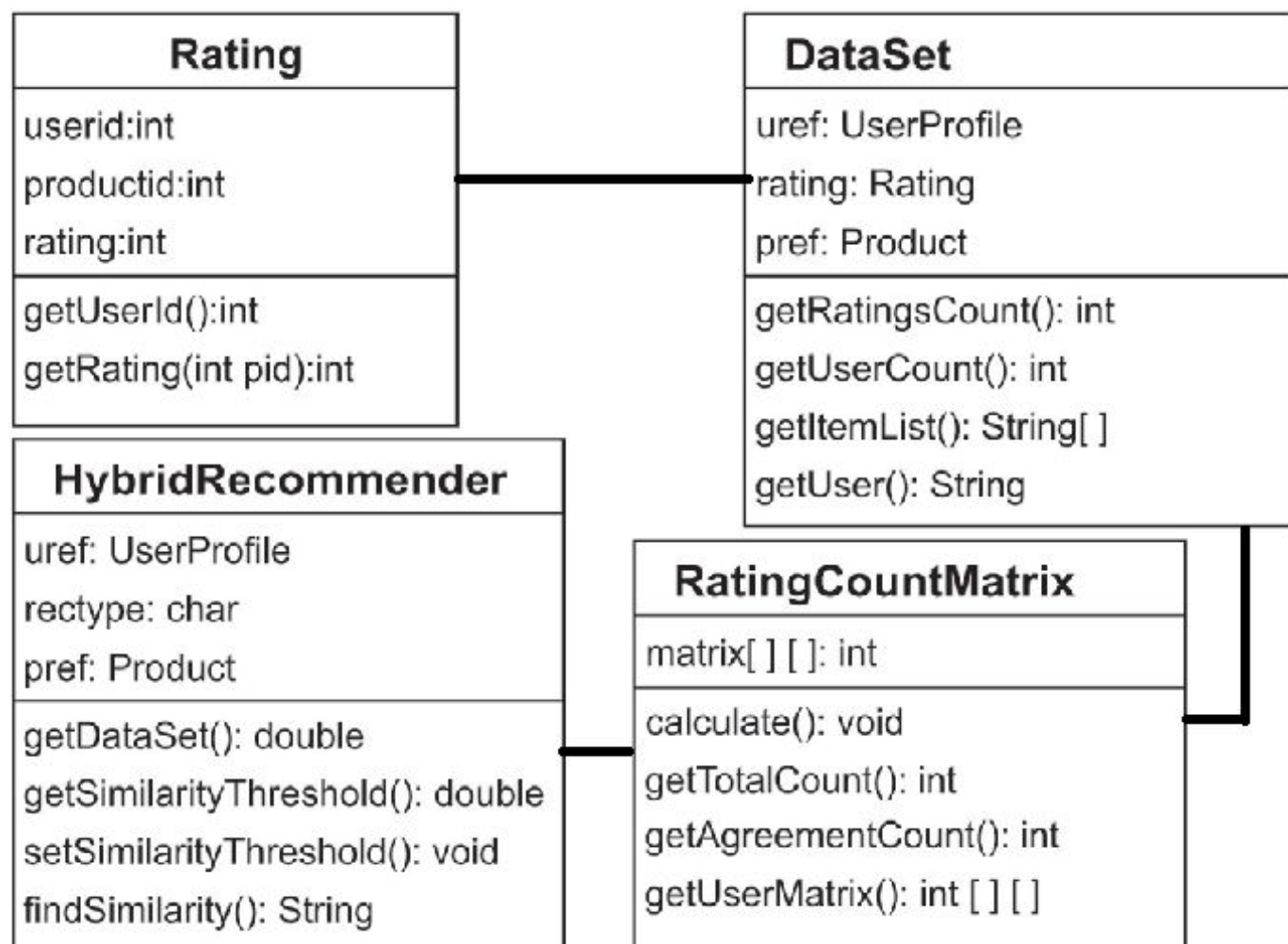
3.6 SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.



3.8 CLASS DIAGRAM

Class diagram in the unified modeling language (UML) is a type of static structure diagram that structures a system by showing the system classes attribute operators.



