PRIORITY BASED COMPARISON SYSTEM

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENT OF THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

SUSHMITA GUPTA (1402913111)
SARTHAK AGARWAL (1402913088)
MOHAMMAD SHADAN (1402913063)

Supervised by

Ms. NIDHI GOYAL

Session 2017-18

DEPARTMENT OF INFORMATION TECHNOLOGY

KIET GROUP OF INSTITUTIONS, GHAZIABAD

(Affiliated to Dr. A. P. J. Abdul Kalam Technical University, Lucknow, U.P., India)

DECLARATION

We declare that

a. the work contained in this report is original and has been done by us under the guidance of our supervisor.
b. the work has not been submitted to any other institute for any degree or diploma.
c. we have followed the guidelines provided by the institute to prepare the report.
d. we have conformed to the norms and guidelines given in the ethical code of conduct of the institute.
e. wherever we have used materials (data, theoretical analysis, figures and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references.

Signature of the student
Name: Sushmita Gupta
Roll number: 1402913111

Signature of the student
Name: Sarthak Agarwal
Roll number: 1402913088

Signature of the student
Name: Mohammad Shadan
Roll number: 1402913063

Place: KIET Group of Institutions, Ghaziabad
Date:

CERTIFICATE

This is to certify that the Project Report entitled, “Priority based comparison system” submitted by Ms. Sushmita Gupta, Mr. Sarthak Agarwal and Mr. Mohammad Shadan in the Department of Information Technology of K.I.E.T Group of Institutions, Muradnagar, affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India, is a record of bonafide project work carried out by them under my supervision and guidance, and is worthy of consideration for the award of the degree of Bachelor of Technology in Information Technology of the Institute.

Signature of supervisor

Supervisor Name: Ms. Nidhi Goyal

Date:

List of Figures

List of Acronyms

PBCS	*Priority Based Comparison System*
MEAN	*Mongo DB, Express JS, Angular JS, Node JS*

Abstract

E-commerce (electronic commerce or EC) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business. The terms e-commerce and e-business are often used interchangeably. In the present scenario, E-Com business is a very competitive business. Various websites provide a different and heavy discount over a wide range of items to attract customers toward them. The advantage of this competition is directly related to the customer as a customer can get better quality at less price. But there are many E-Com websites available today and one must drill a lot to get the best deals for a product. This is a very time-consuming process. A comparison shopping website, sometimes called a price comparison website, Price Analysis tool, comparison shopping agent, shopbot or comparison shopping engine, is a vertical search engine that shoppers use to filter and compare products based on price, features, reviews and other criteria. Most comparison shopping sites aggregate product listings from many different retailers but do not directly sell products themselves instead mostly they make use of affiliate marketing to earn. Very often this comparison is not only in terms of price but other attributes as well. For example- in laptops we compare its specifications such as HDD, RAM etc. Our project intends to further enhance this comparison mechanism where it not only compare in terms of price and attributes but also takes in account the priority of user when selecting attributes for comparison, i.e. the features which are more desirable will hold greater weightage. This allows the user to further narrow down the search spectrum and ease the lookup mechanism for a product. This project will include only laptops as products and will work over hundreds of laptops for priority comparison. This will help in saving the time of customer from doing door to door shopping for comparing the project. It should also be noted that no such tool is yet available in the market and idea is itself unique.

Contents

Page No.

Chapter 1

INTRODUCTION

1.1. Problem Definition

As an age-old saying goes, ' Necessity is the mother of invention'. Internet offers an overwhelming answer to all this turmoil of an online shopper in the form of "Price Comparison Sites". Price Comparison sites save online shoppers from undertaking a hectic web research, and basically cater the eventual product of the entire procedure of web research, otherwise undertaken by an avid web shopper before making a buying decision. Price Comparison websites are also known as "Aggregators" because they aggregate information on a product from many websites offering online shopping of the product. These sites render you easy and lucid comparisons of the prices on offer of a product under consideration, without any toiling involved, on a single screen in a matter of a few clicks of the mouse! These sites manage to save a substantial amount of effort, time and money of the prospective shopper, and above all, take away the headache of compiling information.

1.2. Project Overview

The project being developed is enhancing the abilities of such comparison website by allowing the user to specify its priorities when selecting the attributes. For one user RAM may be preferential while buying the laptop while for the other one processor might be the foremost priority. Today, the comparison sites return and rank the products as per their standards without taking in account the needs of user. This inability is being overcome by this idea of priority based comparison.

The priority based comparison will deploy algorithms at work which will first filter the products as per the requirements and then sort them in order of priority of attributes. Further, the scope of the project can be expanded in the future by comparing the hardware having the same specification on the abstract layer but differ real-time performance. Such a system will be far more efficient and will provide more reliable comparisons as compared to the existing systems.

This website has been developed using the MEAN stack technology. The term MEAN stack refers to a collection of JavaScript based technologies used to develop web applications. MEAN is an acronym for MongoDB, ExpressJS, AngularJS and Node.js. From client to server to database, MEAN is full stack JavaScript.

Node.js is a server-side JavaScript execution environment. It’s a platform built on Google Chrome’s V8 JavaScript runtime. It helps in building highly scalable and concurrent applications rapidly. Express is lightweight framework used to build web applications in Node. It provides several robust features for building single and multi-page web application. Express is inspired by the popular Ruby framework, Sinatra. MongoDB is a schema-less NoSQL database system. MongoDB saves data in binary JSON format which makes it easier to pass data between client and server. AngularJS is a JavaScript framework developed by Google. It provides some awesome features like the two-way data binding. It’s a complete solution for rapid and awesome front end development. For the backend development python is used along with its many useful modules like Beautiful soup and PyMongo. All the scripts are being written into python language.

1.3. Purpose of project

Priority based comparison system fulfills the requirements of user. The user does not have to go shop to shop for comparing the laptops according to the specifications. He/she can avail the facility of filtering and sorting the products at home with great ease. The purpose of this project is to make as user friendly as it can be. And because of this priority based comparison system user will be able to know the present scenario of the products according to its range wise as well its quality wise. .

Chapter 2

LITERATURE REVIEW

Since the mid-90s, a large number of comparison-shopping agents of various types have developed in order to assist consumers to make shopping decisions in online environments. The first commercial shopping agent, called Jango, was produced by Netbot, a start - up company from Seattle founded by University of Washington professors Oren Etzioni and Daniel S.Weld Junglee. It was the comparison shopping technology pioneer and was soon acquired by Amazon.com. Other early comparison shopping agents include pricewatch.com and killerapp.com. Most of them were price comparison for computer related products used by geeks therefore it didn't attract much public attention.

In the early development stage from 1995 to 2000, comparison shopping agents include not only price comparison, but also rating and review services for online suppliers and products. Websites like Epinion.com provided reviews and rating services for products. Overall, they included three categories of comparison shopping services. Further these services were consolidated resulting from 2000, Shopping.com and Shopzilla have Pricegrabber to three leading comparison shopping agents presents an overview of comparison-shopping services and shopbot design, then and summarizes the evolution of comparison-shopping in the last 13 years. The book covers the three different domain related to comparison shopping: the technical design of shopbots, the comparison shopping user behaviors, and the economics of the comparison shopping, the technical design and the business model are both considered since they affect one another. Before 1995 research was mainly focused on how to retrieve data from heterogeneous data sources, later on the research trend focused on optimize the data retrieval algorithms and improve consumer experience.

According to contemporary CSEs may not present complete and accurate price dispersion information mainly due to consumer heterogeneity. In the researchers also identifies the limitation of current design shopbots and offered the use of semantic web and web services, including presentation of theoretical models for the study of CS as well as online shopping in general.

According to most of the shopping agents are price dominated, unreflective of the nature of supplier/consumer differentiation or the changing course of differentiation over time, therefore they presented an advanced shopping agent. It is a personalized and integrative shopping agent, were both supplier and consumer differentiation are taken into account. This agent can leverage the interactive power of the Web for a more accurate understanding of consumer’s preferences. It comprises of a product/merchant information collector, a consumer behavior extractor, a user profile manager, and an on-line learning personalized ranking module.

Comparison shopping agents/engines may be represented by early examples of the Semantic Web, but early systems used wrappers to extract structured information about products from Web pages. Wrapper construction requires extensive programming and results in a fragile system, since they need to be reprogrammed when an online store changes its layout. Modern comparison shopping systems get most of their data from relational data feeds generated by suppliers. While this typically yields more robust results, it requires supplier’s cooperation and may produce less comprehensive listings.

Comparison-shopping is the third most popular online shopping option after eBay and Amazon. However, instead of being dominated by one large player, there are many service providers in this field that compete. They include independent comparison-shopping service providers like shopping.com, nextag.com, shopzilla.com as well as services by established online portals or search engines like Bing Shopping by Microsoft and Google Product Search by Google. The large number of players makes the mutual learning and innovation in this field faster and it gives a diverse field for research.

According to current research on comparison-shopping and recommendation agents is largely separated by different research disciplines, ranging from computer science and information systems, economics, consumer behavior, to electronic commerce.

Chapter 3

FEASIBILITY STUDY

A feasibility study is made to see if the project on completion will serve the purpose of the organization for work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of the system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as technical, economic, operational and behavioral. Following are its features:

3.1 Economical Feasibility

The developing system must be justified by cost and benefit Criteria to ensure that efforts are concentrated on the project, which will give the best return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during the preliminary investigation:

a. The cost conducts a full system investigation.
b. The cost of the hardware and software.
c. The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available it gives an indication of the system is economically possible for development. It includes:

a. Cost of the software development.
b. Selling cost of the developed project.
c. Revenue/Profits etc.

3.2 Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems and satisfies the requirements identified in the requirements analysis phase of system development. It includes:

a) Degree to which the proposed development projects fits in with the existing business environment and objectives
b) The willingness of each stakeholder for the development of the S/W Project

3.3 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of inputs, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system of running the system once it has been designed. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system Technical issues raised during the investigation are:

a. Does the existing technology sufficient for the suggested one?
b. Can the system expand if developed?
c. Sufficient technical expertise (or Team members)
d. Reliability, Availability and capability of S/w, H/w and networks used for the development.

Chapter 4

PROPOSED SYSTEM

4.1. Existing System

The current system does not have the common platform where the prioritization and comparison of products could be done. Further, sometimes customers want to specifically know positive or negative aspect of the product for which he must classify and review comments manually. The 21st century is the century of knowledge. For every answer people first, use Google. The E-com giant like amazon.in, flipkart.com, snap deal offers various discounts for trending products to gain an advantage over each other but there is no comparison available at any of these websites.

There are some websites like Smartprix, GSM Arena, NDTV Gadgets which allows types of comparisons, but they don’t prioritize according to the user input.

4.1.1. Drawback

There is no existing system which compares according to the priority of user inputs of the given product specifications. There is no authentic website where you can get the comparative study of trending products based on reviews.

4.2. Proposed System

Several comparison websites have been studied to prepare the base idea and objectives for this project. The flaws and deficient features of the websites have been studied and the primary objective of this project is to overcome the basic limitations encountered. Our proposed system is

To provide the product comparison with respect of the attributes of the products user have enquired for.

- To map the various attributes with a product score and make a general non-priority based comparison on the basis of attributes of the product to provide an overall generalized score to the product.
- To provide the results on the basis of priority of the attributes as a result in difference in weightage being allotted to the different attributes resulting in ranking on the basis of attribute priority.
- The comparison will be based on the key attributes of the product for example if the product is laptop then its key attributes can be processor (Dual Core or Quad Core), RAM, Hard Drive size, Battery backup, Windows type, release date of the laptop, design (Dimension, weight) of the laptop, display (type, touch, size).

The user just needs to log in and select the trending product. This is done in four phases

- Phase 1: Build the UI using the node.js and angular.js
- Phase 2: Scrap the product details from different websites and store in the database
- Phase 3: Comparison algorithm works when the user inputs it's priority.
- Phase 4: Display the results in the UI.

4.2.1 Algorithm and Code

There are two self-designed algorithms used for prioritizing and comparing the products. These algorithms help the user to understand how the specification score is coming all along the way. Following is the flow diagram of how the project will be initiated:

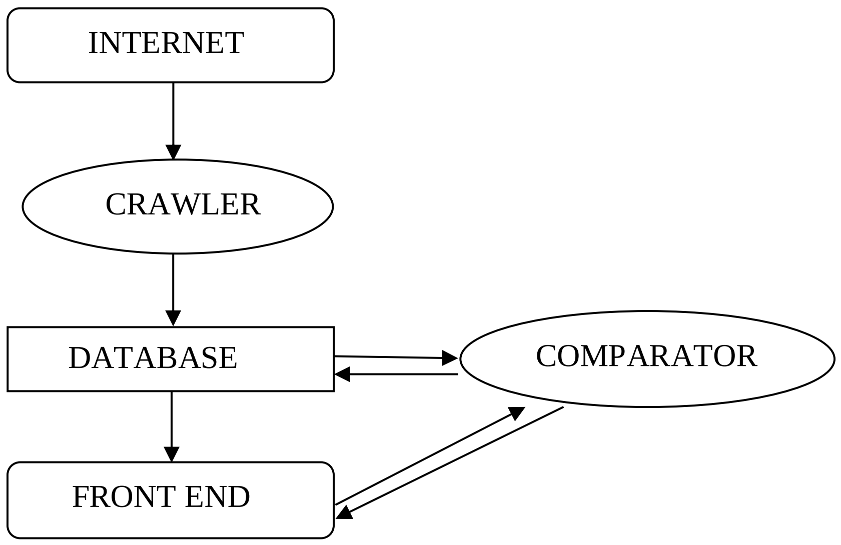


Fig (1) Flow diagram

Our project works on three major modules. There are three algorithms, each under one of the module. The modules along with the algorithms are as follows:

4.2.1.1. Crawler Module

A crawler is a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index. The flow diagram of our crawler can be seen as:

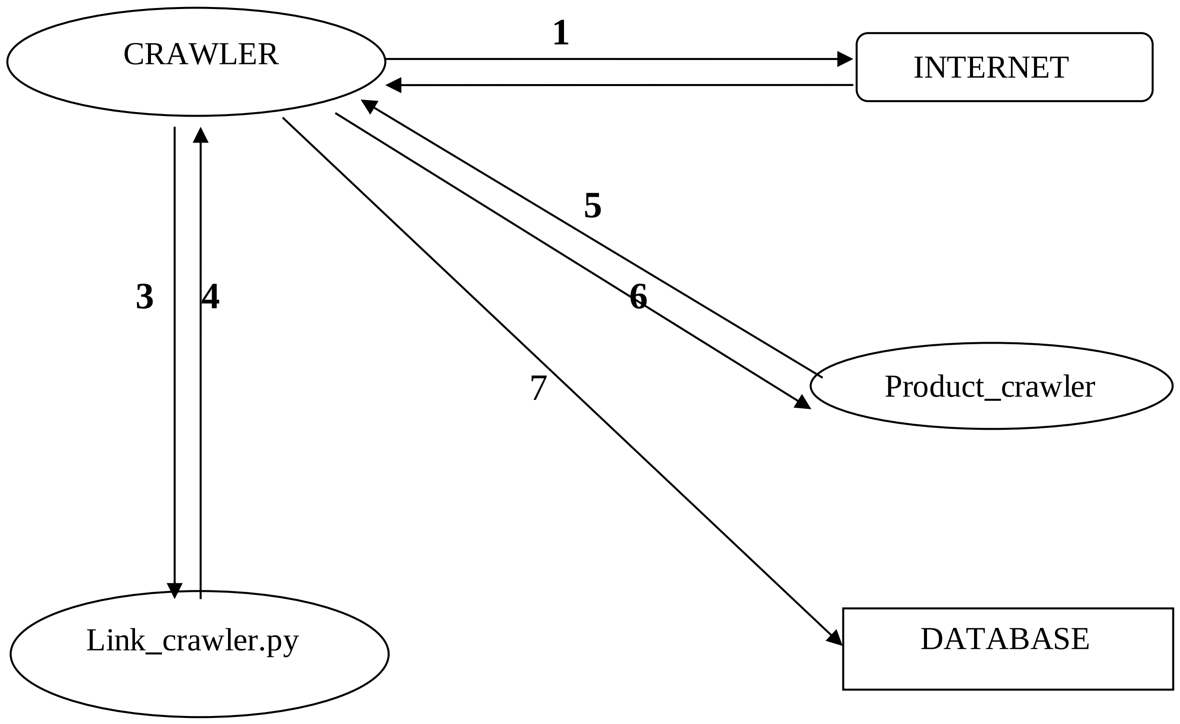
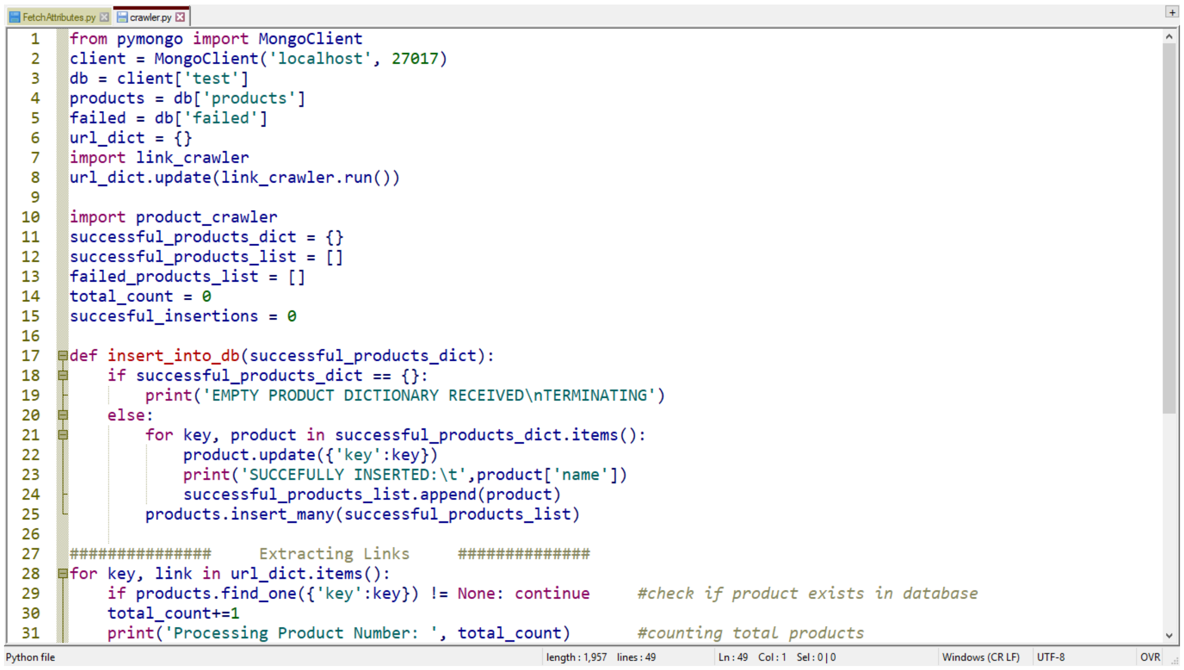


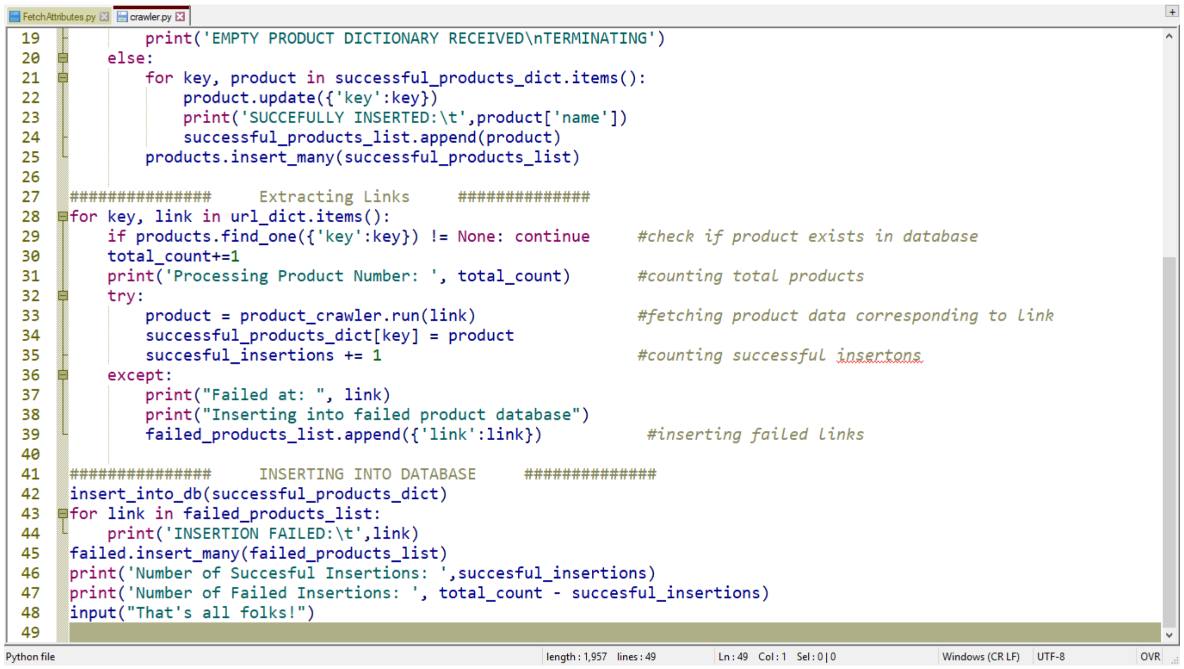
Fig (2) Crawler Flow diagram

The number represent the following functions:

1. Request links.
2. Links Fetched
3. link_crawler called
4. Parsed URL with individual product links returned.
5. Product link passed.
6. Product data parsed into structured form is returned.
7. Product data inserted into database.



```python
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['test']
products = db['products']
failed = db['failed']
url_dict = {}
import link_crawler
url_dict.update(link_crawler.run())

import product_crawler
successful_products_dict = {}
successful_products_list = []
failed_products_list = []
total_count = 0
succesful_insertions = 0

def insert_into_db(successful_products_dict):
    if successful_products_dict == {}:
        print('EMPTY PRODUCT DICTIONARY RECEIVED\nTERMINATING')
    else:
        for key, product in successful_products_dict.items():
            product.update({'key':key})
            print('SUCCEFULLY INSERTED:\t',product['name'])
            successful_products_list.append(product)
        products.insert_many(successful_products_list)

###############     Extracting Links     ##############
for key, link in url_dict.items():
    if products.find_one({'key':key}) != None: continue     #check if product exists in database
    total_count+=1
    print('Processing Product Number: ', total_count)       #counting total products
```



```python
        print('EMPTY PRODUCT DICTIONARY RECEIVED\nTERMINATING')
    else:
        for key, product in successful_products_dict.items():
            product.update({'key':key})
            print('SUCCEFULLY INSERTED:\t',product['name'])
            successful_products_list.append(product)
        products.insert_many(successful_products_list)

###############     Extracting Links     ##############
for key, link in url_dict.items():
    if products.find_one({'key':key}) != None: continue     #check if product exists in database
    total_count+=1
    print('Processing Product Number: ', total_count)       #counting total products
    try:
        product = product_crawler.run(link)                 #fetching product data corresponding to link
        successful_products_dict[key] = product
        succesful_insertions += 1                           #counting successful insertons
    except:
        print("Failed at: ", link)
        print("Inserting into failed product database")
        failed_products_list.append({'link':link})           #inserting failed links

###############     INSERTING INTO DATABASE     ##############
insert_into_db(successful_products_dict)
for link in failed_products_list:
    print('INSERTION FAILED:\t',link)
failed.insert_many(failed_products_list)
print('Number of Succesful Insertions: ',succesful_insertions)
print('Number of Failed Insertions: ', total_count - succesful_insertions)
input("That's all folks!")
```

Fig (3) Crawler

4.2.1.2. Comparator Module

4.2.1.2.1 Fetch Attributes Algorithm

The following flow diagram will explain the functioning of this module.