CHAPTER 4

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

1. Operating System : Windows, Linux, Mac (any one)
2. Programming Language : Python, Node.js

4.2 REQUIRED PROGRAMMING LIBRARIES

1. Python: Pandas,Scipy,numpy,pymongo
2. Node.js : Express.js,Mongoose ODM

4.3 HARDWARE REQUIREMENTS

1. CPU Type : Intel Pentium 4
2. Clock Speed : 3.0 GHz
3. Ram Size : 4 GB
4. Hard Disk Size : 40 GB
5. Monitor Type : 15 Inch Color Monitor
6. Keyboard Type : Internet Keyboard

CHAPTER 5

SOFTWARE SPECIFICATIONS

5.1 Python

Python is an interpreter, high-level, general-purpose programming language. It is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library

The python code you write is compiled into python byte code, which creates a file with extension `.pyc`. The byte code compilation happened internally, and almost completely hidden from the developer. Compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of your source code. Roughly, each of your source statements is translated into a group of byte code instructions. This bytecode translation is performed to speed execution byte code can be run much quicker than the original source code statements.

The `.pyc` file, created in compilation step, is then executed by appropriate virtual machines. The Virtual Machine is just a big loop that iterates through your byte code instructions, one by one, to carry out their operations. The Virtual Machine is the runtime engine of Python and it is always present as part of the Python system, and is the component that truly runs the Python scripts. Technically, it's just the last step of what is called the Python interpreter.

In our project the python program is responsible for computing the similarity score by using the Euclidean distance algorithm.

5.1.1 Required Python Libraries:

Numpy:

it's a library written for a python programming language, it provides large, multi-dimensional arrays and matrices, along with a large collection Numerical computing tools and high-level mathematical functions to operate on these arrays

Pandas:

It is a data manipulation library written for the Python programming language for a large quantity of data manipulation and analysis. In particular, it offers data structures and operations of numerical tables and data frames. It was released under the three-clause BSD license.

Scipy:

SciPy is a free and open-source Python library, providing various mathematical functions such as linear algebra, integration, interpolation.

OS:

The built-in OS module in python, to perform specific operations and interacting with the operating system.

5.2 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js - or just Node as it's commonly called - is used for developing applications that make heavy use of the ability to run JavaScript both on the client, as well as on server side and therefore benefit from the re-usability of code and the lack of context switching.

It's well suitable for building web applications and REST API development. In our project it is responsible to deliver the product recommendation from reading the database.

5.2.1 Required Nodejs Libraries:

Express.js:

It's a minimalist open-source web framework for developing a web application server in Nodejs, It is highly used to build and APIs.

Mongoose.js:

It's used in object modeling for MongoDB databases. provides a schema based solution to manage MongoDB collections.

Dotenv:

It's used for reading environment variables from a .env file into process .env. For secure, sensitive variables in the server.

CHAPTER 6

IMPLEMENTATION

6.1 IMPLEMENTATION PSYCHOGRAPHICS

Create a psychographic dataset to the product by conducting **Brainstorming** in each Cognitive attribute. Find the Lowest and Highest possible value (1....10)

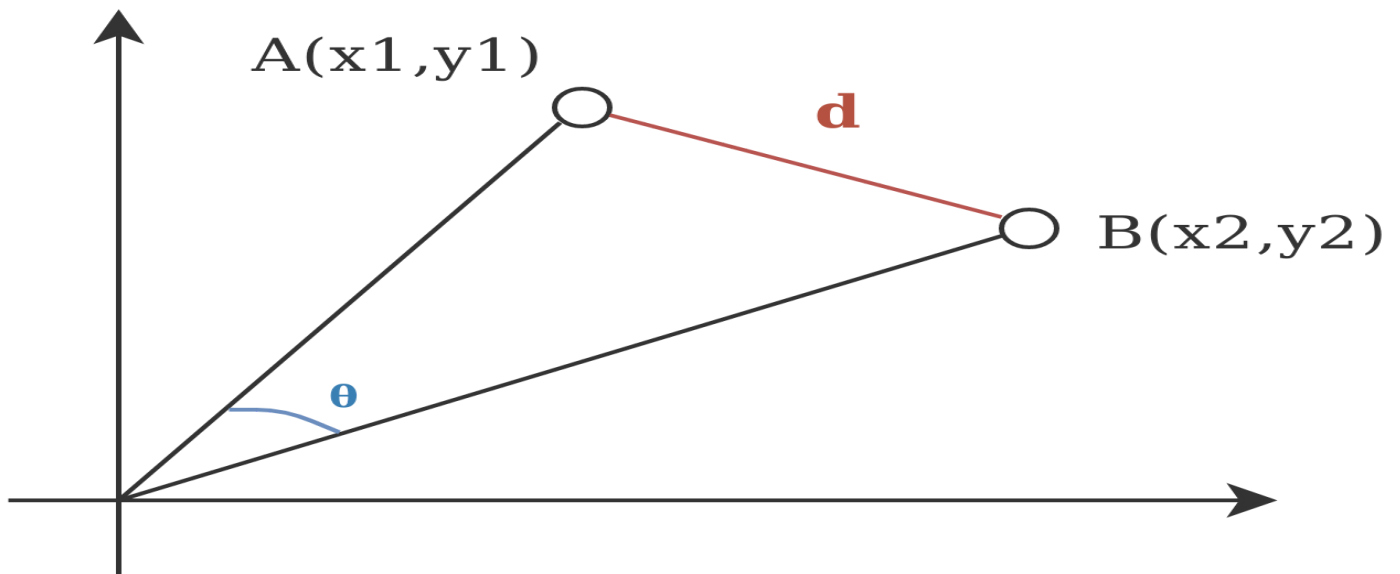
FORMULA:

Compare the similarity between the two vector products by using **Euclidean** distance. and calculate the mean value

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

EUCLIDEAN DISTANCE:

The Euclidean distance is used to find the distance between two vectors



6.1 IMPLEMENTATION COLLABORATIVE FILTERING

The collaborative filtering is predict the ratings and calculate the similarity score

FORMULA:

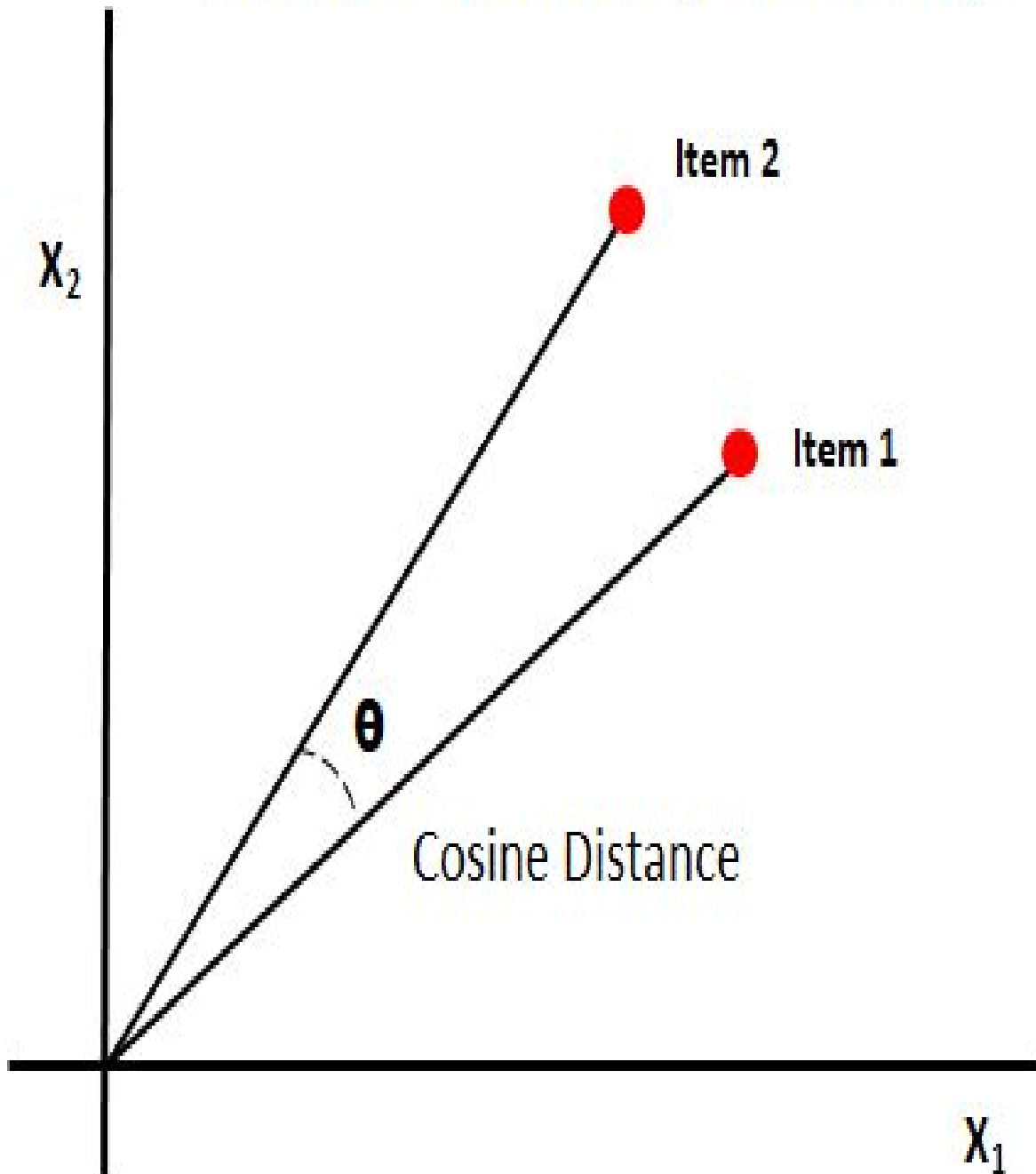
Compare the similarity between the two vector products by using **Cosine similarity**