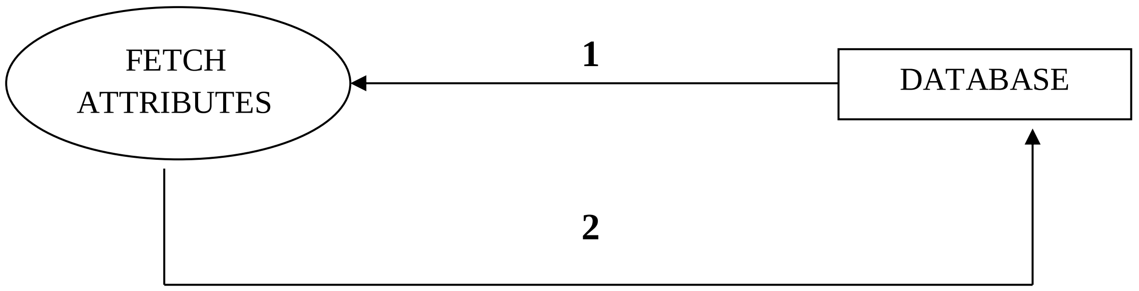
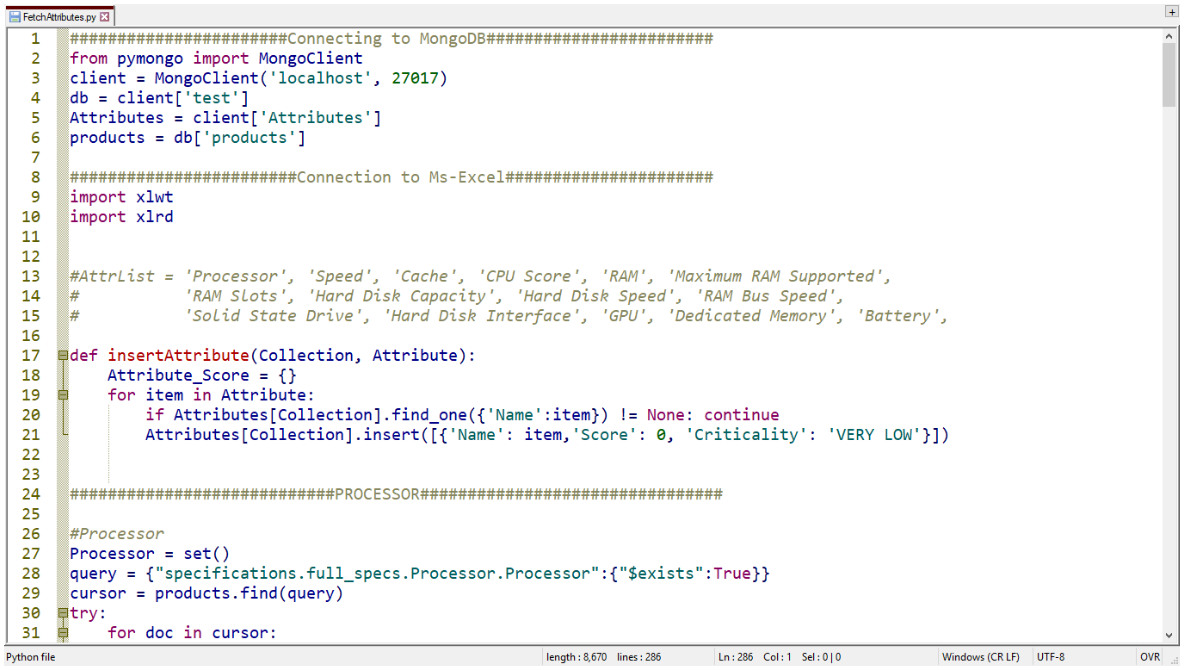


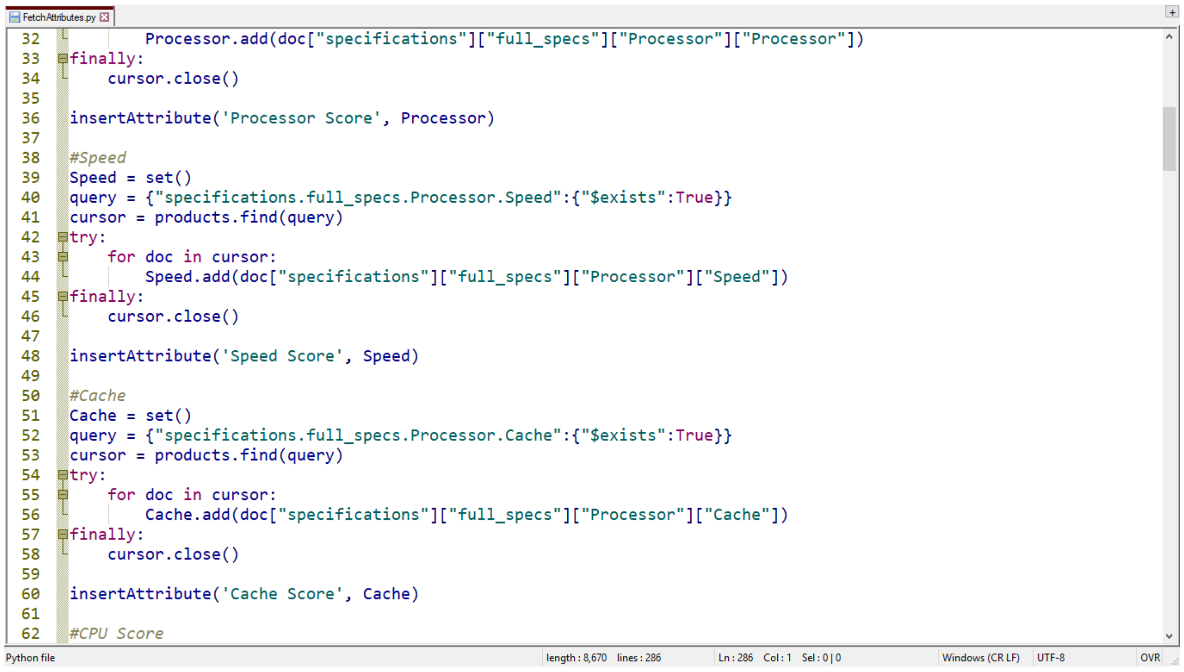
Fig (4) Fetch Attributes Flow Diagram

The number represent the following functions:

1. Read all the products from database.
2. Create new documents and collections with attribute data.

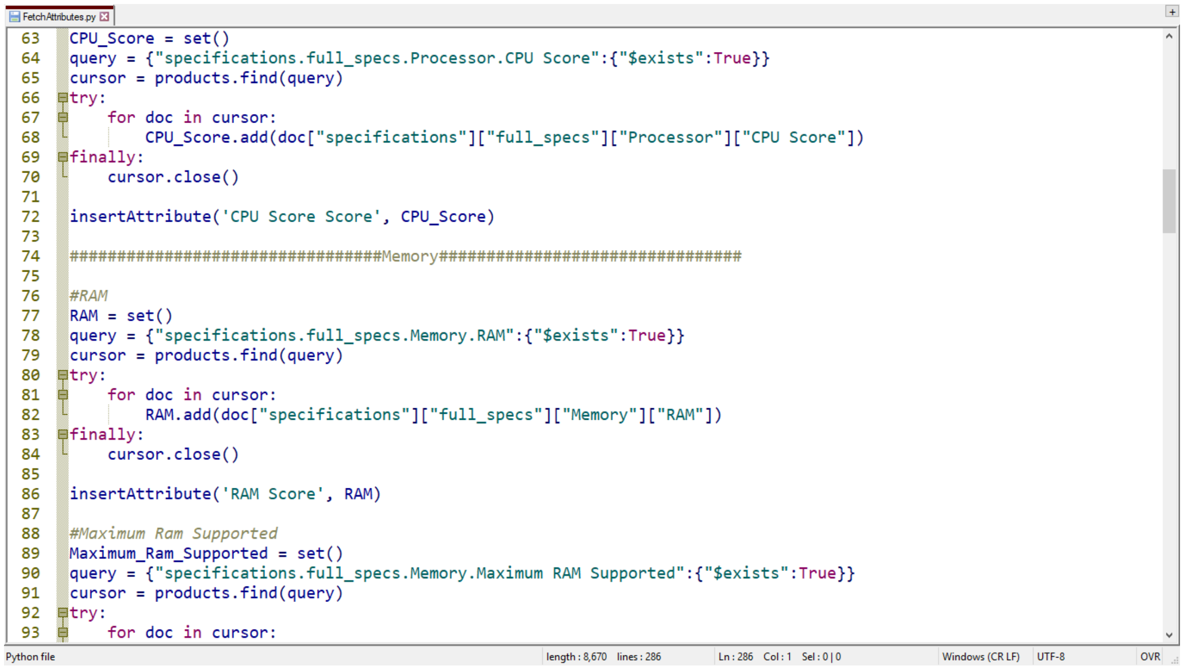


```
#######################Connecting to MongoDB########################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['test']
Attributes = client['Attributes']
products = db['products']

########################Connection to Ms-Excel#####################
import xlwt
import xlrd


#AttrList = 'Processor', 'Speed', 'Cache', 'CPU Score', 'RAM', 'Maximum RAM Supported',
#           'RAM Slots', 'Hard Disk Capacity', 'Hard Disk Speed', 'RAM Bus Speed',
#           'Solid State Drive', 'Hard Disk Interface', 'GPU', 'Dedicated Memory', 'Battery',

def insertAttribute(Collection, Attribute):
    Attribute_Score = {}
    for item in Attribute:
        if Attributes[Collection].find_one({'Name':item}) != None: continue
        Attributes[Collection].insert([{'Name': item,'Score': 0, 'Criticality': 'VERY LOW'}])


############################PROCESSOR################################

#Processor
Processor = set()
query = {"specifications.full_specs.Processor.Processor":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
```


FetchAttributes.py

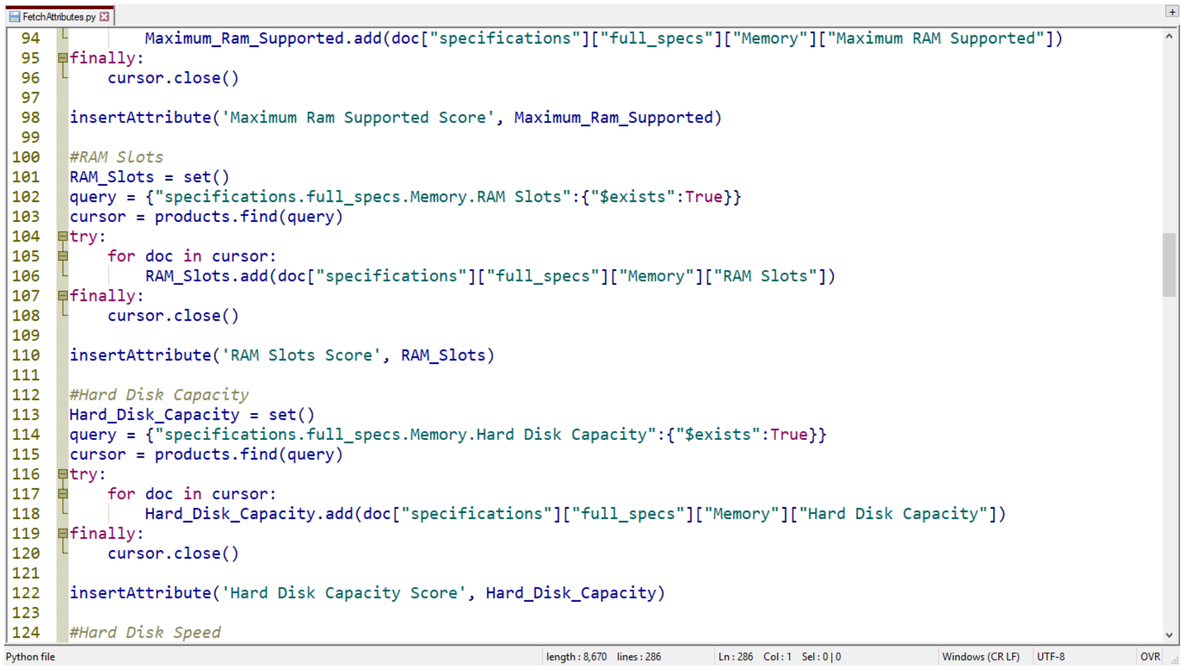
```python
            Processor.add(doc["specifications"]["full_specs"]["Processor"]["Processor"])
finally:
    cursor.close()

insertAttribute('Processor Score', Processor)

#Speed
Speed = set()
query = {"specifications.full_specs.Processor.Speed":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Speed.add(doc["specifications"]["full_specs"]["Processor"]["Speed"])
finally:
    cursor.close()

insertAttribute('Speed Score', Speed)

#Cache
Cache = set()
query = {"specifications.full_specs.Processor.Cache":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Cache.add(doc["specifications"]["full_specs"]["Processor"]["Cache"])
finally:
    cursor.close()

insertAttribute('Cache Score', Cache)

#CPU Score
```

Python file   length : 8,670   lines : 286   Ln : 286   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR

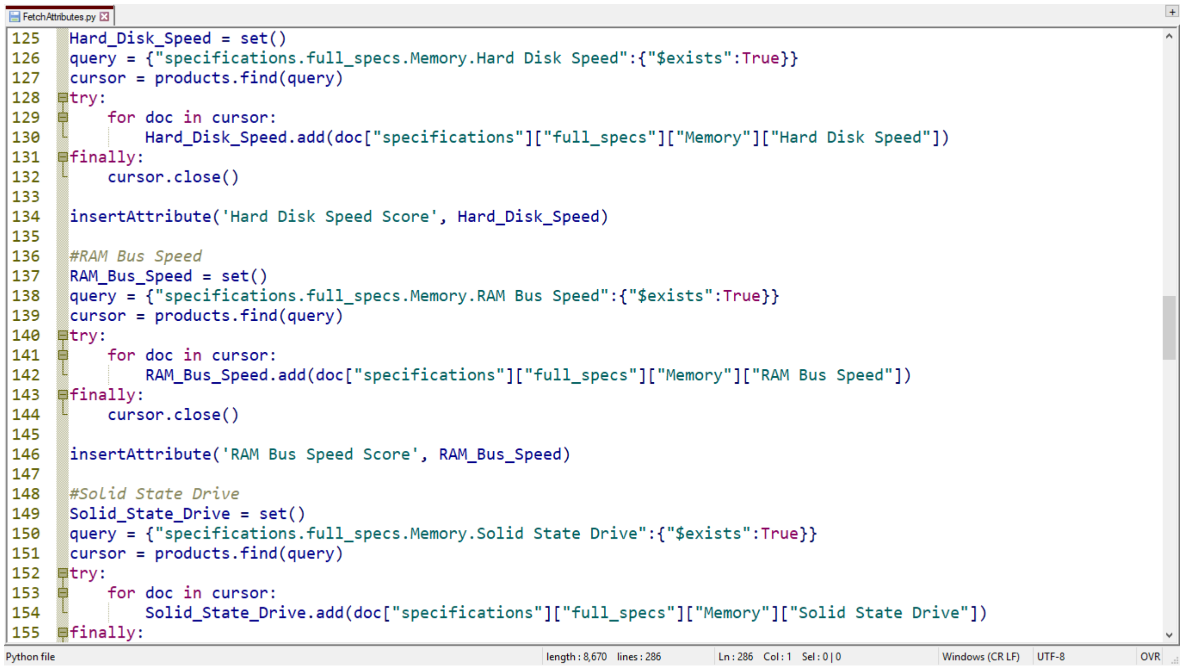

FetchAttributes.py

```python
CPU_Score = set()
query = {"specifications.full_specs.Processor.CPU Score":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        CPU_Score.add(doc["specifications"]["full_specs"]["Processor"]["CPU Score"])
finally:
    cursor.close()

insertAttribute('CPU Score Score', CPU_Score)

#################################Memory################################

#RAM
RAM = set()
query = {"specifications.full_specs.Memory.RAM":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        RAM.add(doc["specifications"]["full_specs"]["Memory"]["RAM"])
finally:
    cursor.close()

insertAttribute('RAM Score', RAM)

#Maximum Ram Supported
Maximum_Ram_Supported = set()
query = {"specifications.full_specs.Memory.Maximum RAM Supported":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
```

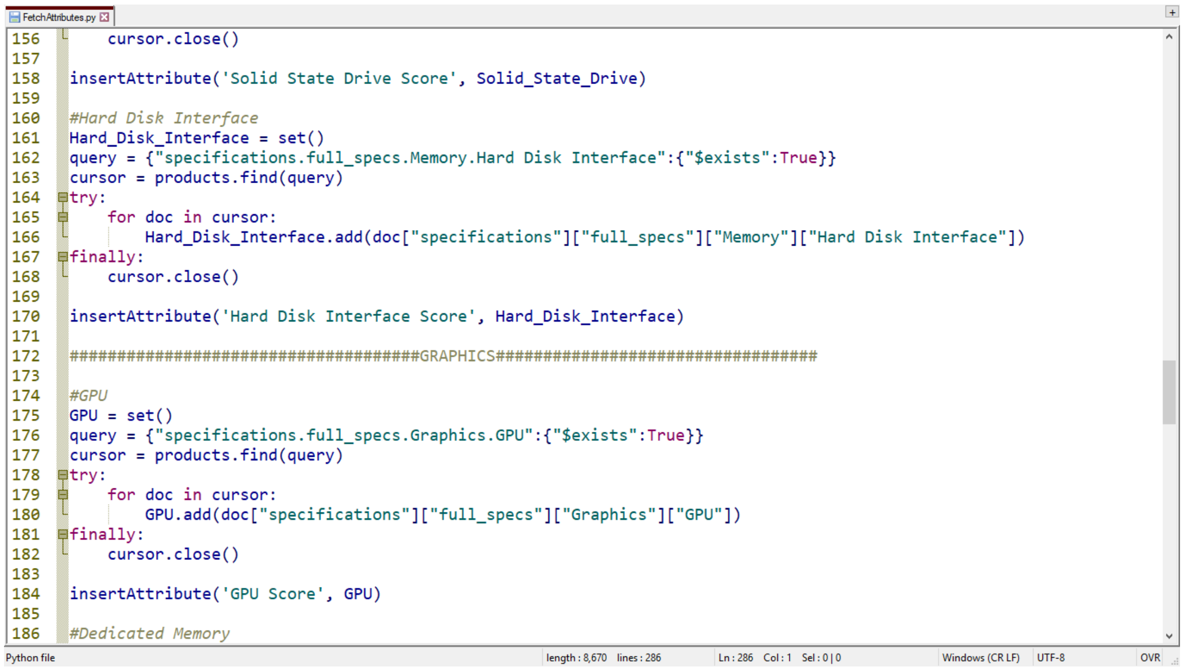
Python file   length : 8,670   lines : 286   Ln : 286   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR



```
            Maximum_Ram_Supported.add(doc["specifications"]["full_specs"]["Memory"]["Maximum RAM Supported"])
finally:
    cursor.close()

insertAttribute('Maximum Ram Supported Score', Maximum_Ram_Supported)

#RAM Slots
RAM_Slots = set()
query = {"specifications.full_specs.Memory.RAM Slots":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        RAM_Slots.add(doc["specifications"]["full_specs"]["Memory"]["RAM Slots"])
finally:
    cursor.close()

insertAttribute('RAM Slots Score', RAM_Slots)

#Hard Disk Capacity
Hard_Disk_Capacity = set()
query = {"specifications.full_specs.Memory.Hard Disk Capacity":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Hard_Disk_Capacity.add(doc["specifications"]["full_specs"]["Memory"]["Hard Disk Capacity"])
finally:
    cursor.close()

insertAttribute('Hard Disk Capacity Score', Hard_Disk_Capacity)

#Hard Disk Speed
```

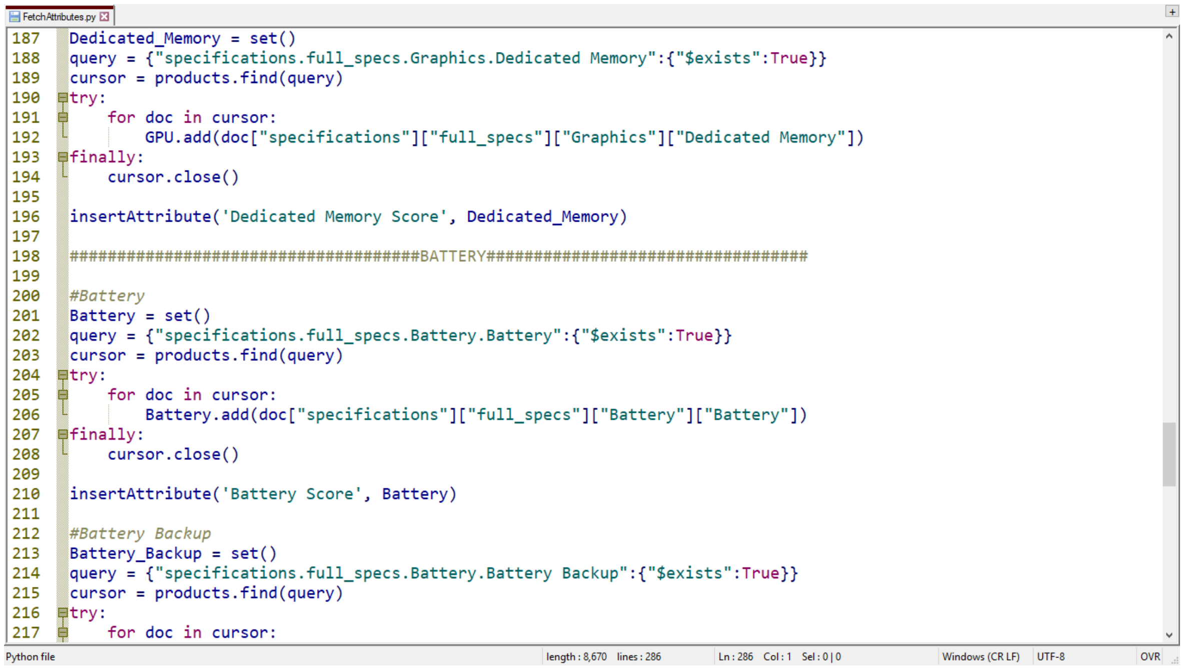


```
Hard_Disk_Speed = set()
query = {"specifications.full_specs.Memory.Hard Disk Speed":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Hard_Disk_Speed.add(doc["specifications"]["full_specs"]["Memory"]["Hard Disk Speed"])
finally:
    cursor.close()

insertAttribute('Hard Disk Speed Score', Hard_Disk_Speed)

#RAM Bus Speed
RAM_Bus_Speed = set()
query = {"specifications.full_specs.Memory.RAM Bus Speed":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        RAM_Bus_Speed.add(doc["specifications"]["full_specs"]["Memory"]["RAM Bus Speed"])
finally:
    cursor.close()

insertAttribute('RAM Bus Speed Score', RAM_Bus_Speed)

#Solid State Drive
Solid_State_Drive = set()
query = {"specifications.full_specs.Memory.Solid State Drive":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Solid_State_Drive.add(doc["specifications"]["full_specs"]["Memory"]["Solid State Drive"])
finally:
```


FetchAttributes.py

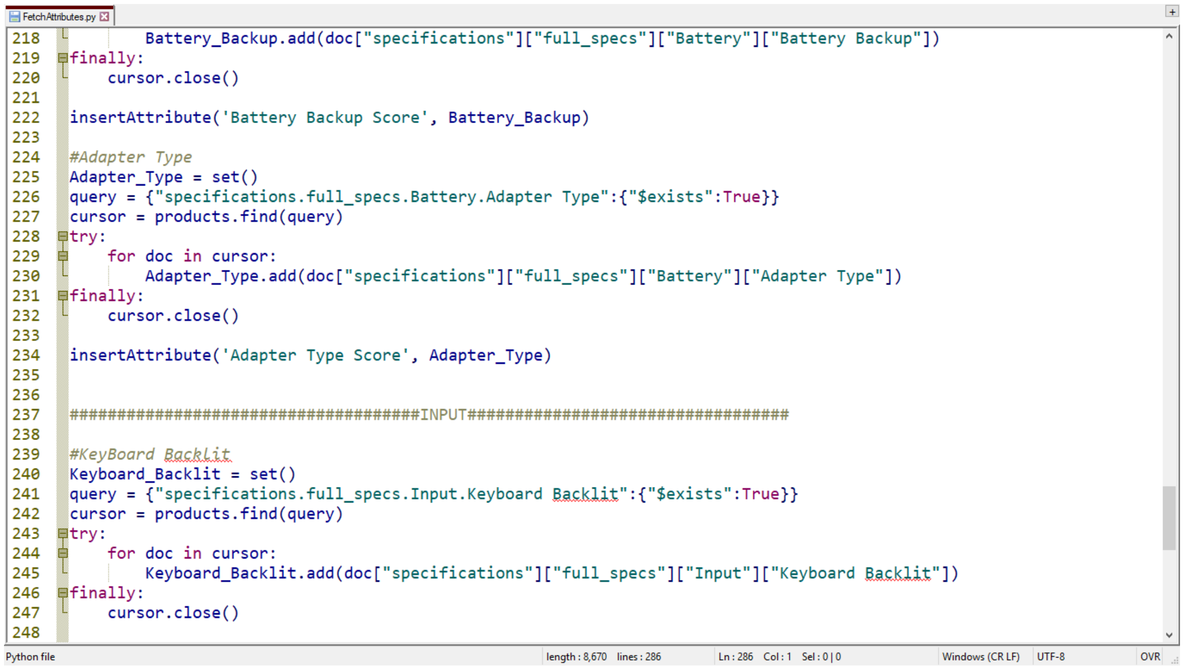
```
    cursor.close()

insertAttribute('Solid State Drive Score', Solid_State_Drive)

#Hard Disk Interface
Hard_Disk_Interface = set()
query = {"specifications.full_specs.Memory.Hard Disk Interface":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Hard_Disk_Interface.add(doc["specifications"]["full_specs"]["Memory"]["Hard Disk Interface"])
finally:
    cursor.close()

insertAttribute('Hard Disk Interface Score', Hard_Disk_Interface)

#####################################GRAPHICS##################################

#GPU
GPU = set()
query = {"specifications.full_specs.Graphics.GPU":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        GPU.add(doc["specifications"]["full_specs"]["Graphics"]["GPU"])
finally:
    cursor.close()

insertAttribute('GPU Score', GPU)

#Dedicated Memory
```

Python file length : 8,670 lines : 286 Ln : 286 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 OVR


FetchAttributes.py

```
Dedicated_Memory = set()
query = {"specifications.full_specs.Graphics.Dedicated Memory":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        GPU.add(doc["specifications"]["full_specs"]["Graphics"]["Dedicated Memory"])
finally:
    cursor.close()

insertAttribute('Dedicated Memory Score', Dedicated_Memory)

#####################################BATTERY##################################

#Battery
Battery = set()
query = {"specifications.full_specs.Battery.Battery":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Battery.add(doc["specifications"]["full_specs"]["Battery"]["Battery"])
finally:
    cursor.close()

insertAttribute('Battery Score', Battery)

#Battery Backup
Battery_Backup = set()
query = {"specifications.full_specs.Battery.Battery Backup":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
```

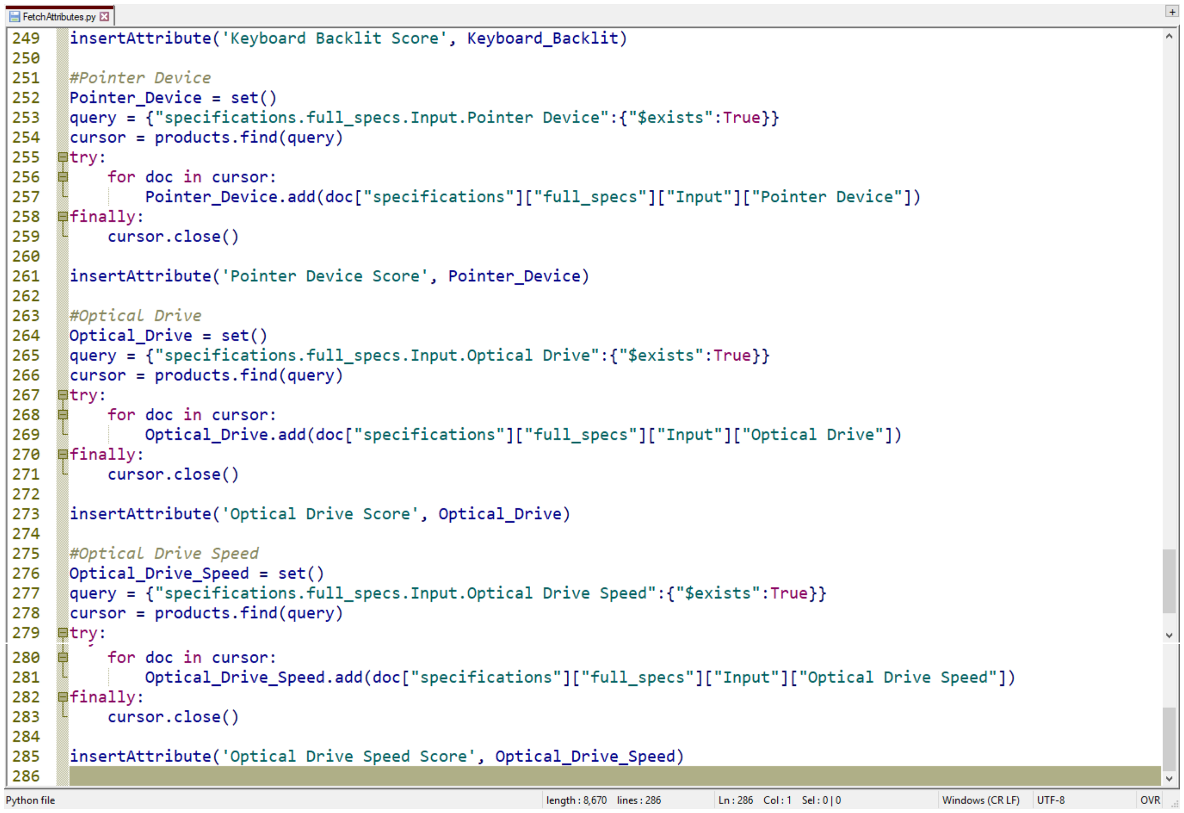
Python file length : 8,670 lines : 286 Ln : 286 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 OVR



FetchAttributes.py

```
        Battery_Backup.add(doc["specifications"]["full_specs"]["Battery"]["Battery Backup"])
finally:
    cursor.close()

insertAttribute('Battery Backup Score', Battery_Backup)

#Adapter Type
Adapter_Type = set()
query = {"specifications.full_specs.Battery.Adapter Type":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Adapter_Type.add(doc["specifications"]["full_specs"]["Battery"]["Adapter Type"])
finally:
    cursor.close()

insertAttribute('Adapter Type Score', Adapter_Type)


#####################################INPUT##################################

#KeyBoard Backlit
Keyboard_Backlit = set()
query = {"specifications.full_specs.Input.Keyboard Backlit":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Keyboard_Backlit.add(doc["specifications"]["full_specs"]["Input"]["Keyboard Backlit"])
finally:
    cursor.close()

```

Python file   length : 8,670   lines : 286   Ln : 286   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR

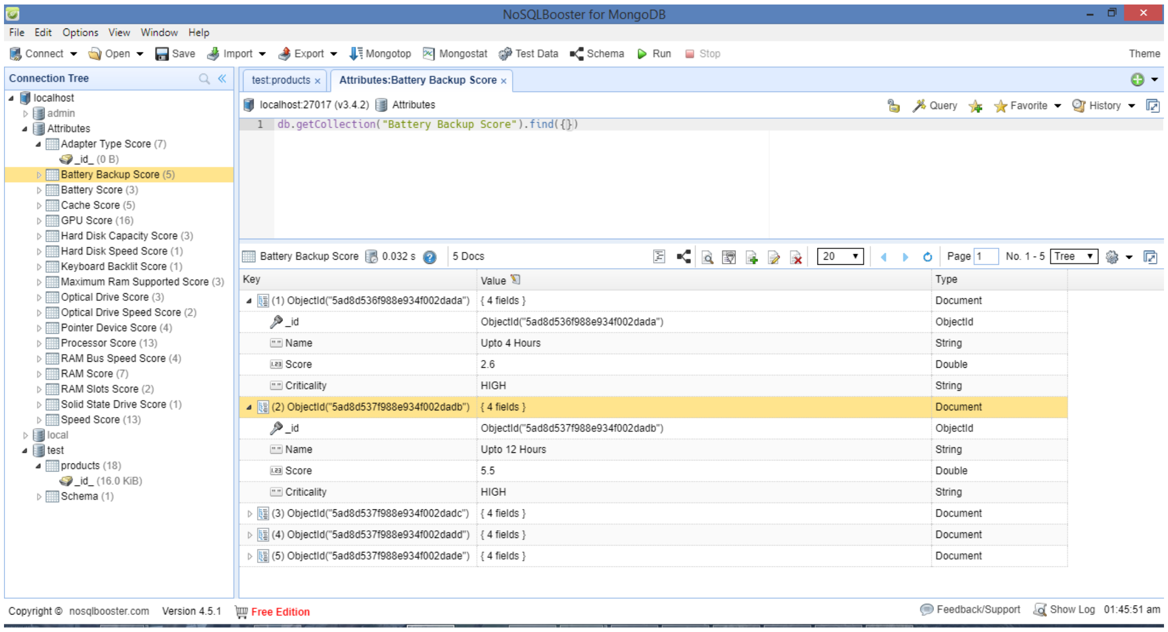


FetchAttributes.py

```
insertAttribute('Keyboard Backlit Score', Keyboard_Backlit)

#Pointer Device
Pointer_Device = set()
query = {"specifications.full_specs.Input.Pointer Device":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Pointer_Device.add(doc["specifications"]["full_specs"]["Input"]["Pointer Device"])
finally:
    cursor.close()

insertAttribute('Pointer Device Score', Pointer_Device)

#Optical Drive
Optical_Drive = set()
query = {"specifications.full_specs.Input.Optical Drive":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Optical_Drive.add(doc["specifications"]["full_specs"]["Input"]["Optical Drive"])
finally:
    cursor.close()

insertAttribute('Optical Drive Score', Optical_Drive)

#Optical Drive Speed
Optical_Drive_Speed = set()
query = {"specifications.full_specs.Input.Optical Drive Speed":{"$exists":True}}
cursor = products.find(query)
try:
    for doc in cursor:
        Optical_Drive_Speed.add(doc["specifications"]["full_specs"]["Input"]["Optical Drive Speed"])
finally:
    cursor.close()

insertAttribute('Optical Drive Speed Score', Optical_Drive_Speed)

```

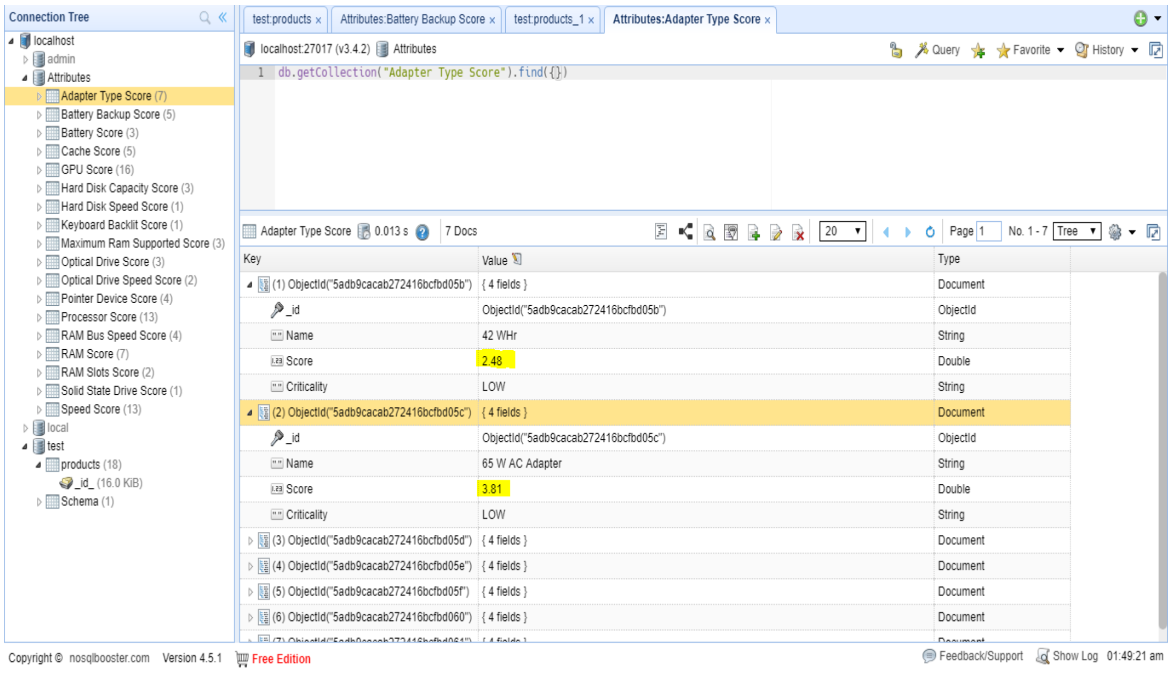
Python file   length : 8,670   lines : 286   Ln : 286   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR

Fig (5) Fetch Attributes Algorithm


NoSQLBooster for MongoDB
File Edit Options View Window Help
Connect Open Save Import Export Mongotop Mongostat Test Data Schema Run Stop Theme
Connection Tree
localhost
admin
Attributes
Adapter Type Score (7)
_id_ (0 B)
Battery Backup Score (5)
Battery Score (3)
Cache Score (5)
GPU Score (16)
Hard Disk Capacity Score (3)
Hard Disk Speed Score (1)
Keyboard Backlit Score (1)
Maximum Ram Supported Score (3)
Optical Drive Score (3)
Optical Drive Speed Score (2)
Pointer Device Score (4)
Processor Score (13)
RAM Bus Speed Score (4)
RAM Score (7)
RAM Slots Score (2)
Solid State Drive Score (1)
Speed Score (13)
local
test
products (18)
_id_ (16.0 KiB)
Schema (1)
test:products
Attributes:Battery Backup Score
localhost:27017 (v3.4.2) Attributes
Query Favorite History
Battery Backup Score 0.032 s 5 Docs
20 Page 1 No. 1 - 5 Tree
Copyright © nosqlbooster.com Version 4.5.1 Free Edition
Feedback/Support Show Log 01:45:51 am

```
db.getCollection("Battery Backup Score").find({})
```

Key	Value	Type
(1) ObjectId("5ad8d536f988e934f002dada")	{ 4 fields }	Document
_id	ObjectId("5ad8d536f988e934f002dada")	ObjectId
Name	Upto 4 Hours	String
Score	2.6	Double
Criticality	HIGH	String
(2) ObjectId("5ad8d537f988e934f002dadb")	{ 4 fields }	Document
_id	ObjectId("5ad8d537f988e934f002dadb")	ObjectId
Name	Upto 12 Hours	String
Score	5.5	Double
Criticality	HIGH	String
(3) ObjectId("5ad8d537f988e934f002dadc")	{ 4 fields }	Document
(4) ObjectId("5ad8d537f988e934f002dadd")	{ 4 fields }	Document
(5) ObjectId("5ad8d537f988e934f002dade")	{ 4 fields }	Document



```
db.getCollection("Adapter Type Score").find({})
```