COSINE SIMILARITY

Cosine similarity is the cosine of the angle between two n-dimensional vectors in an n-dimensional space. It is the dot product of the two vectors divided by the product of the two vectors' lengths (or magnitudes).

$$\text{similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

Cosine Distance/Similarity



CHAPTER 7

RESULT 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)

# Create a new Empty dataframe
newdf = pd.DataFrame(index=rows,columns=rows)

# Loop each record(min max) in dataframe and find euclidean 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)

# Create 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('Connection 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 similarity
app.get("/all",(req,res)=>{
  // select the all record
  model.find({},(err,result)=>{
    if(err){
      // send a error response
      res.json({
        status:"fail",
        data:err
      })
    }
    else{
      // Get a success response
      res.json({
        status:"success",
        data:result
      })
    }
  })
})

// Initialize an 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 successful 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 response 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

Node Js-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",
  "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:

```
Activities X-terminal-emulator Tue May 12 10:07 AM
zero@alpha: ~/workspace/cognitive-recsys

zero@alpha:~/workspace/cognitive-recsys$ python ca.py
product0 product1 product2 product3 product4 product5 product6 ... product93 product94 product95 product96 product97 product98 prod
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.
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.
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.
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 0.8
... ... ... ... ... ... ... ... ... ... ...
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.
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 0.8
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.
product98 1.84391 1.41421 3.1305 1.52315 1.41421 0.894427 2.05913 ... 1 1.7088 1 2.03961 1 0 2.
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$
```