Key	Value	Type
(1) ObjectId("5adb9cacab272416bcfbd05b")	{ 4 fields }	Document
_id	ObjectId("5adb9cacab272416bcfbd05b")	ObjectId
Name	42 WHr	String
Score	2.48	Double
Criticality	LOW	String
(2) ObjectId("5adb9cacab272416bcfbd05c")	{ 4 fields }	Document
_id	ObjectId("5adb9cacab272416bcfbd05c")	ObjectId
Name	65 W AC Adapter	String
Score	3.81	Double
Criticality	LOW	String
(3) ObjectId("5adb9cacab272416bcfbd05d")	{ 4 fields }	Document
(4) ObjectId("5adb9cacab272416bcfbd05e")	{ 4 fields }	Document
(5) ObjectId("5adb9cacab272416bcfbd05f")	{ 4 fields }	Document
(6) ObjectId("5adb9cacab272416bcfbd060")	{ 4 fields }	Document
(7) ObjectId("5adb9cacab272416bcfbd061")	{ 4 fields }	Document

Fig (6) After fetching in database

4.2.1.2.2. Main Comparator Algorithm

In this the script is receiving upto 4 product keys and returning the respective product scores in JSON format. The following flow diagram illustrates its flow

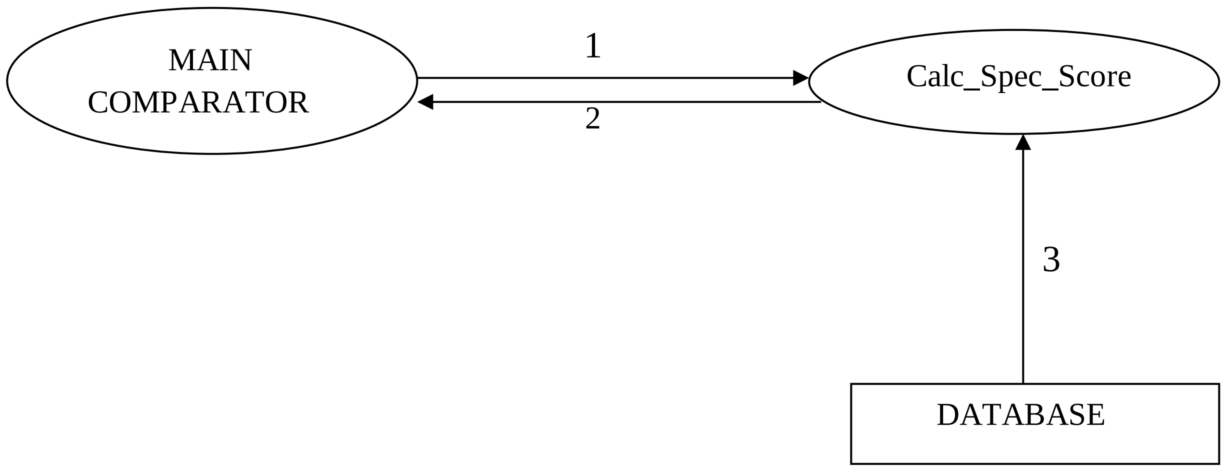
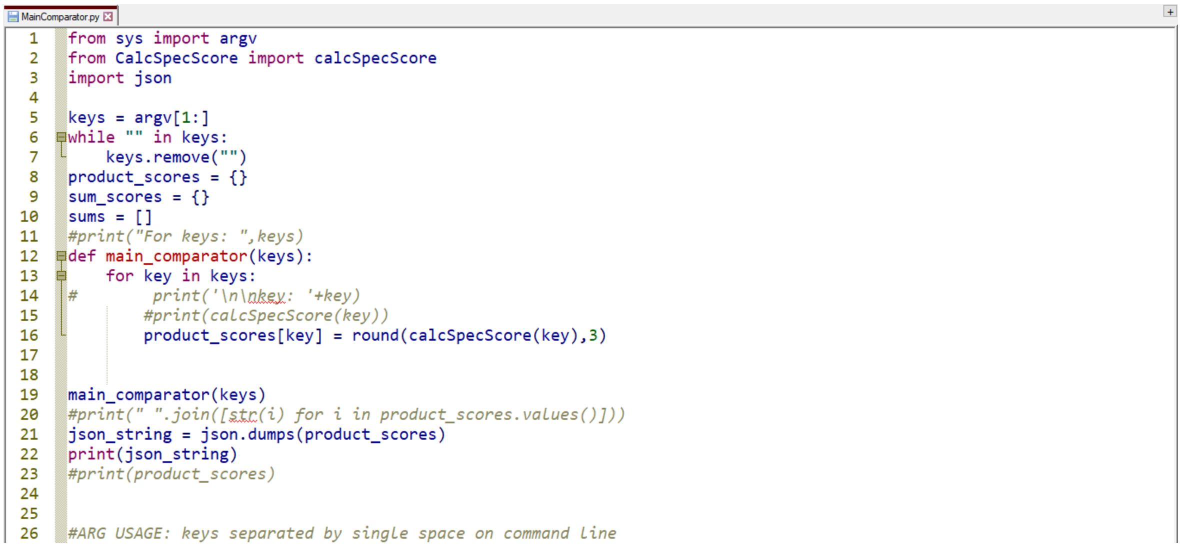


Fig (7) Main Comparator Flow Diagram

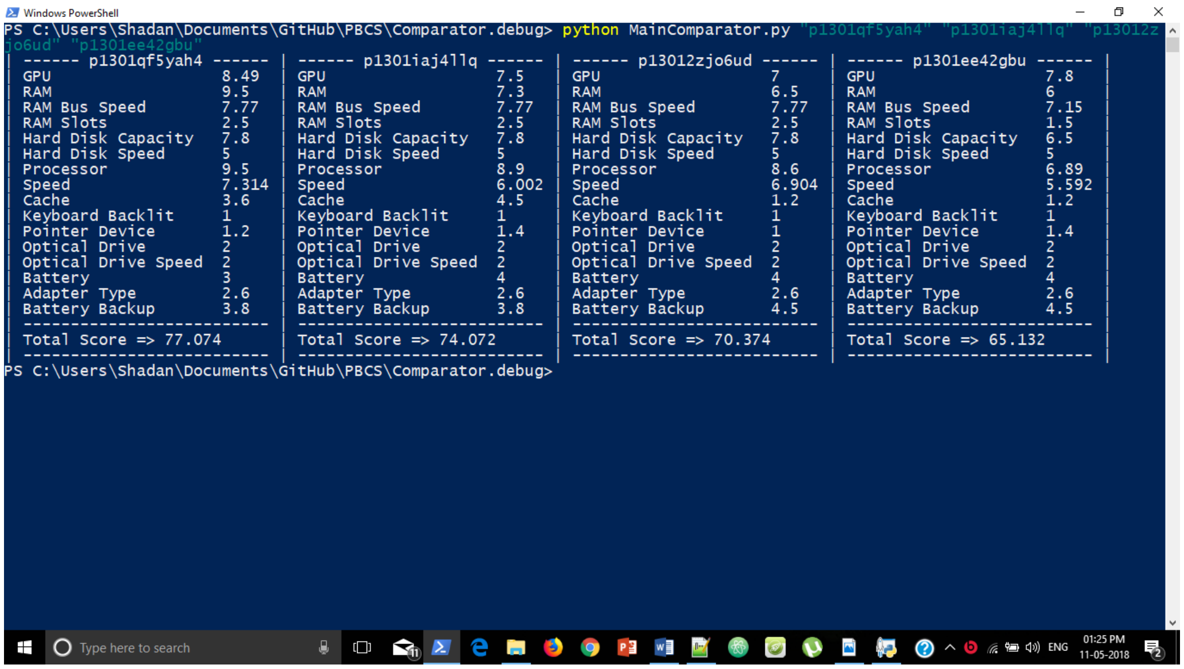
The number represent the following functions:

1. Requests specification score for each product key passed.
2. Specification score for respective product key returned.
3. Read values of attributes and returns score for each attributed.



```python
from sys import argv
from CalcSpecScore import calcSpecScore
import json

keys = argv[1:]
while "" in keys:
    keys.remove("")
product_scores = {}
sum_scores = {}
sums = []
#print("For keys: ",keys)
def main_comparator(keys):
    for key in keys:
#           print('\n\nkey: '+key)
        #print(calcSpecScore(key))
        product_scores[key] = round(calcSpecScore(key),3)


main_comparator(keys)
#print(" ".join([str(i) for i in product_scores.values()]))
json_string = json.dumps(product_scores)
print(json_string)
#print(product_scores)


#ARG USAGE: keys separated by single space on command line
```

Fig (8) Main Comparator

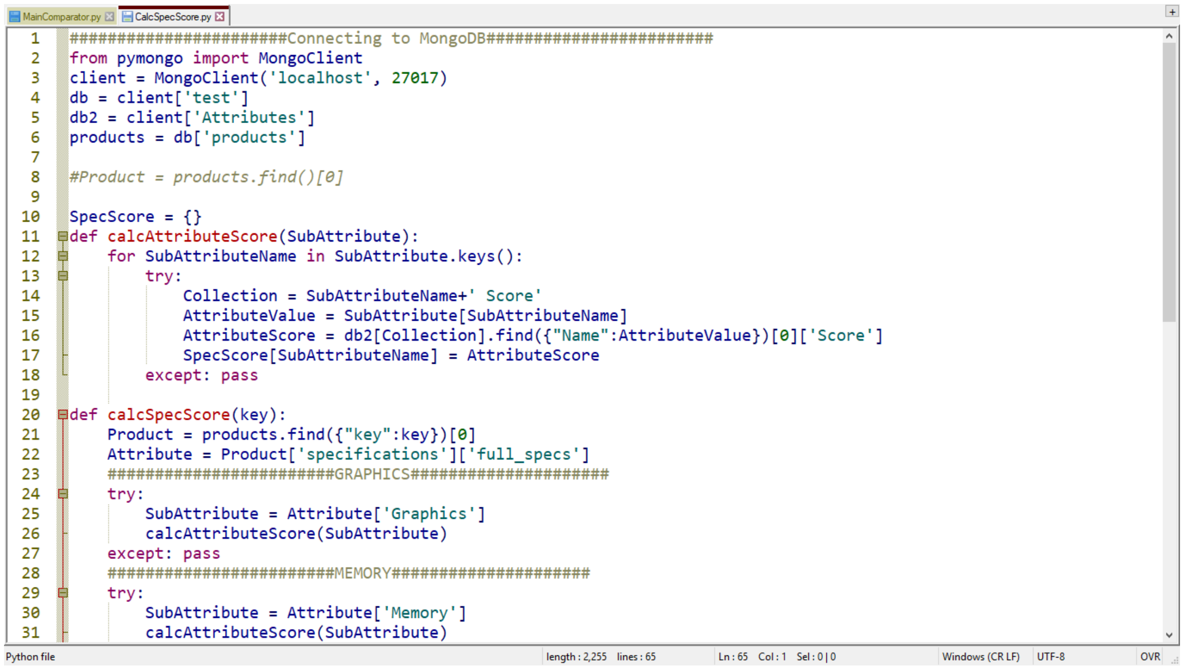


```
PS C:\Users\Shadan\Documents\GitHub\PBCS\Comparator.debug> python MainComparator.py "p1301qf5yah4" "p1301iaj4llq" "p13012z
jo6ud" "p1301ee42gbu"
| ------ p1301qf5yah4 ------  | ------ p1301iaj4llq ------  | ------ p13012zjo6ud ------  | ------ p1301ee42gbu ------  |
| GPU                  8.49   | GPU                  7.5    | GPU                  7      | GPU                  7.8    |
| RAM                  9.5    | RAM                  7.3    | RAM                  6.5    | RAM                  6      |
| RAM Bus Speed        7.77   | RAM Bus Speed        7.77   | RAM Bus Speed        7.77   | RAM Bus Speed        7.15   |
| RAM Slots            2.5    | RAM Slots            2.5    | RAM Slots            2.5    | RAM Slots            1.5    |
| Hard Disk Capacity   7.8    | Hard Disk Capacity   7.8    | Hard Disk Capacity   7.8    | Hard Disk Capacity   6.5    |
| Hard Disk Speed      5      | Hard Disk Speed      5      | Hard Disk Speed      5      | Hard Disk Speed      5      |
| Processor            9.5    | Processor            8.9    | Processor            8.6    | Processor            6.89   |
| Speed                7.314  | Speed                6.002  | Speed                6.904  | Speed                5.592  |
| Cache                3.6    | Cache                4.5    | Cache                1.2    | Cache                1.2    |
| Keyboard Backlit     1      | Keyboard Backlit     1      | Keyboard Backlit     1      | Keyboard Backlit     1      |
| Pointer Device       1.2    | Pointer Device       1.4    | Pointer Device       1      | Pointer Device       1.4    |
| Optical Drive        2      | Optical Drive        2      | Optical Drive        2      | Optical Drive        2      |
| Optical Drive Speed  2      | Optical Drive Speed  2      | Optical Drive Speed  2      | Optical Drive Speed  2      |
| Battery              3      | Battery              4      | Battery              4      | Battery              4      |
| Adapter Type         2.6    | Adapter Type         2.6    | Adapter Type         2.6    | Adapter Type         2.6    |
| Battery Backup       3.8    | Battery Backup       3.8    | Battery Backup       4.5    | Battery Backup       4.5    |
| --------------------------  | --------------------------  | --------------------------  | --------------------------  |
| Total Score => 77.074       | Total Score => 74.072       | Total Score => 70.374       | Total Score => 65.132       |
| --------------------------  | --------------------------  | --------------------------  | --------------------------  |
PS C:\Users\Shadan\Documents\GitHub\PBCS\Comparator.debug>
```

Fig (9) Main Comparator output

4.2.1.2.3 Calculate Specification Score Algorithm

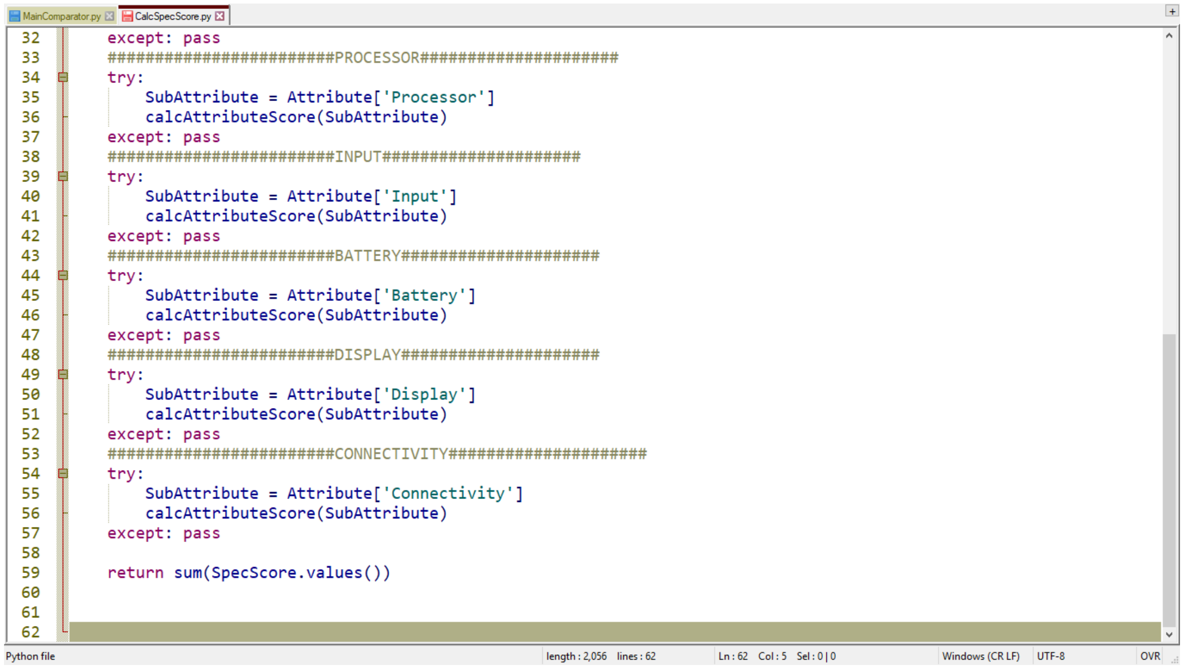
This algorithm is called to check the specification score for each of the attributes passed for the corresponding key and it returns the sum of scores corresponding to each value of attributes.



MainComparator.py | CalcSpecScore.py

```python
######################Connecting to MongoDB########################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['test']
db2 = client['Attributes']
products = db['products']

#Product = products.find()[0]

SpecScore = {}
def calcAttributeScore(SubAttribute):
    for SubAttributeName in SubAttribute.keys():
        try:
            Collection = SubAttributeName+' Score'
            AttributeValue = SubAttribute[SubAttributeName]
            AttributeScore = db2[Collection].find({"Name":AttributeValue})[0]['Score']
            SpecScore[SubAttributeName] = AttributeScore
        except: pass

def calcSpecScore(key):
    Product = products.find({"key":key})[0]
    Attribute = Product['specifications']['full_specs']
    ########################GRAPHICS#####################
    try:
        SubAttribute = Attribute['Graphics']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################MEMORY#####################
    try:
        SubAttribute = Attribute['Memory']
        calcAttributeScore(SubAttribute)
```

Python file   length : 2,255   lines : 65   Ln : 65   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR



MainComparator.py | CalcSpecScore.py

```python
    except: pass
    ########################PROCESSOR#####################
    try:
        SubAttribute = Attribute['Processor']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################INPUT#####################
    try:
        SubAttribute = Attribute['Input']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################BATTERY#####################
    try:
        SubAttribute = Attribute['Battery']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################DISPLAY#####################
    try:
        SubAttribute = Attribute['Display']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################CONNECTIVITY#####################
    try:
        SubAttribute = Attribute['Connectivity']
        calcAttributeScore(SubAttribute)
    except: pass

    return sum(SpecScore.values())
```

Python file   length : 2,056   lines : 62   Ln : 62   Col : 5   Sel : 0 | 0   Windows (CR LF)   UTF-8   OVR

Fig (10) Calculation of specification score

4.2.1.2.4 Prioritise Algorithm

The flow diagram of this algorithm is as follows:

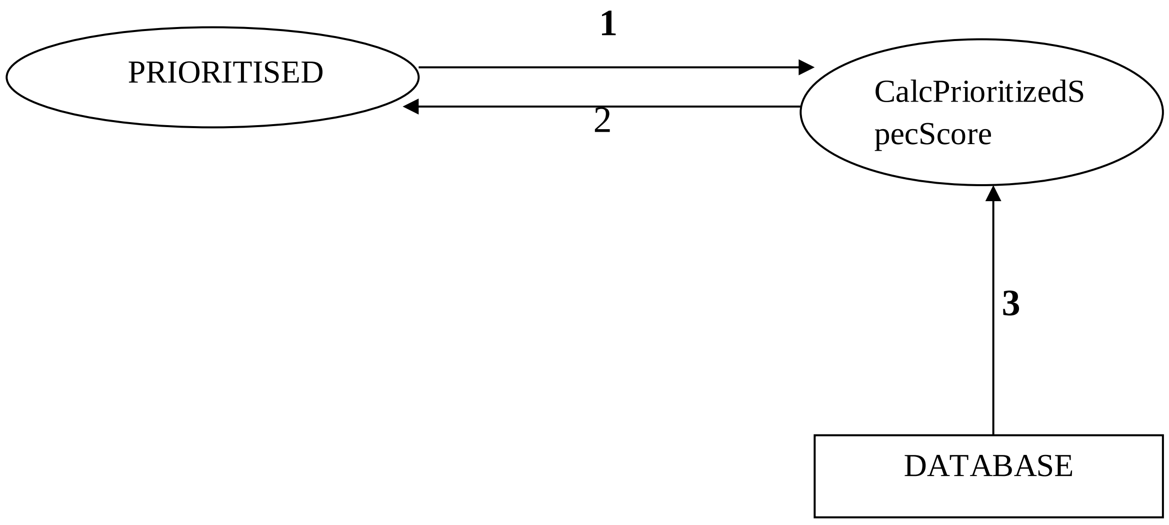
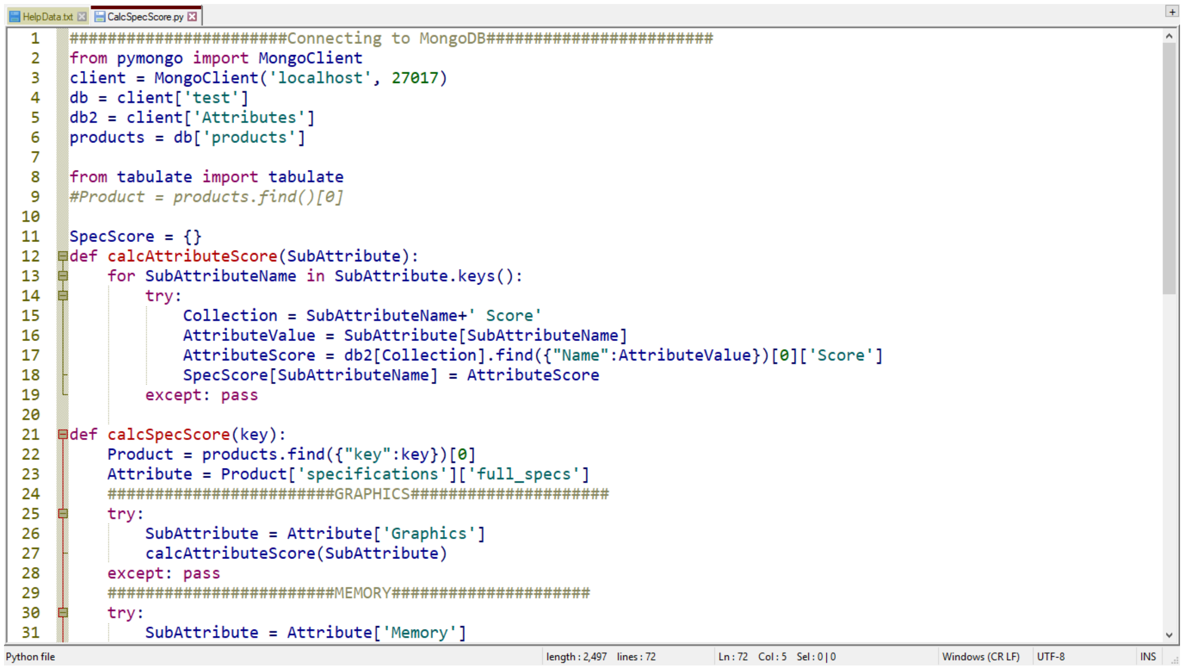


Fig (11) Prioritised Flow diagram

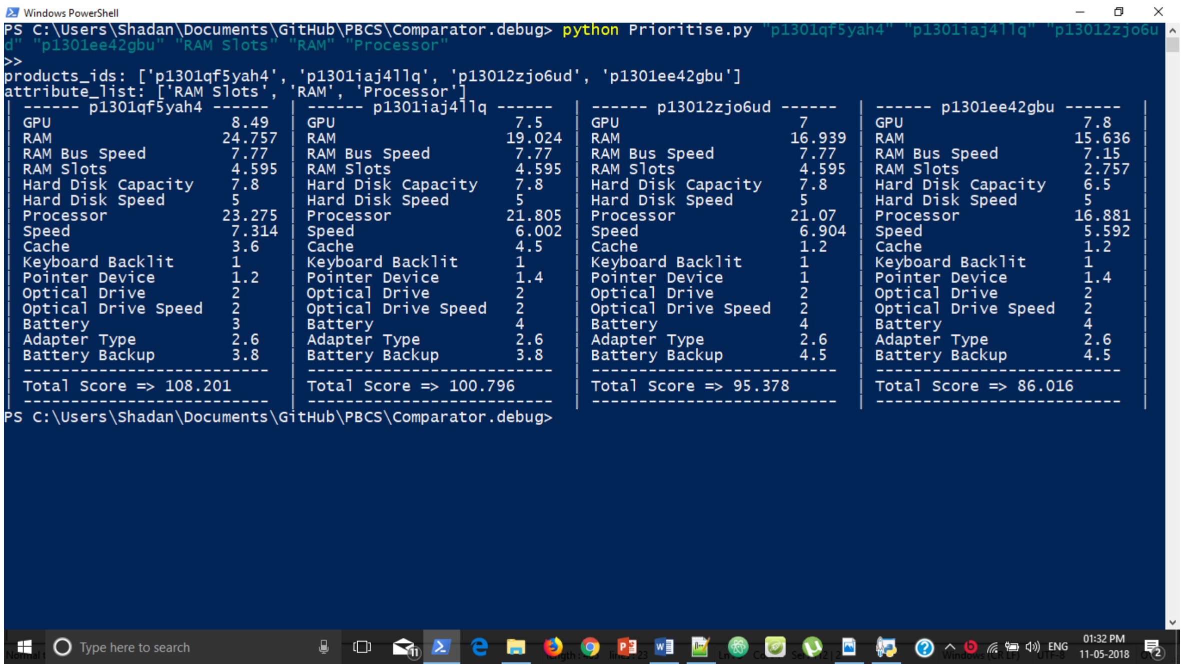
The number represent the following functions:

1. Requests specification score for each product key and product key passed.
2. Prioritised score for respective product key returned.
3. Read values of attributes and returns score for each attributed.



```python
#######################Connecting to MongoDB########################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['test']
db2 = client['Attributes']
products = db['products']

from tabulate import tabulate
#Product = products.find()[0]

SpecScore = {}
def calcAttributeScore(SubAttribute):
    for SubAttributeName in SubAttribute.keys():
        try:
            Collection = SubAttributeName+' Score'
            AttributeValue = SubAttribute[SubAttributeName]
            AttributeScore = db2[Collection].find({"Name":AttributeValue})[0]['Score']
            SpecScore[SubAttributeName] = AttributeScore
        except: pass

def calcSpecScore(key):
    Product = products.find({"key":key})[0]
    Attribute = Product['specifications']['full_specs']
    #######################GRAPHICS#####################
    try:
        SubAttribute = Attribute['Graphics']
        calcAttributeScore(SubAttribute)
    except: pass
    ########################MEMORY#####################
    try:
        SubAttribute = Attribute['Memory']
```

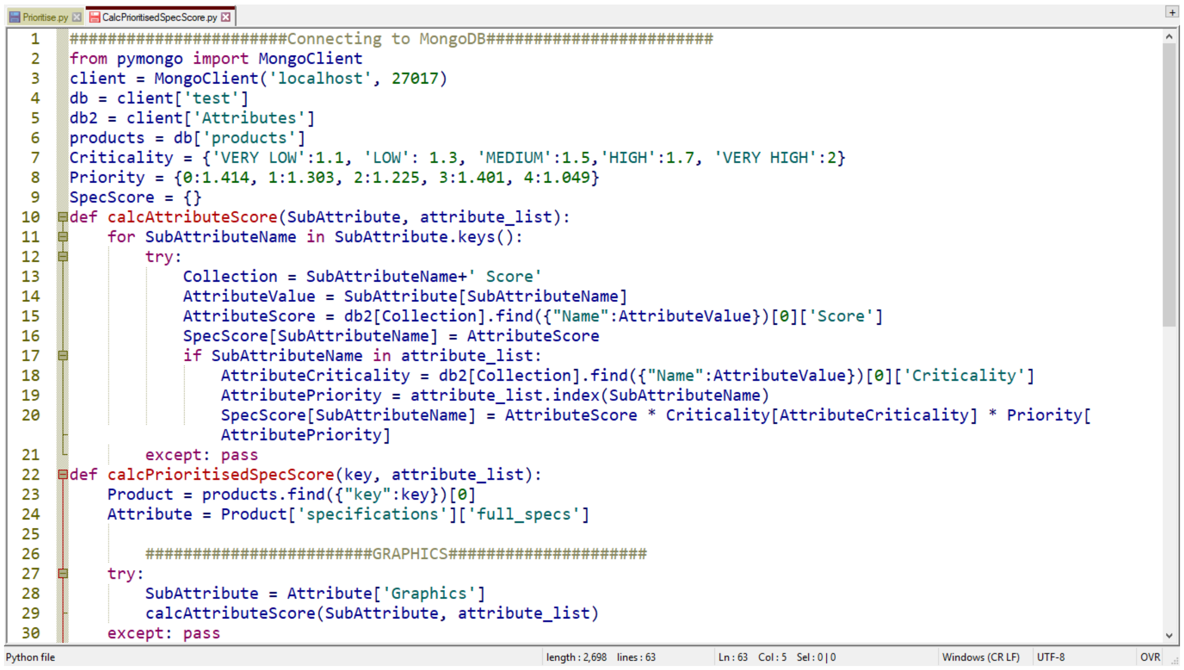
Fig (12) Prioritise



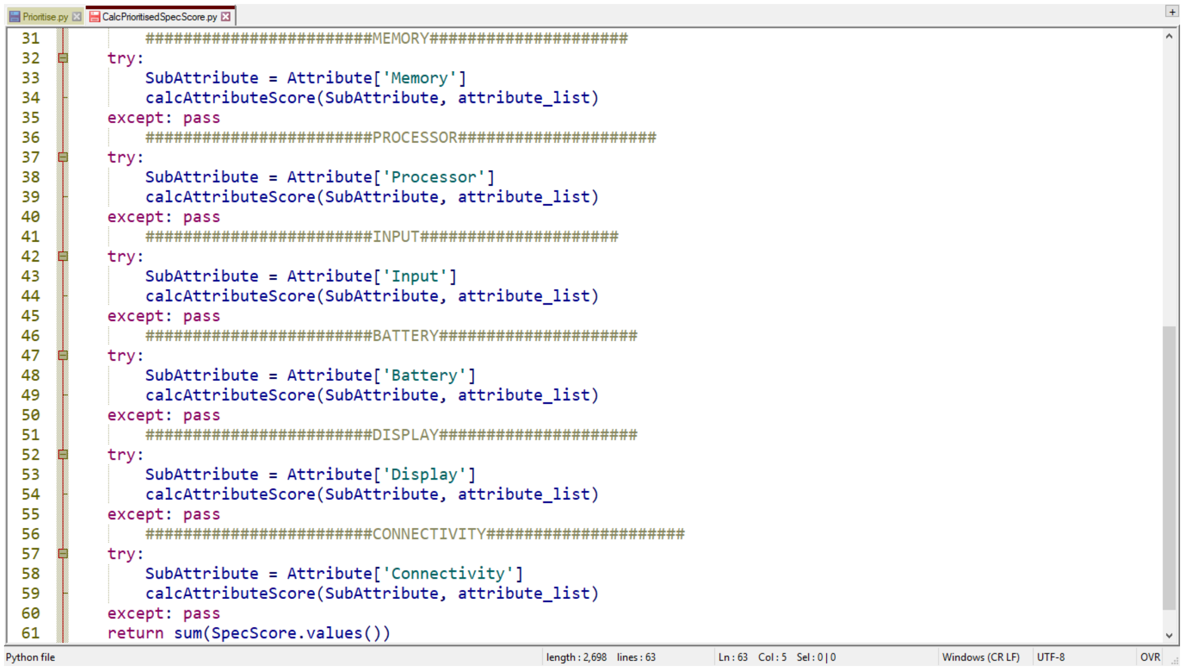
```
PS C:\Users\Shadan\Documents\GitHub\PBCS\Comparator.debug> python Prioritise.py "p1301qf5yah4" "p1301iaj4llq" "p13012zjo6u
d" "p1301ee42gbu" "RAM Slots" "RAM" "Processor"
>>
products_ids: ['p1301qf5yah4', 'p1301iaj4llq', 'p13012zjo6ud', 'p1301ee42gbu']
attribute_list: ['RAM Slots', 'RAM', 'Processor']
| ------ p1301qf5yah4 ------  | ------ p1301iaj4llq ------  | ------ p13012zjo6ud ------  | ------ p1301ee42gbu ------  |
| GPU                   8.49  | GPU                   7.5   | GPU                   7     | GPU                   7.8   |
| RAM                  24.757 | RAM                  19.024 | RAM                  16.939 | RAM                  15.636 |
| RAM Bus Speed         7.77  | RAM Bus Speed         7.77  | RAM Bus Speed         7.77  | RAM Bus Speed         7.15  |
| RAM Slots             4.595 | RAM Slots             4.595 | RAM Slots             4.595 | RAM Slots             2.757 |
| Hard Disk Capacity    7.8   | Hard Disk Capacity    7.8   | Hard Disk Capacity    7.8   | Hard Disk Capacity    6.5   |
| Hard Disk Speed       5     | Hard Disk Speed       5     | Hard Disk Speed       5     | Hard Disk Speed       5     |
| Processor            23.275 | Processor            21.805 | Processor            21.07  | Processor            16.881 |
| Speed                 7.314 | Speed                 6.002 | Speed                 6.904 | Speed                 5.592 |
| Cache                 3.6   | Cache                 4.5   | Cache                 1.2   | Cache                 1.2   |
| Keyboard Backlit      1     | Keyboard Backlit      1     | Keyboard Backlit      1     | Keyboard Backlit      1     |
| Pointer Device        1.2   | Pointer Device        1.4   | Pointer Device        1     | Pointer Device        1.4   |
| Optical Drive         2     | Optical Drive         2     | Optical Drive         2     | Optical Drive         2     |
| Optical Drive Speed   2     | Optical Drive Speed   2     | Optical Drive Speed   2     | Optical Drive Speed   2     |
| Battery               3     | Battery               4     | Battery               4     | Battery               4     |
| Adapter Type          2.6   | Adapter Type          2.6   | Adapter Type          2.6   | Adapter Type          2.6   |
| Battery Backup        3.8   | Battery Backup        3.8   | Battery Backup        4.5   | Battery Backup        4.5   |
| --------------------------  | --------------------------  | --------------------------  | --------------------------  |
| Total Score => 108.201      | Total Score => 100.796      | Total Score => 95.378       | Total Score => 86.016       |
| --------------------------  | --------------------------  | --------------------------  | --------------------------  |
PS C:\Users\Shadan\Documents\GitHub\PBCS\Comparator.debug>
```

Fig (13) Prioritised comparator output

4.2.1.2.5. Calculate Prioritised score


Prioritise.py
CalcPrioritisedSpecScore.py
Python file
length : 2,698 lines : 63
Ln : 63 Col : 5 Sel : 0 | 0
Windows (CR LF)
UTF-8
OVR

```python
######################Connecting to MongoDB#######################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['test']
db2 = client['Attributes']
products = db['products']
Criticality = {'VERY LOW':1.1, 'LOW': 1.3, 'MEDIUM':1.5,'HIGH':1.7, 'VERY HIGH':2}
Priority = {0:1.414, 1:1.303, 2:1.225, 3:1.401, 4:1.049}
SpecScore = {}
def calcAttributeScore(SubAttribute, attribute_list):
    for SubAttributeName in SubAttribute.keys():
        try:
            Collection = SubAttributeName+' Score'
            AttributeValue = SubAttribute[SubAttributeName]
            AttributeScore = db2[Collection].find({"Name":AttributeValue})[0]['Score']
            SpecScore[SubAttributeName] = AttributeScore
            if SubAttributeName in attribute_list:
                AttributeCriticality = db2[Collection].find({"Name":AttributeValue})[0]['Criticality']
                AttributePriority = attribute_list.index(SubAttributeName)
                SpecScore[SubAttributeName] = AttributeScore * Criticality[AttributeCriticality] * Priority[
                AttributePriority]
        except: pass
def calcPrioritisedSpecScore(key, attribute_list):
    Product = products.find({"key":key})[0]
    Attribute = Product['specifications']['full_specs']

        ########################GRAPHICS#####################
    try:
        SubAttribute = Attribute['Graphics']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
```


Prioritise.py
CalcPrioritisedSpecScore.py
Python file
length : 2,698 lines : 63
Ln : 63 Col : 5 Sel : 0 | 0
Windows (CR LF)
UTF-8
OVR

```python
        ########################MEMORY#####################
    try:
        SubAttribute = Attribute['Memory']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
        ########################PROCESSOR#####################
    try:
        SubAttribute = Attribute['Processor']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
        ########################INPUT#####################
    try:
        SubAttribute = Attribute['Input']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
        ########################BATTERY#####################
    try:
        SubAttribute = Attribute['Battery']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
        ########################DISPLAY#####################
    try:
        SubAttribute = Attribute['Display']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
        ########################CONNECTIVITY#####################
    try:
        SubAttribute = Attribute['Connectivity']
        calcAttributeScore(SubAttribute, attribute_list)
    except: pass
    return sum(SpecScore.values())
```

Fig (14) Calculate prioritized score

4.2.1.3. Update Module

The following flow diagram explains the update module functioning.

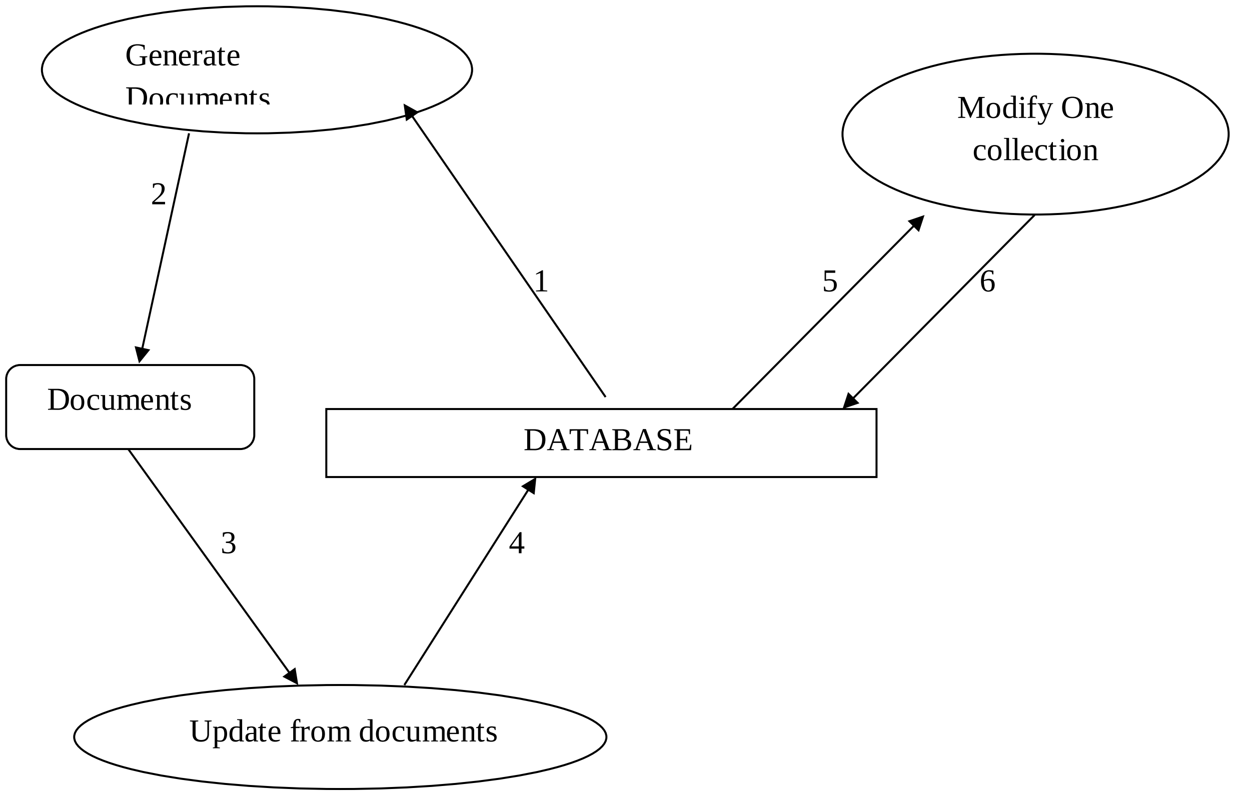


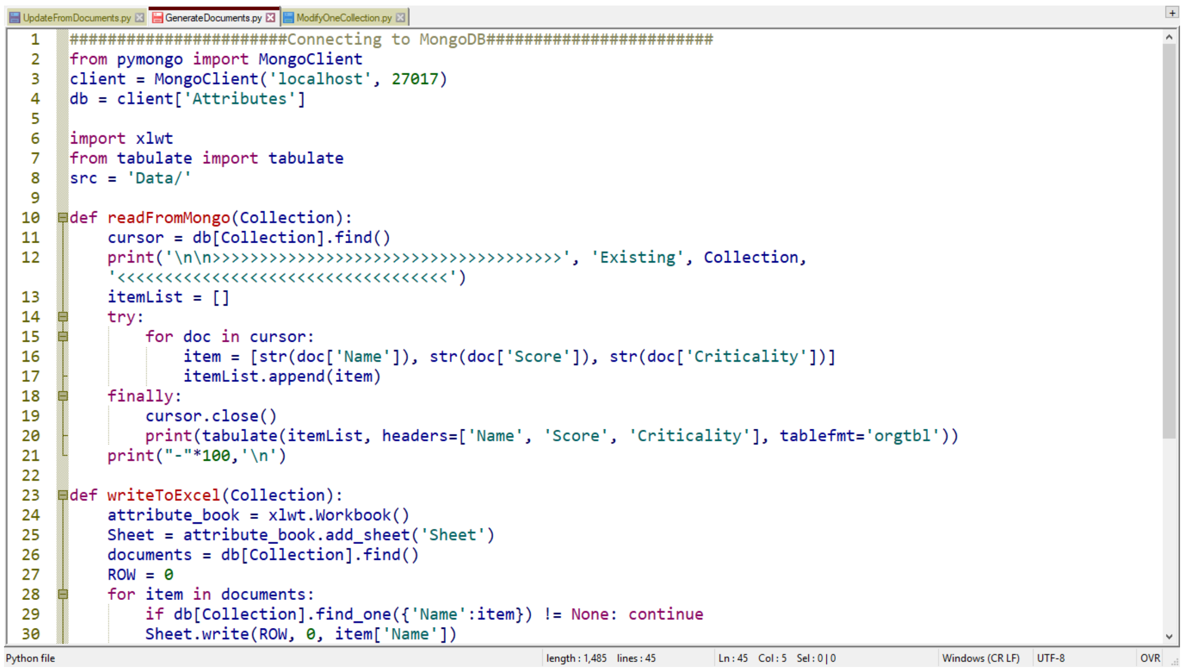
Fig (15) Update Flow Diagram

The number represent the following functions:

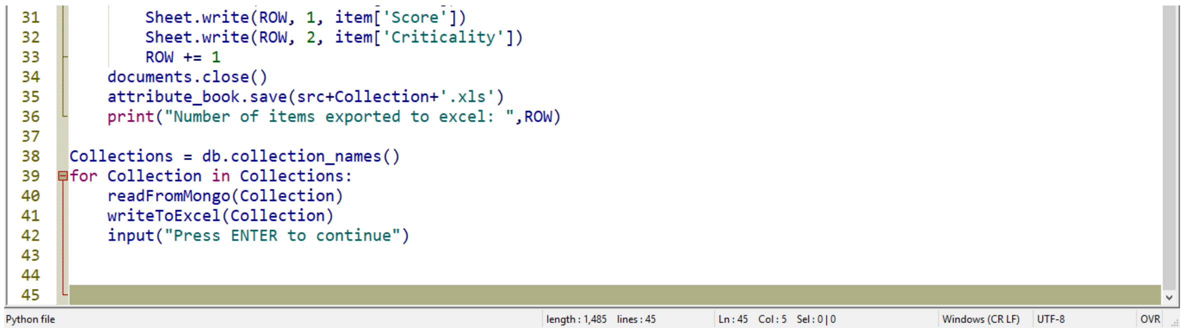
1. Read attribute collections.
2. Generate excel sheets of the respective collections.
3. Read from excel sheets.
4. Update the changes in the database.
5. Request collection data.
6. Save changes to the database.