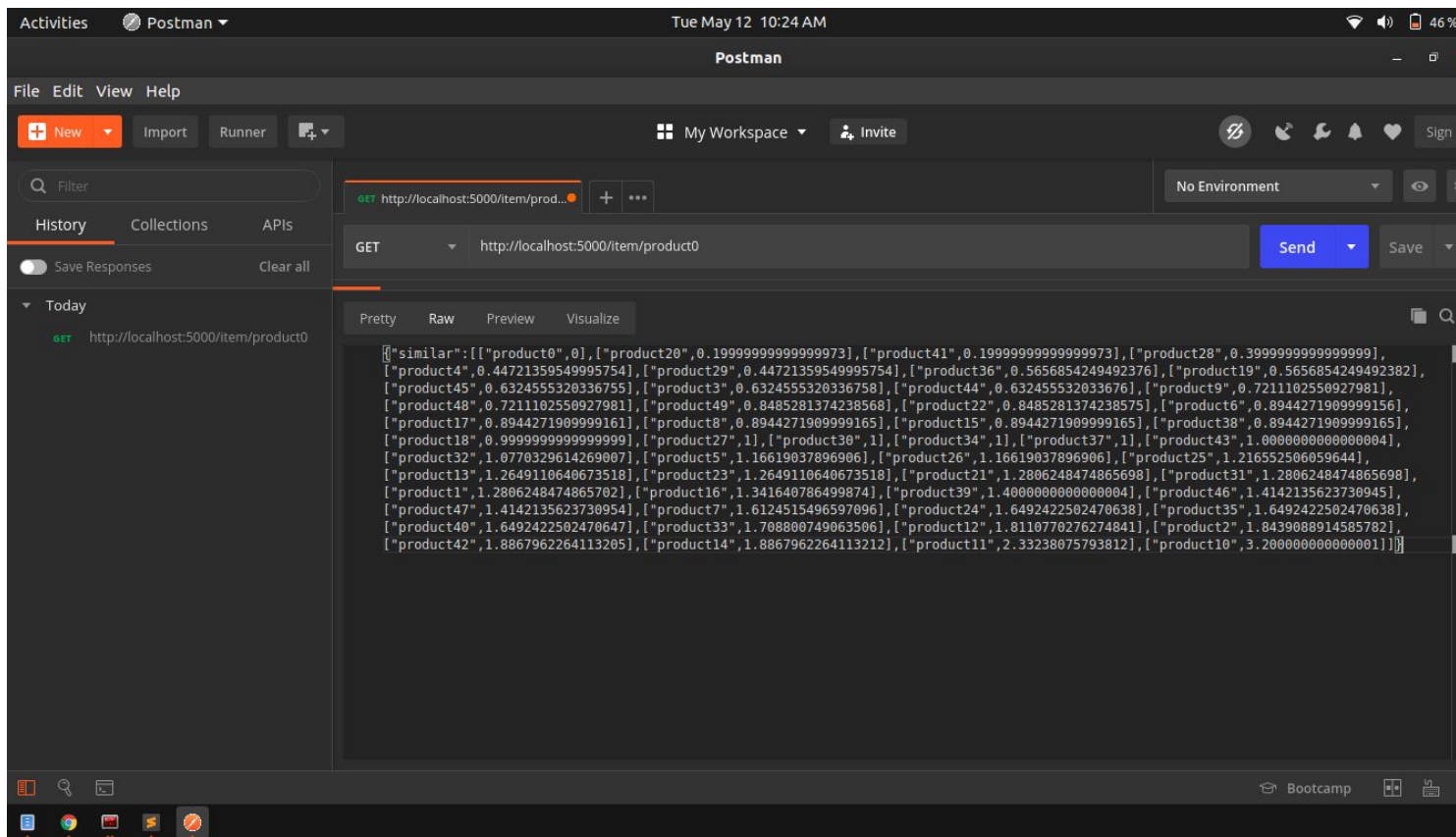
Description:

Here the python application reads the data sets and calculates the euclidean distance between the scores finally it constructs a matrix and fills up all values. the similarity matrix used to make look easier to fetch the relevant produces

STORE THE RESULT ON THE DATABASE:

```
zero@alpha: ~/workspace/cognitive-analytica/score_calcula...
zero@alpha:~/workspace/cognitive-analytica/score_calculation$ python3 ca.py
product0 product1 product2 product3 product4
product0 0 1.96977 1.4 2.8 1.84391
product1 1.96977 0 0.894427 1.28062 0.4
product2 1.4 0.894427 0 1.4 0.565685
product3 2.8 1.28062 1.4 0 1.07703
product4 1.84391 0.4 0.565685 1.07703 0
5f65bda9b607ce9be0c7bfaa
Added to Database Collection
5f65bdaab607ce9be0c7bfab
Added to Database Collection
5f65bdabb607ce9be0c7bfac
Added to Database Collection
5f65bdabb607ce9be0c7bfad
Added to Database Collection
5f65bdabb607ce9be0c7bfae
Added to Database Collection
Completed
zero@alpha:~/workspace/cognitive-analytica/score_calculation$
```