4.2.1.3.1 Generate Documents

This algorithm fetches the attribute values for the attributes we have chosen along with their scores and criticality, and the generates excel sheets of the same which can be modified manually.



```python
#######################Connecting to MongoDB########################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['Attributes']

import xlwt
from tabulate import tabulate
src = 'Data/'

def readFromMongo(Collection):
    cursor = db[Collection].find()
    print('\n\n>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>', 'Existing', Collection, '<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<')
    itemList = []
    try:
        for doc in cursor:
            item = [str(doc['Name']), str(doc['Score']), str(doc['Criticality'])]
            itemList.append(item)
    finally:
        cursor.close()
        print(tabulate(itemList, headers=['Name', 'Score', 'Criticality'], tablefmt='orgtbl'))
    print("-"*100,'\n')

def writeToExcel(Collection):
    attribute_book = xlwt.Workbook()
    Sheet = attribute_book.add_sheet('Sheet')
    documents = db[Collection].find()
    ROW = 0
    for item in documents:
        if db[Collection].find_one({'Name':item}) != None: continue
        Sheet.write(ROW, 0, item['Name'])
```

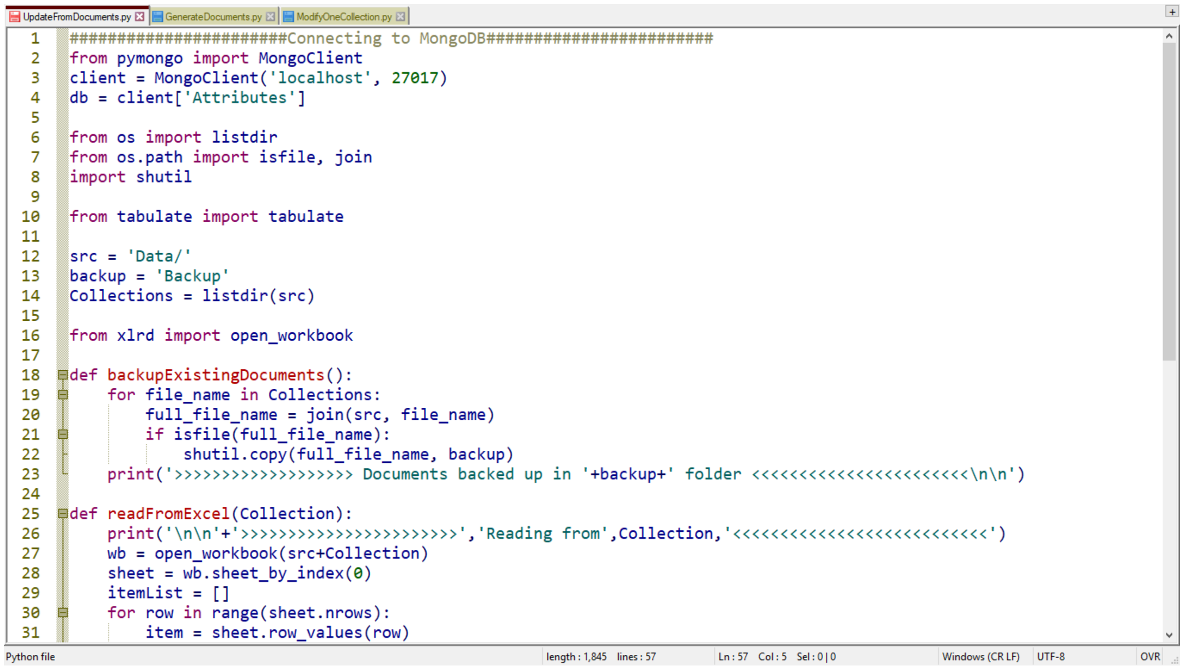


```python
        Sheet.write(ROW, 1, item['Score'])
        Sheet.write(ROW, 2, item['Criticality'])
        ROW += 1
    documents.close()
    attribute_book.save(src+Collection+'.xls')
    print("Number of items exported to excel: ",ROW)

Collections = db.collection_names()
for Collection in Collections:
    readFromMongo(Collection)
    writeToExcel(Collection)
    input("Press ENTER to continue")


```

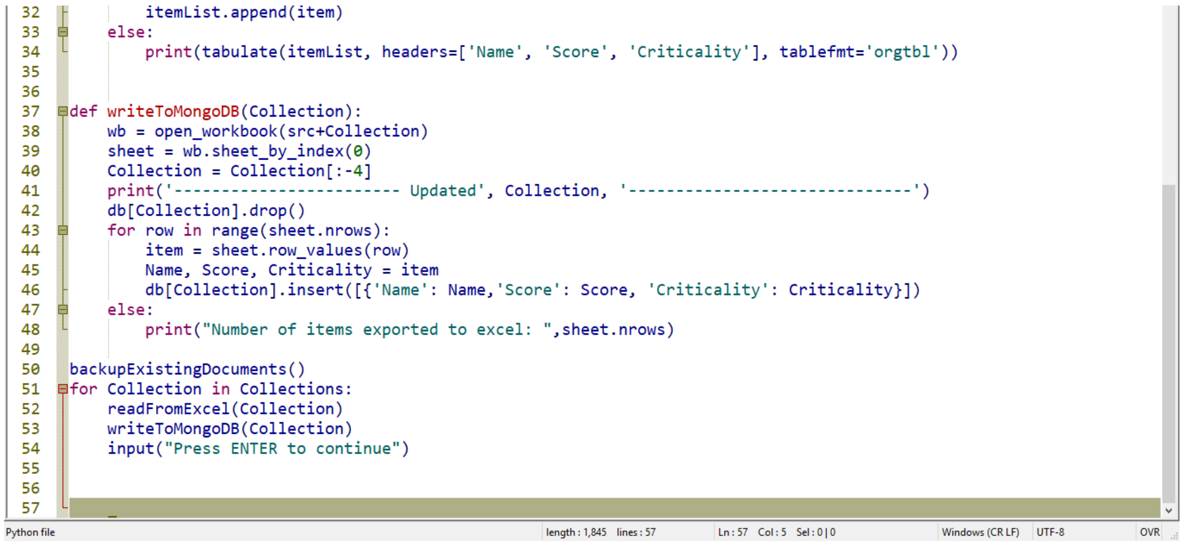
Fig (16) Generate documents

4.2.1.3.2. Update from documents

The data from the modified excel sheets is fetched and pushed into the database resulting in the updation of attribute values and criticality for the various attributes.



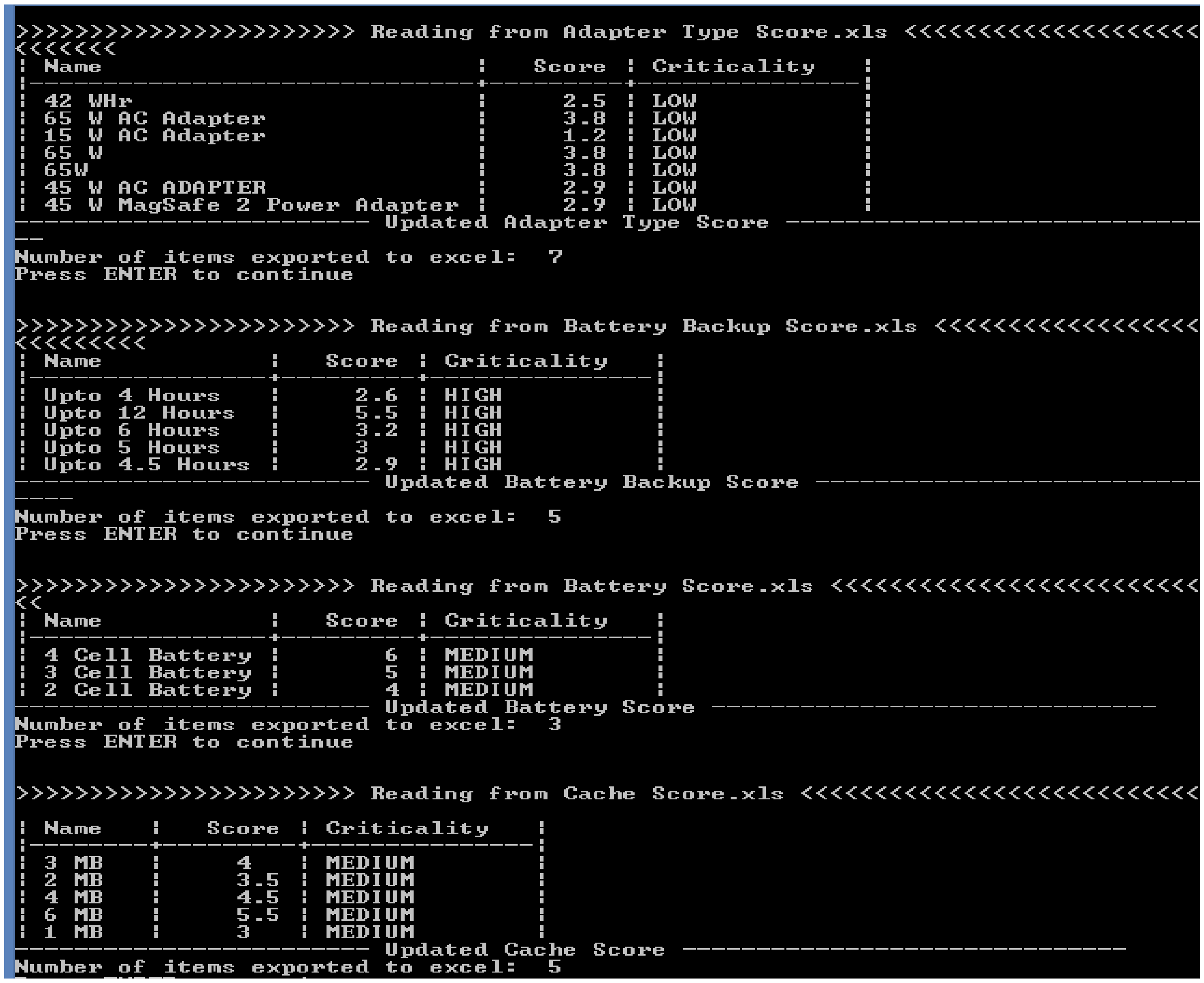
```python
#######################Connecting to MongoDB########################
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['Attributes']

from os import listdir
from os.path import isfile, join
import shutil

from tabulate import tabulate

src = 'Data/'
backup = 'Backup'
Collections = listdir(src)

from xlrd import open_workbook

def backupExistingDocuments():
    for file_name in Collections:
        full_file_name = join(src, file_name)
        if isfile(full_file_name):
            shutil.copy(full_file_name, backup)
    print('>>>>>>>>>>>>>>>>>> Documents backed up in '+backup+' folder <<<<<<<<<<<<<<<<<<<<<<\n\n')

def readFromExcel(Collection):
    print('\n\n'+'>>>>>>>>>>>>>>>>>>>>>>','Reading from',Collection,'<<<<<<<<<<<<<<<<<<<<<<<<<<')
    wb = open_workbook(src+Collection)
    sheet = wb.sheet_by_index(0)
    itemList = []
    for row in range(sheet.nrows):
        item = sheet.row_values(row)
```



```python
        itemList.append(item)
    else:
        print(tabulate(itemList, headers=['Name', 'Score', 'Criticality'], tablefmt='orgtbl'))


def writeToMongoDB(Collection):
    wb = open_workbook(src+Collection)
    sheet = wb.sheet_by_index(0)
    Collection = Collection[:-4]
    print('---------------------- Updated', Collection, '----------------------------')
    db[Collection].drop()
    for row in range(sheet.nrows):
        item = sheet.row_values(row)
        Name, Score, Criticality = item
        db[Collection].insert([{'Name': Name,'Score': Score, 'Criticality': Criticality}])
    else:
        print("Number of items exported to excel: ",sheet.nrows)

backupExistingDocuments()
for Collection in Collections:
    readFromExcel(Collection)
    writeToMongoDB(Collection)
    input("Press ENTER to continue")


    _
```

Fig (17) Update from documents

The following figure shows the output window during updation. The results are freflected back in the database.

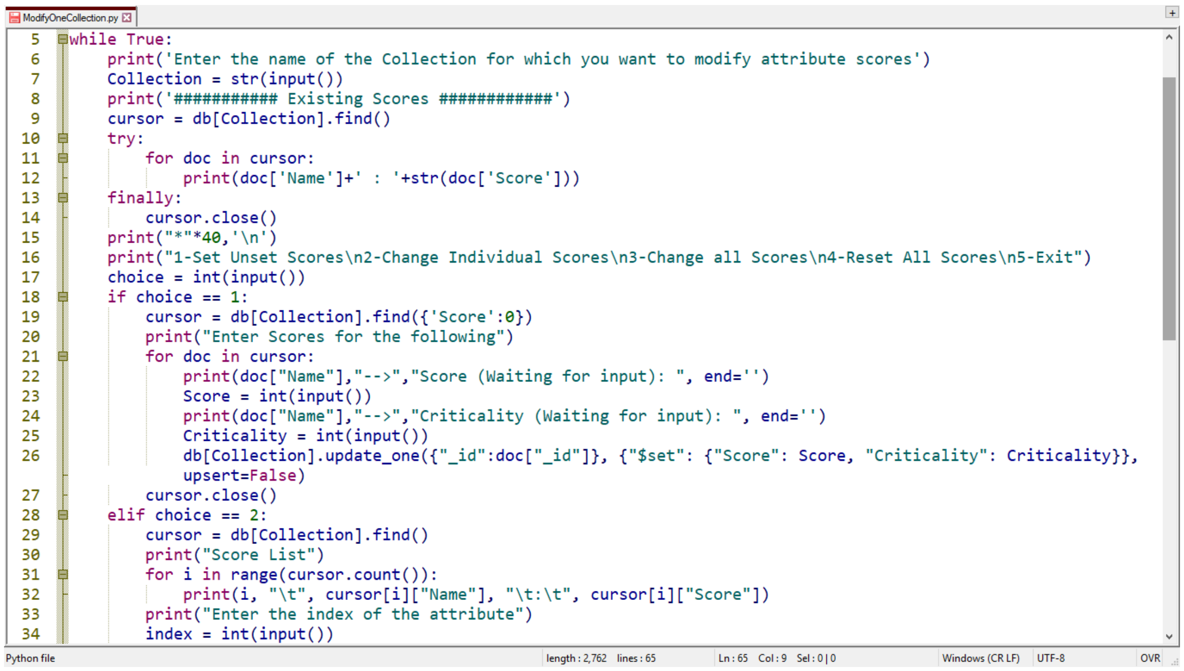


```
>>>>>>>>>>>>>>>>>>>>>>> Reading from Adapter Type Score.xls <<<<<<<<<<<<<<<<<<<<
<<<<<<<
| Name                            |   Score | Criticality    |
|---------------------------------+---------+----------------|
| 42 WHr                          |     2.5 | LOW            |
| 65 W AC Adapter                 |     3.8 | LOW            |
| 15 W AC Adapter                 |     1.2 | LOW            |
| 65 W                            |     3.8 | LOW            |
| 65W                             |     3.8 | LOW            |
| 45 W AC ADAPTER                 |     2.9 | LOW            |
| 45 W MagSafe 2 Power Adapter |     2.9 | LOW            |
------------------------ Updated Adapter Type Score -----------------------------
--
Number of items exported to excel:  7
Press ENTER to continue

>>>>>>>>>>>>>>>>>>>>>>> Reading from Battery Backup Score.xls <<<<<<<<<<<<<<<<<<
<<<<<<<<<
| Name             |   Score | Criticality    |
|------------------+---------+----------------|
| Upto 4 Hours     |     2.6 | HIGH           |
| Upto 12 Hours    |     5.5 | HIGH           |
| Upto 6 Hours     |     3.2 | HIGH           |
| Upto 5 Hours     |     3   | HIGH           |
| Upto 4.5 Hours   |     2.9 | HIGH           |
------------------------ Updated Battery Backup Score ---------------------------
----

Number of items exported to excel:  5
Press ENTER to continue

>>>>>>>>>>>>>>>>>>>>>>> Reading from Battery Score.xls <<<<<<<<<<<<<<<<<<<<<<<<<
<<
| Name             |   Score | Criticality    |
|------------------+---------+----------------|
| 4 Cell Battery |       6 | MEDIUM         |
| 3 Cell Battery |       5 | MEDIUM         |
| 2 Cell Battery |       4 | MEDIUM         |
------------------------ Updated Battery Score ------------------------------
Number of items exported to excel:  3
Press ENTER to continue

>>>>>>>>>>>>>>>>>>>>>>> Reading from Cache Score.xls <<<<<<<<<<<<<<<<<<<<<<<<<<<

| Name    |   Score | Criticality    |
|---------+---------+----------------|
| 3 MB    |     4   | MEDIUM         |
| 2 MB    |     3.5 | MEDIUM         |
| 4 MB    |     4.5 | MEDIUM         |
| 6 MB    |     5.5 | MEDIUM         |
| 1 MB    |     3   | MEDIUM         |
------------------------ Updated Cache Score ------------------------------
Number of items exported to excel:  5
```

Fig (18) After updation

4.2.1.3.2. Modify from collection

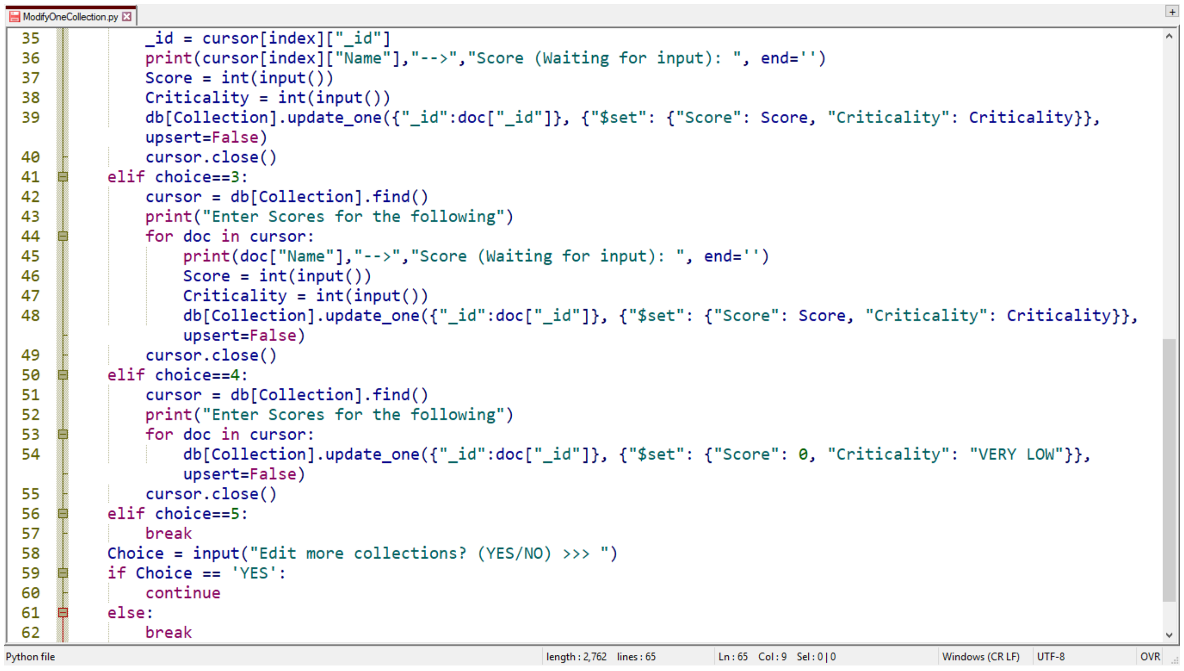
This algorithm modifies one attribute at a time. It does not generate any excel sheets. Instead, it access and changes score and criticality of the attributes directly into the database.



ModifyOneCollection.py

```python
while True:
    print('Enter the name of the Collection for which you want to modify attribute scores')
    Collection = str(input())
    print('########### Existing Scores ############')
    cursor = db[Collection].find()
    try:
        for doc in cursor:
            print(doc['Name']+' : '+str(doc['Score']))
    finally:
        cursor.close()
    print("*"*40,'\n')
    print("1-Set Unset Scores\n2-Change Individual Scores\n3-Change all Scores\n4-Reset All Scores\n5-Exit")
    choice = int(input())
    if choice == 1:
        cursor = db[Collection].find({'Score':0})
        print("Enter Scores for the following")
        for doc in cursor:
            print(doc["Name"],"-->","Score (Waiting for input): ", end='')
            Score = int(input())
            print(doc["Name"],"-->","Criticality (Waiting for input): ", end='')
            Criticality = int(input())
            db[Collection].update_one({"_id":doc["_id"]}, {"$set": {"Score": Score, "Criticality": Criticality}},
            upsert=False)
        cursor.close()
    elif choice == 2:
        cursor = db[Collection].find()
        print("Score List")
        for i in range(cursor.count()):
            print(i, "\t", cursor[i]["Name"], "\t:\t", cursor[i]["Score"])
        print("Enter the index of the attribute")
        index = int(input())
```

Python file length : 2,762 lines : 65 Ln : 65 Col : 9 Sel : 0 | 0 Windows (CR LF) UTF-8 OVR



ModifyOneCollection.py

```python
        _id = cursor[index]["_id"]
        print(cursor[index]["Name"],"-->","Score (Waiting for input): ", end='')
        Score = int(input())
        Criticality = int(input())
        db[Collection].update_one({"_id":doc["_id"]}, {"$set": {"Score": Score, "Criticality": Criticality}},
        upsert=False)
        cursor.close()
    elif choice==3:
        cursor = db[Collection].find()
        print("Enter Scores for the following")
        for doc in cursor:
            print(doc["Name"],"-->","Score (Waiting for input): ", end='')
            Score = int(input())
            Criticality = int(input())
            db[Collection].update_one({"_id":doc["_id"]}, {"$set": {"Score": Score, "Criticality": Criticality}},
            upsert=False)
        cursor.close()
    elif choice==4:
        cursor = db[Collection].find()
        print("Enter Scores for the following")
        for doc in cursor:
            db[Collection].update_one({"_id":doc["_id"]}, {"$set": {"Score": 0, "Criticality": "VERY LOW"}},
            upsert=False)
        cursor.close()
    elif choice==5:
        break
    Choice = input("Edit more collections? (YES/NO) >>> ")
    if Choice == 'YES':
        continue
    else:
        break
```

Python file length : 2,762 lines : 65 Ln : 65 Col : 9 Sel : 0 | 0 Windows (CR LF) UTF-8 OVR

Fig (19) Modify from collection

4.3. System Testing

4.3.1. Functional Testing

This is a type of black-box the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

4.3.1.1 Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Testing cannot catch every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that a developer can use to verify a source code. After having exhausted all the options, there is no choice but to stop unit testing and merge the code segment with other units.

4.3.1.2. Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

4.3.1.3. Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas of the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Regression testing is important because of the following reasons:

- Minimize the gaps in testing when an application with changes made must be tested.
- Testing the new changes to verify that the changes made did not affect any other area of the application.
- Mitigates risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

4.4. Acceptance Testing

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

More ideas will be shared with the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application.

By performing acceptance tests on an application, the testing team will deduce how the application will perform in the production. There are also legal and contractual requirements for acceptance of the system.

Chapter 5

RESULT AND DISCUSSION

As discussed above the system will generate the result according to the prioritization of the user. The demand of the user is firstly being compared to the show the best result and then according to their priority, it is sorted with its specification score and displayed. PBCS now only provides the option for laptops only. The user can select multiple laptops for comparison which is according to their priority.

Fig (20) List of laptops

On the top of PBCS users have three options home, feedback, report. On clicking the Home, the user will directly navigate to the home page of PBCS. As we are only making it for laptops, so the number of various laptops will be shown on the screen. User will now have to select at least two laptops for comparison, which the user will select according to its priority and compare button will be clicked. After this process a new web page will be opened which will show the products detail with its multiple attributes and specification score.

Feedback allows the user to give feedback about the website.

Report allows the user to report the error about the laptop as well as any glitch in the website.

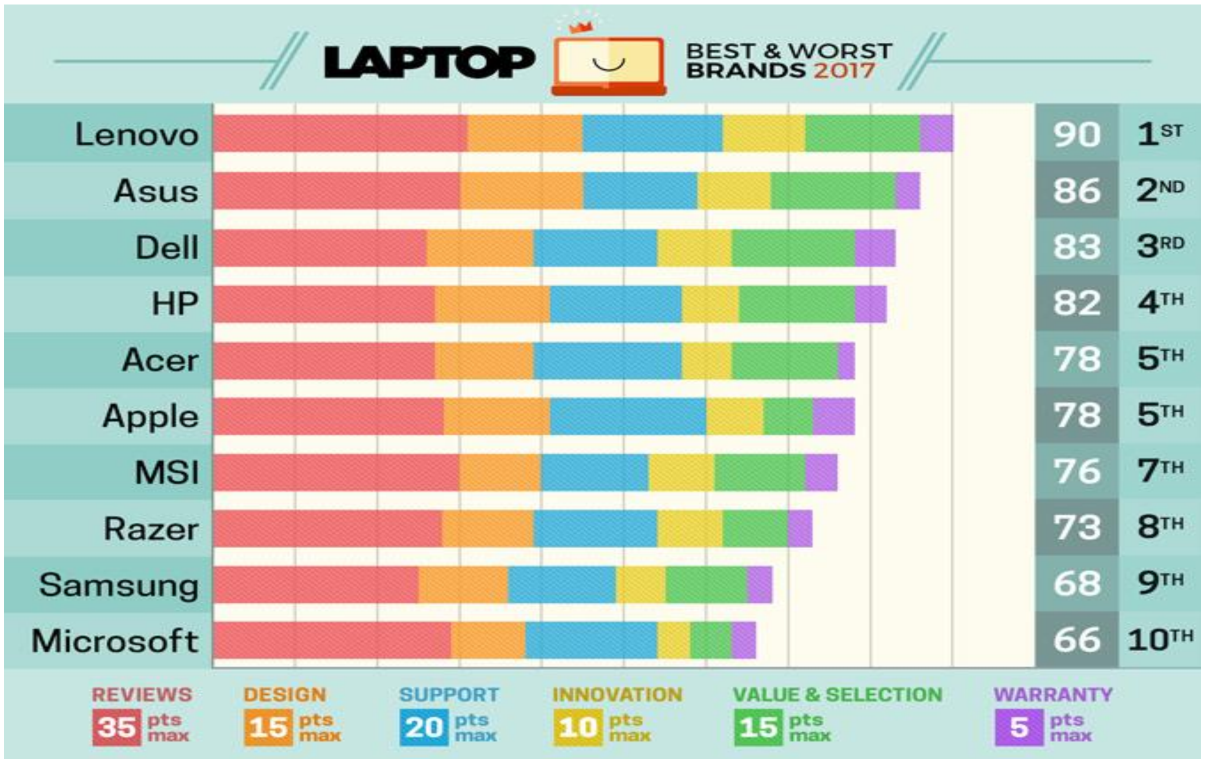


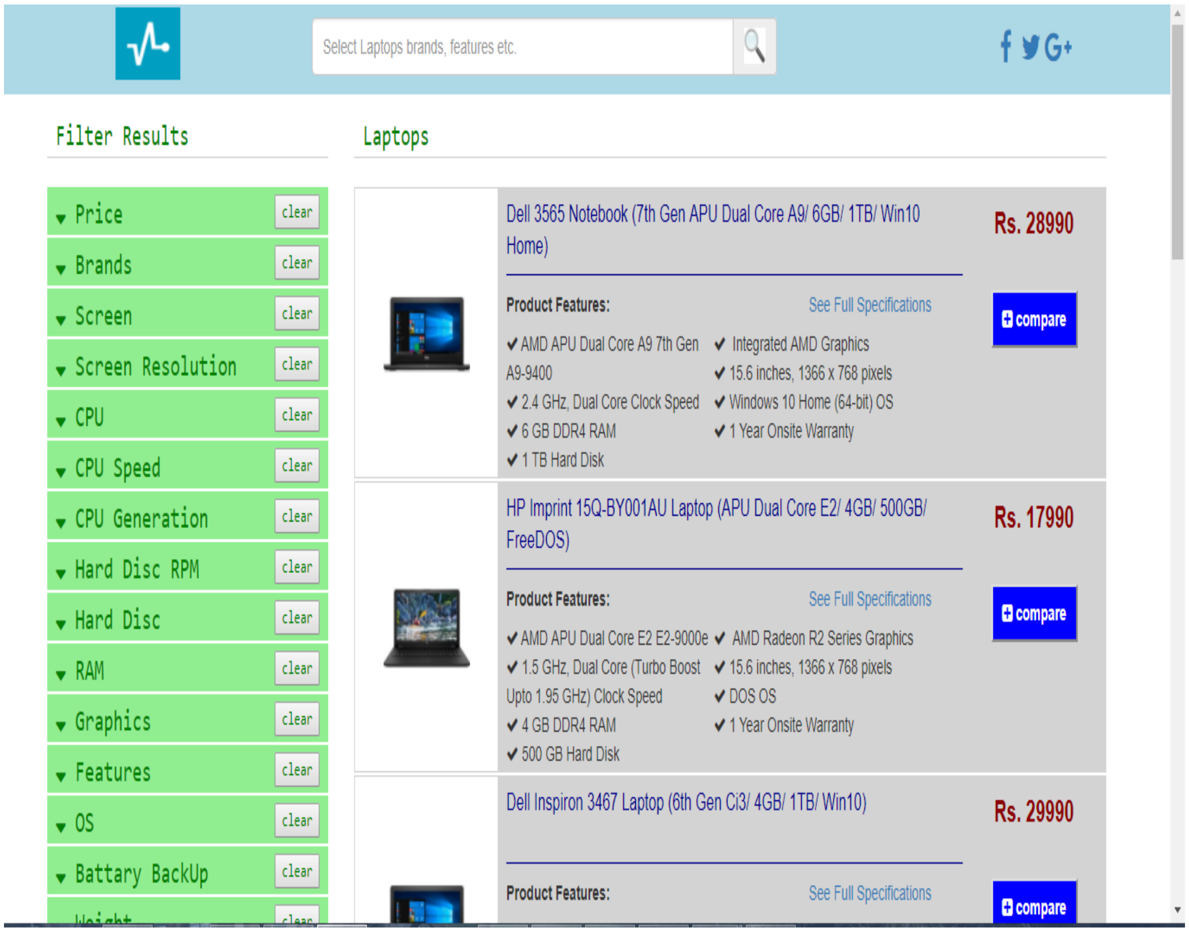
Fig (21) List of popular laptops

PBCS also has a row for newly launched laptops. The date of release decides which laptop is new. It has various laptops of various companies. After clicking on any laptop user will be able to see its full specification and details.

5.1 Graphical User Interface

The graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

Our project interface is as follows:



Dell 3565 Notebook (7th Gen APU Dual Core A9/ 6GB/ 1TB/ Win10 Home)

Rs. 28990

Product Features: See Full Specifications

compare

✔ AMD APU Dual Core A9 7th Gen A9-9400
✔ 2.4 GHz, Dual Core Clock Speed
✔ 6 GB DDR4 RAM
✔ 1 TB Hard Disk
✔ Integrated AMD Graphics
✔ 15.6 inches, 1366 x 768 pixels
✔ Windows 10 Home (64-bit) OS
✔ 1 Year Onsite Warranty

HP Imprint 15Q-BY001AU Laptop (APU Dual Core E2/ 4GB/ 500GB/ FreeDOS)

Rs. 17990

Product Features: See Full Specifications

compare

✔ AMD APU Dual Core E2 E2-9000e
✔ 1.5 GHz, Dual Core (Turbo Boost Upto 1.95 GHz) Clock Speed
✔ 4 GB DDR4 RAM
✔ 500 GB Hard Disk
✔ AMD Radeon R2 Series Graphics
✔ 15.6 inches, 1366 x 768 pixels
✔ DOS OS
✔ 1 Year Onsite Warranty

Dell Inspiron 3467 Laptop (6th Gen Ci3/ 4GB/ 1TB/ Win10)

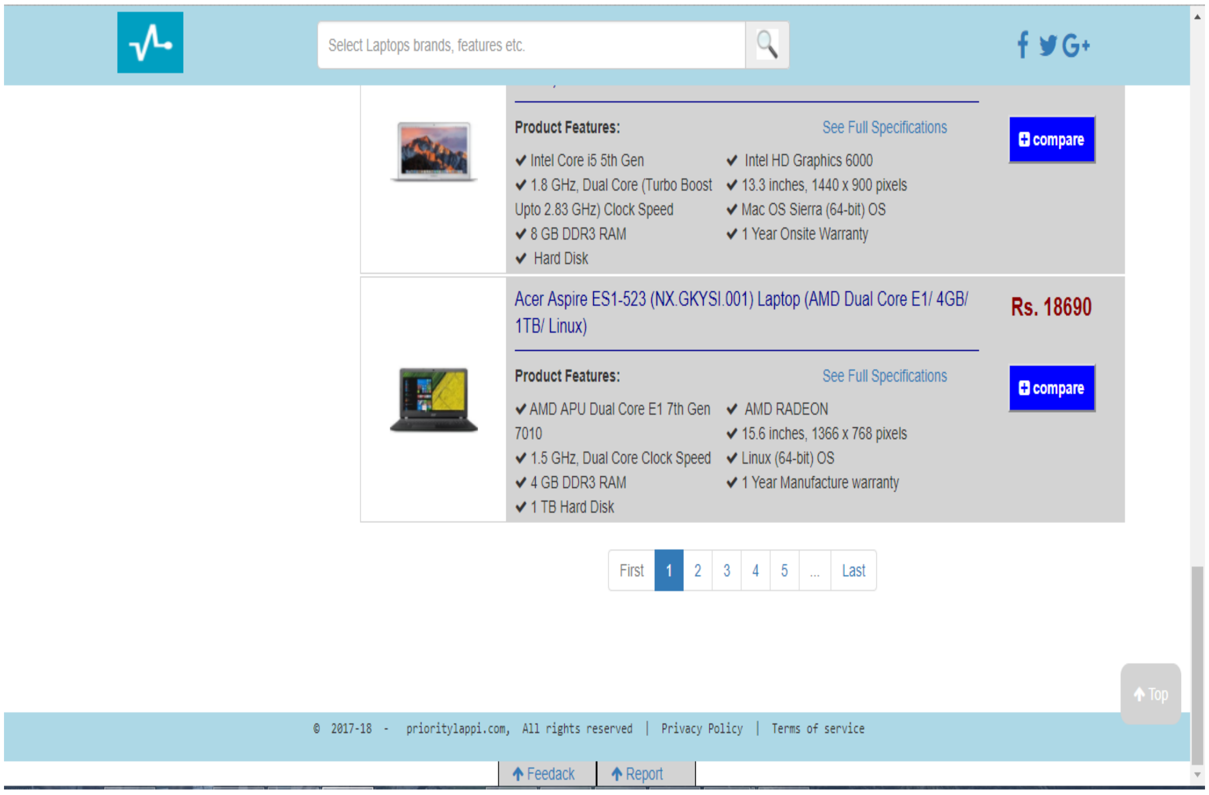
Rs. 29990

Product Features: See Full Specifications

compare

Fig (22) Home Page

The homepage will show all the laptops which are being scrapped from Smartprix, an e-commerce website. In our homepage sidebar includes the filter results. And in the main body, we have all our laptops. In the menu bar, we have made search bar which will help in searching the required laptops. The laptops are being shown according to the latest trend. And in the footer portion, we have the feedback and report option.



Select Laptops brands, features etc.

Product Features: See Full Specifications

✔ Intel Core i5 5th Gen
✔ 1.8 GHz, Dual Core (Turbo Boost Upto 2.83 GHz) Clock Speed
✔ 8 GB DDR3 RAM
✔ Hard Disk
✔ Intel HD Graphics 6000
✔ 13.3 inches, 1440 x 900 pixels
✔ Mac OS Sierra (64-bit) OS
✔ 1 Year Onsite Warranty

compare

Acer Aspire ES1-523 (NX.GKYSI.001) Laptop (AMD Dual Core E1/ 4GB/ 1TB/ Linux)

Rs. 18690

Product Features: See Full Specifications

✔ AMD APU Dual Core E1 7th Gen 7010
✔ 1.5 GHz, Dual Core Clock Speed
✔ 4 GB DDR3 RAM
✔ 1 TB Hard Disk
✔ AMD RADEON
✔ 15.6 inches, 1366 x 768 pixels
✔ Linux (64-bit) OS
✔ 1 Year Manufacture warranty

compare

First 1 2 3 4 5 ... Last

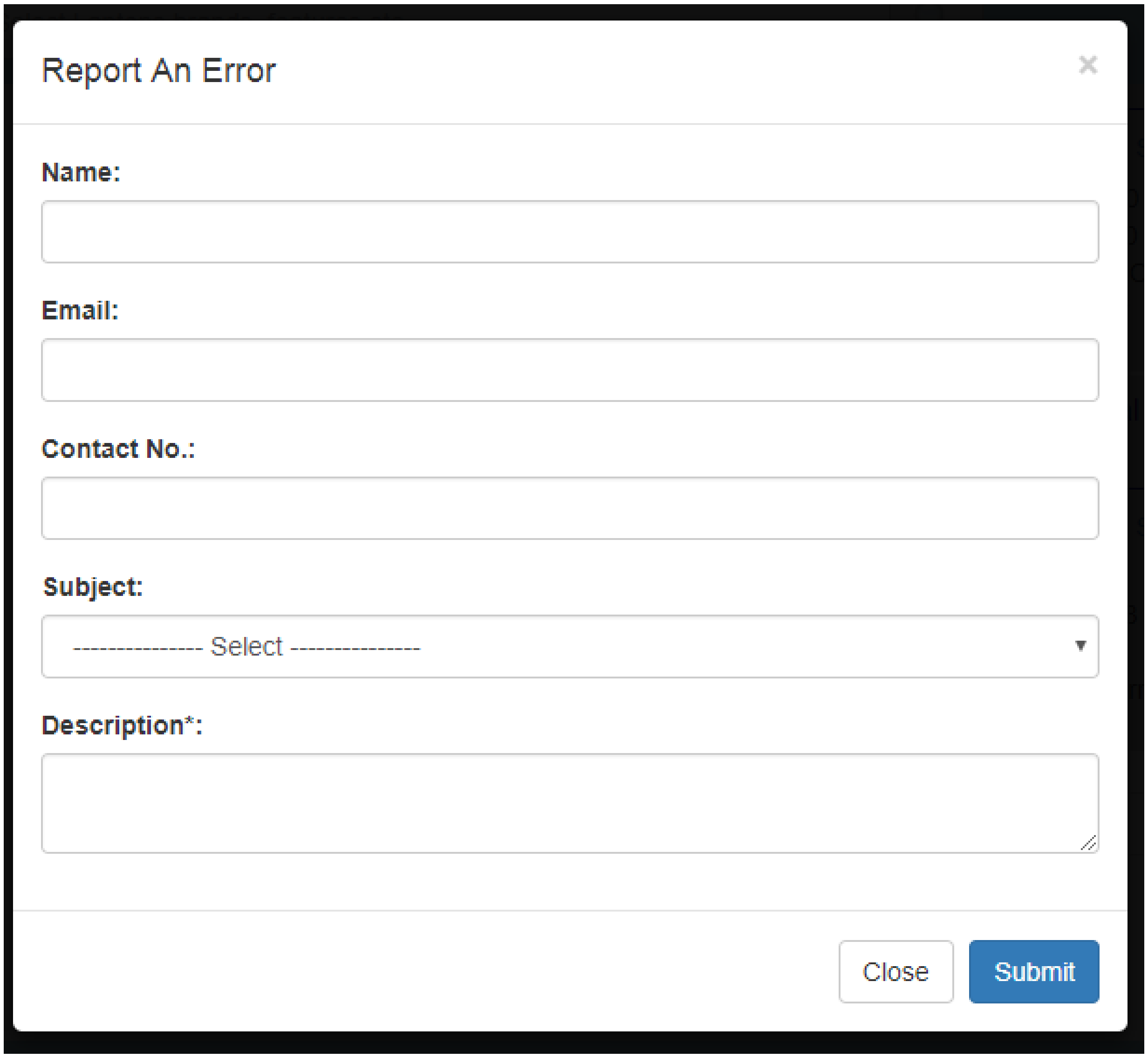
Top

© 2017-18 - prioritylappi.com, All rights reserved | Privacy Policy | Terms of service

Feedack Report

Fig (23) Home page Footer

The process of reporting errors is sometimes referred to as disclosure of errors, causing confusion. For reporting any faults in the website or any product related issues. The user can report it easily to the subjects like incorrect price, incorrect/missing product details, coupon not working, found the lowest price somewhere else, or it could be just a feedback.