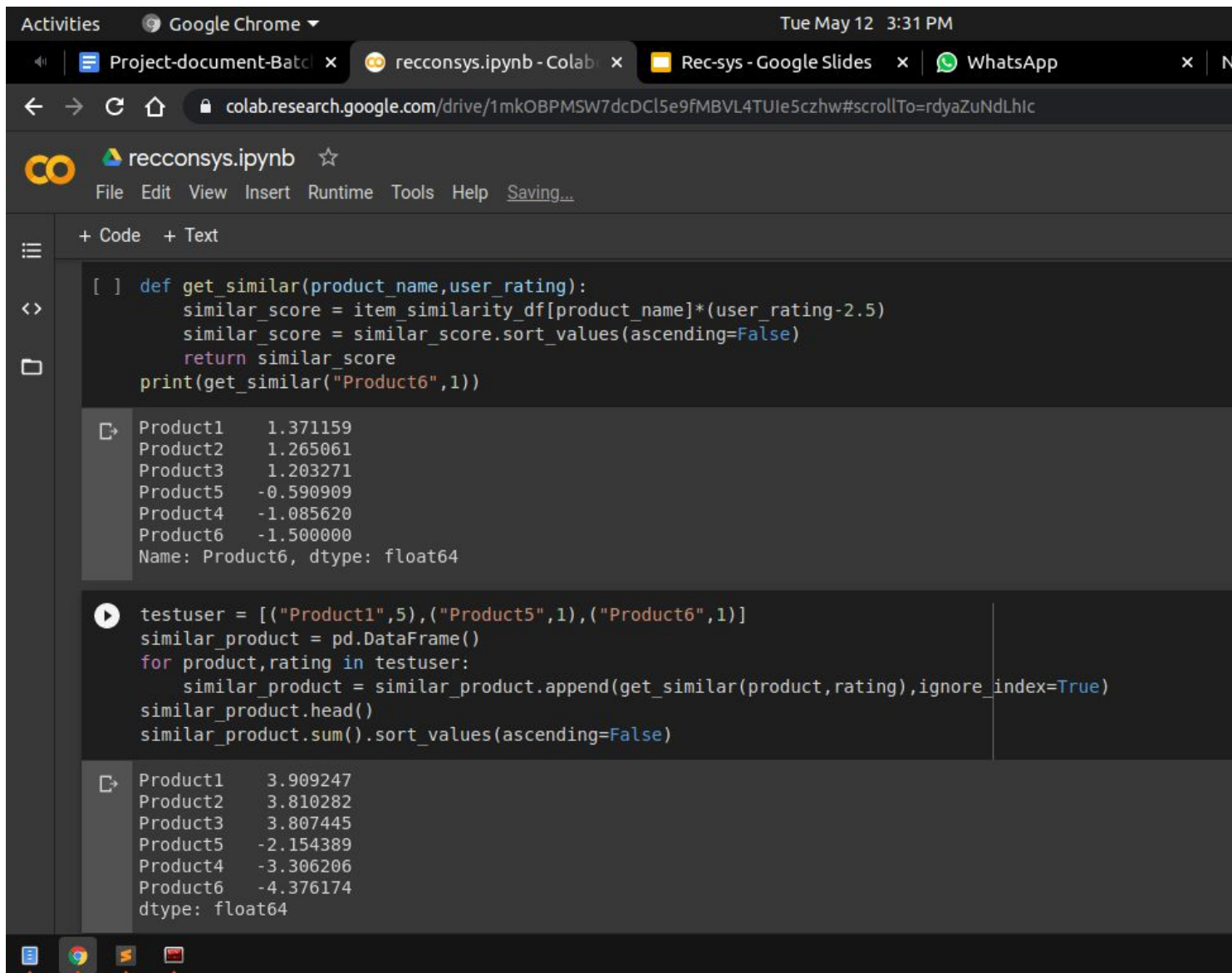
Description: The Calculated score has been stored on database and it returns a each stored data's reference ID



Description:

Here the REST 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 format of objects then return to the user

Collaborative filtering output:



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Project-document-Batch', 'reconsys.ipynb - Colab', 'Rec-sys - Google Slides', and 'WhatsApp'. The address bar shows the Colab URL. The notebook has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and 'Saving...'. The left sidebar shows a file explorer with '+ Code' and '+ Text' buttons. The main code area contains two code cells. The first cell defines a function 'get_similar' and prints its output for 'Product6' with a rating of 1. The second cell creates a test user and iterates through products to calculate similarity scores. The output of the first cell is a list of products and their similarity scores for Product6, with a dtype of float64. The output of the second cell is a list of products and their similarity scores for the test user, also with a dtype of float64.

```
[ ] 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 algorithms.

USER INTERFACE:

WhatsApp x Data | Atlas: MongoDB At x wired_market_admin - Wi x +

localhost:3000/rec/products

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	0.39999999999999999
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

WhatsApp x Meet - kez-ixqt-tgh x ca/ca-visualization-purp x Data | Atlas: MongoDB x wired_market_admin - v x +

localhost:3000/rec/products

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	0.39999999999999999
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

CHAPTER 8

SOFTWARE TESTING

8.1 INTRODUCTION

Software testing is a method of assessing the functionality of a software program. There are many different types of software testing but the two main categories are dynamic testing and static testing. Dynamic testing is an assessment that is conducted while the program is executed; static testing, on the other hand, is an examination of the program's code and associated documentation. Dynamic and static methods are often used together.

8.2 TESTING OBJECTIVES

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has a high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to be working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

1. Correctness
2. Implementation efficiency
3. Computational Complexity

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which reexamines the implementation phase of algorithm development. Tests for computational complexity

amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more the data is entered in all forms separately and whenever an error occurs, it is corrected.

Immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The development process involves various types of testing. Each test type addresses a specific testing requirement. The most common types of testing involved in the development process are:

- Unit Test
- System Test
- Integrating Test
- Functional Test
- Black Box Test
- White Box Test

8.3 UNIT TESTING

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results. Functional and reliability testing in an Engineering environment. Producing tests for the behavior of components (nodes and vertices) of a product to ensure their correct behavior prior to system integration.

8.4 SYSTEM TESTING

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and works towards the integration of the entire computers based system. Nothing is complete without testing, as it is a vital success of the system.

8.5 INTEGRATING SYSTEM

Testing in which modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even

distributed to the public at large in hopes that they will buy the final product when it is released.

8.6 FUNCTIONAL TESTING

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

8.7 BLACK BOX TESTING

Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

8.8 WHITE BOX TESTING

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done using the percentage value of load and energy. The tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Thus the recommender system has been well suitable to anyone from small level application to enterprise needs. because of the modular nature of that application, it can be broken into multiple pieces based on the company architecture requirements. Even it enables us to provide a recommendation as a service from the cloud by third-party vendors.

It doesn't require any constant monitoring and intensive management, the failure of the system can't produce any downtime on the main application, it's easy to spin up another instance of the application if one server is down using load balancers. when the small company grows into a large enterprise, the software also needs to grow by their needs, it's achieved by scalability. that application provides the feature without any problems and performance degradation specifically on the huge amount of datasets. The REST API nature of the application enables to make a multiple version of the recommender of application at the same time, it can be easy to change at any time, so that the adding and testing of new features doesn't require any downtime and development stage.

The performance of the recommendation system can easily be achieved by multiple machines from various locations, and it's possible to create a new instance by programmatically at any high demands situation The Precalculation of the Psychographics score has reduced the cold start problem, even if the initial score did not produce any expected results on the first iteration, that problem has been recorded and instructs to recalculate the score by experts. that iteration has been continued until the recommendation decision performs well. It also makes a marketing team know more about their product and users expectations in the consumer's market, the goal of the system to make visitors into buyers by providing an exact product they are looking for.

9.2 FUTURE ENHANCEMENT

1. Implementing deep learning in the recommendation system. There are several ways to utilize deep learning in recommender systems. Neural networks can be trained to predict ratings or interactions based on item and user attributes. You can use deep neural nets to predict next action based on historical actions and content. Also, deep autoencoders can be used for collaborative filtering projecting interaction vectors to a latent space similarly as matrix factorization.
2. The error and incorrect decision making currently has been identified by manual checks, In a future version of the application comes up with the power of machine learning to detect the anomalies in real-time. the system has been capable of counting how many times that product has been recommended to the user, and how many times that user responded to the product. based on the count and similarity score, it distinguishes the state of decision-making whether its success or not, the poor user engagement clearly shows the failure of decision making
3. In a future version of the application bring more scalability specifically focused on making the scheduled chunks of computation processes favor for the third-party vendors to provide a recommendation as a service much like a software as a service in the cloud platform, it brings the cost of the application as low as possible

CHAPTER 10

REFERENCES

1. Collaborative Filtering Recommendation Algorithm Based on Knowledge Graph - Xiaoqin Zeng , Ruihui Mu -College of computer information hohai university nanjing -April 2018
2. A Comparative Study of Psychographic Segmentation Hui Liu 1,† , Yinghui Huang 1,2,† , Zichao Wang 3 , Kai Liu, Xiangen Hu , and Weijun Wang -Key Laboratory of Adolescent Cyberpsychology and Behavior, Ministry of Education,Central China Normal University, Wuhan - March 2019
3. Augmenting E-Commerce Product Recommendations by Analyzing Customer Personality - Anwesh Marwade, Nakul Kumar, Shubham Mundada, and Jagannath Aghav 2019
4. A Collaborative Recommendation System for Online Courses Recommendations, Raghad Obeidat ; Rehab Duwairi ; Ahmad Al-Aiad
5. Movie Recommendation System Using Semi-Supervised Learning, Sushmita Roy ; Mahendra Sharma ; Santosh Kumar Singh