Report An Error

Name:

Email:

Contact No.:

Subject:

--------------- Select ---------------

Description*:

Fig (24) Report an error

Process in which the effect or output of an action is 'returned' (fed-back) to modify the next action. Feedback is essential to the working and survival of all regulatory mechanisms found throughout living and non-living nature, and in man-made systems such as education system and economy. User is going to give the feedback be it related to website interface, its working, the products qualities, the services given by us.

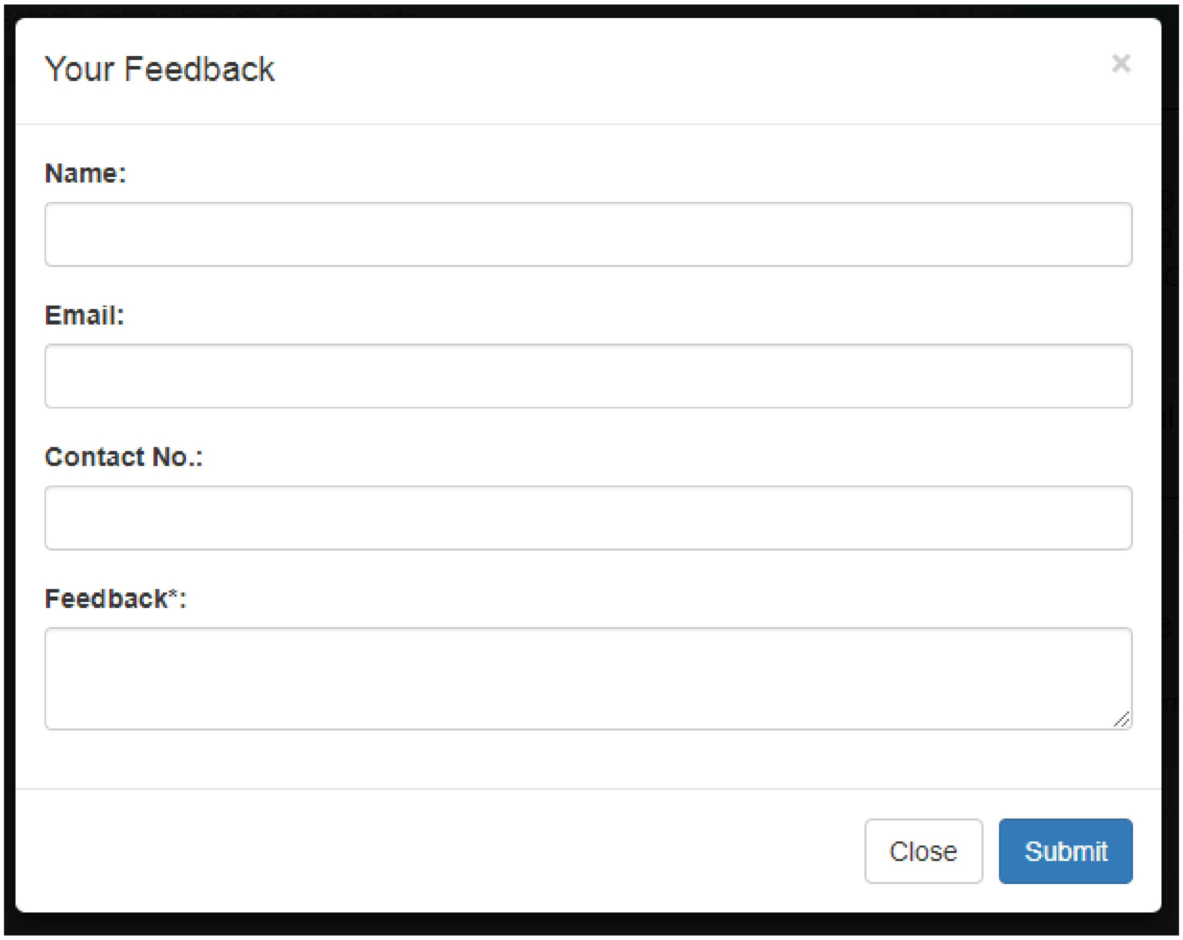
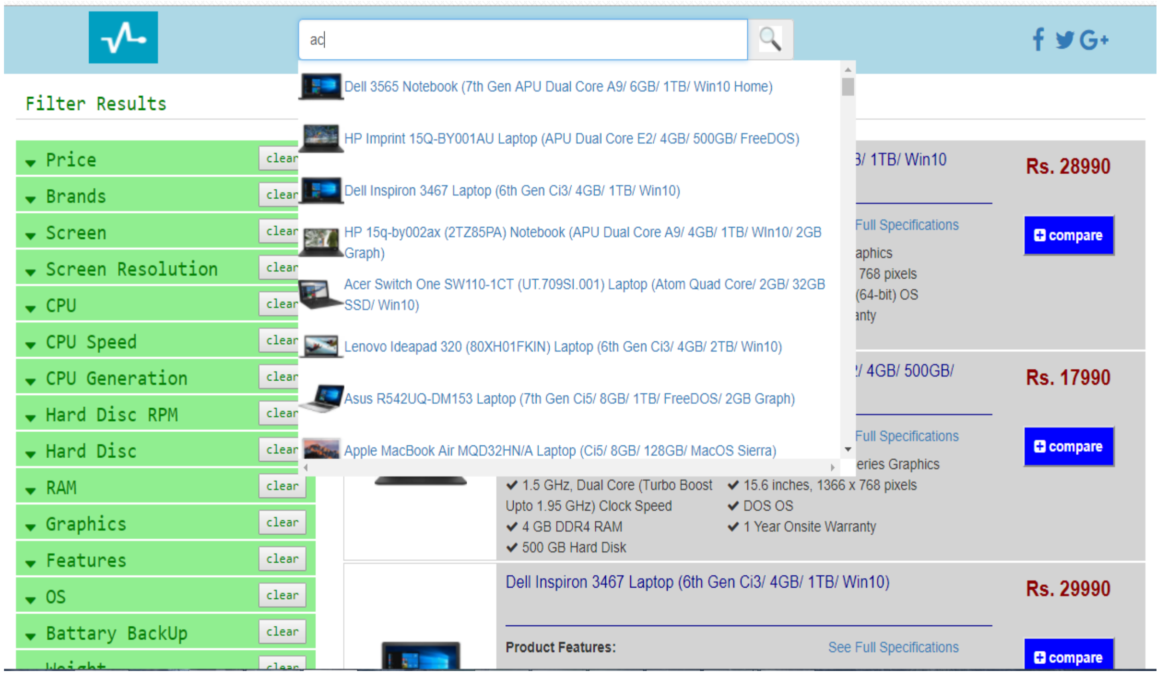


Fig (25) Feedback

Now for searching the products, we will search it in the search bar. The searching is done according to the choice of the user. Once the product is being found will be used by the user. Once the product is being found will be used by the user. Searching saves the time of the user, instead of searching name by name the searching could be done alphabetically which makes the work more reliable and flexible which makes is more attractive to use.



ac

Dell 3565 Notebook (7th Gen APU Dual Core A9/ 6GB/ 1TB/ Win10 Home)

HP Imprint 15Q-BY001AU Laptop (APU Dual Core E2/ 4GB/ 500GB/ FreeDOS)

Dell Inspiron 3467 Laptop (6th Gen Ci3/ 4GB/ 1TB/ Win10)

HP 15q-by002ax (2TZ85PA) Notebook (APU Dual Core A9/ 4GB/ 1TB/ WIn10/ 2GB Graph)

Acer Switch One SW110-1CT (UT.709SI.001) Laptop (Atom Quad Core/ 2GB/ 32GB SSD/ Win10)

Lenovo Ideapad 320 (80XH01FKIN) Laptop (6th Gen Ci3/ 4GB/ 2TB/ Win10)

Asus R542UQ-DM153 Laptop (7th Gen Ci5/ 8GB/ 1TB/ FreeDOS/ 2GB Graph)

Apple MacBook Air MQD32HN/A Laptop (Ci5/ 8GB/ 128GB/ MacOS Sierra)

Filter Results

Price	clear
Brands	clear
Screen	clear
Screen Resolution	clear
CPU	clear
CPU Speed	clear
CPU Generation	clear
Hard Disc RPM	clear
Hard Disc	clear
RAM	clear
Graphics	clear
Features	clear
OS	clear
Battery BackUp	clear

3/ 1TB/ Win10 Rs. 28990

Full Specifications compare

aphics
768 pixels
(64-bit) OS
anty

?/ 4GB/ 500GB/ Rs. 17990

Full Specifications compare

eries Graphics

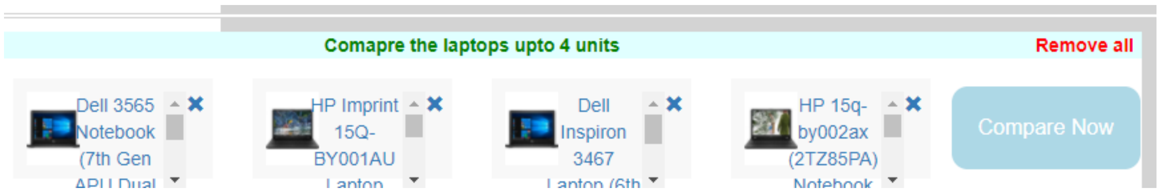
✓ 1.5 GHz, Dual Core (Turbo Boost Upto 1.95 GHz) Clock Speed
✓ 4 GB DDR4 RAM
✓ 500 GB Hard Disk
✓ 15.6 inches, 1366 x 768 pixels
✓ DOS OS
✓ 1 Year Onsite Warranty

Dell Inspiron 3467 Laptop (6th Gen Ci3/ 4GB/ 1TB/ Win10) Rs. 29990

Product Features: See Full Specifications compare

Fig (26) Search bar

Now for comparing the product we have to add the products and maximum four products we can compare at the same time. The comparing will be done according to the comparison algorithms which is a part of backend portion.



Comapre the laptops upto 4 units

Remove all

Dell 3565 Notebook (7th Gen APU Dual

HP Imprint 15Q-BY001AU Laptop

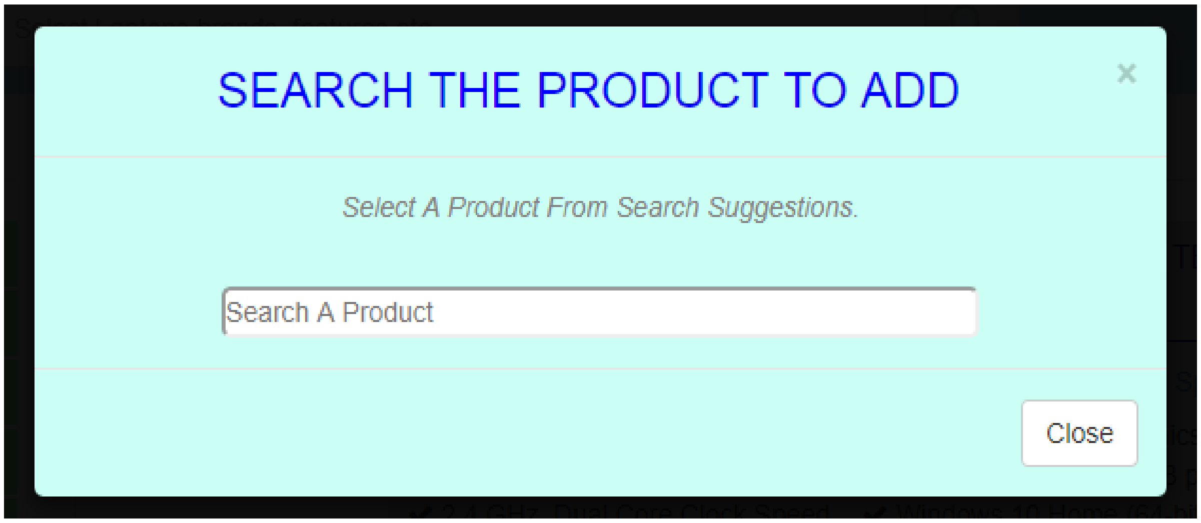
Dell Inspiron 3467 Laptop (6th

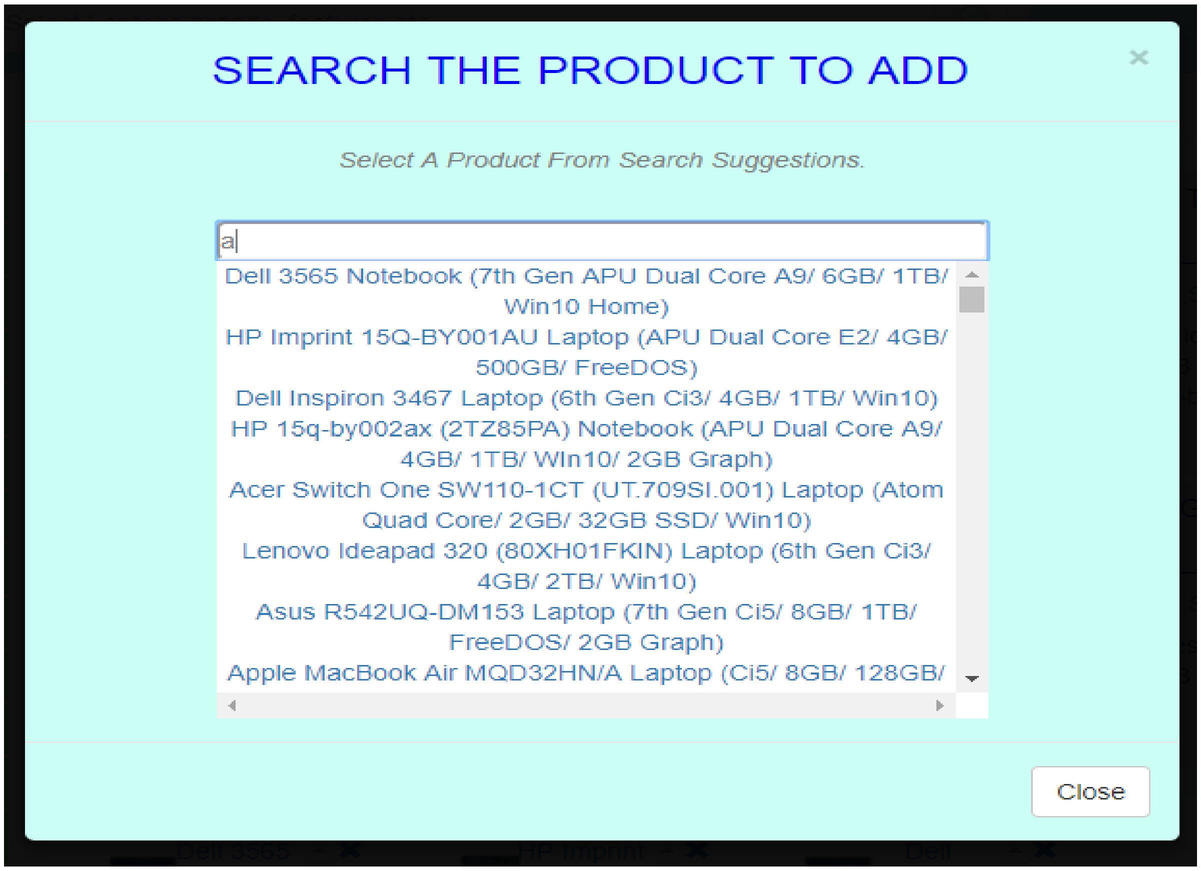
HP 15q-by002ax (2TZ85PA) Notebook

Compare Now

Fig (27) Add products for comparing

Now if the product is not shown on the home page we will search it from our database. The searching concept is same as in the search bar.

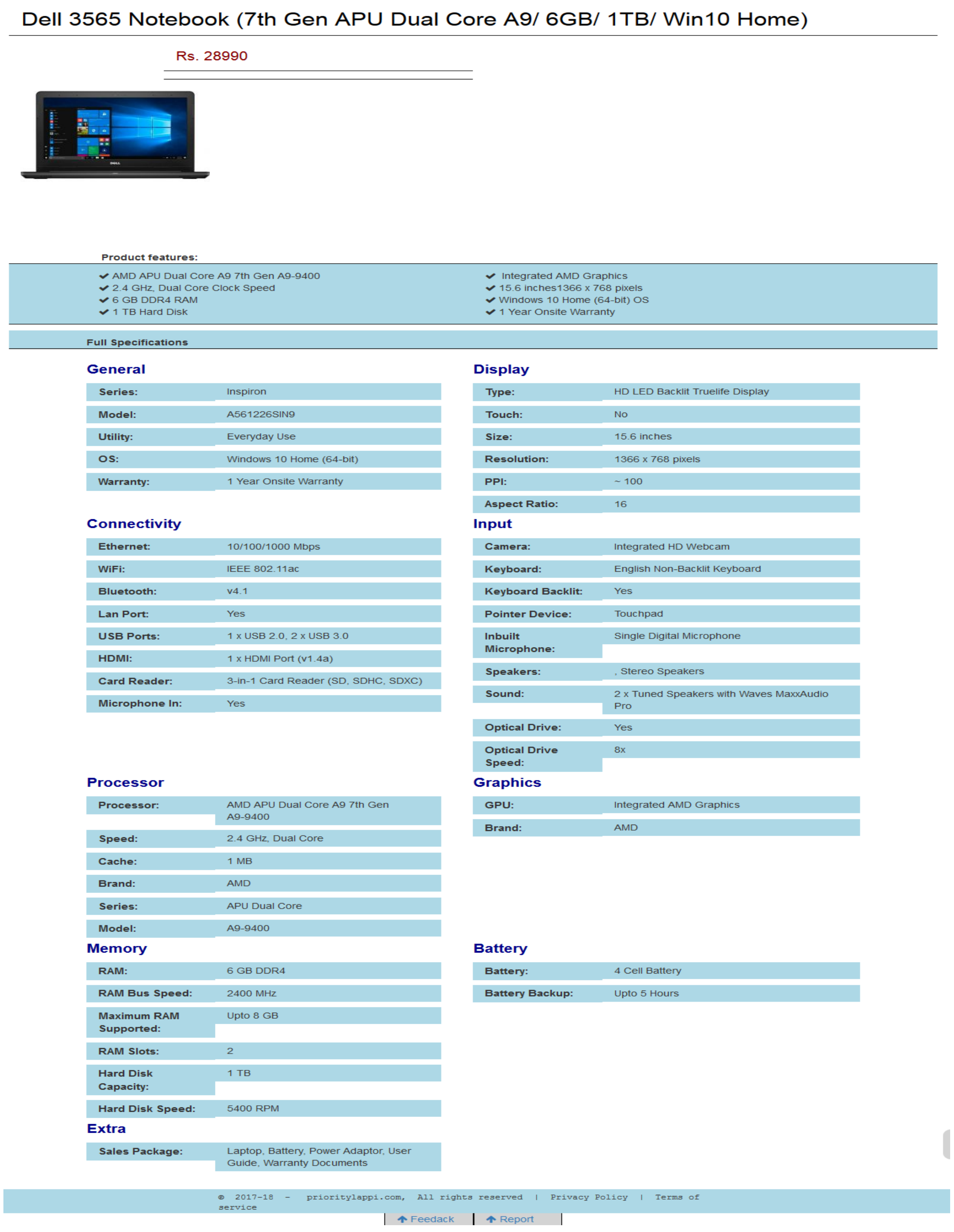

SEARCH THE PRODUCT TO ADD
×
Select A Product From Search Suggestions.
Search A Product
Close


SEARCH THE PRODUCT TO ADD
×
Select A Product From Search Suggestions.
a
Close

Dell 3565 Notebook (7th Gen APU Dual Core A9/ 6GB/ 1TB/ Win10 Home)
HP Imprint 15Q-BY001AU Laptop (APU Dual Core E2/ 4GB/ 500GB/ FreeDOS)
Dell Inspiron 3467 Laptop (6th Gen Ci3/ 4GB/ 1TB/ Win10)
HP 15q-by002ax (2TZ85PA) Notebook (APU Dual Core A9/ 4GB/ 1TB/ WIn10/ 2GB Graph)
Acer Switch One SW110-1CT (UT.709SI.001) Laptop (Atom Quad Core/ 2GB/ 32GB SSD/ Win10)
Lenovo Ideapad 320 (80XH01FKIN) Laptop (6th Gen Ci3/ 4GB/ 2TB/ Win10)
Asus R542UQ-DM153 Laptop (7th Gen Ci5/ 8GB/ 1TB/ FreeDOS/ 2GB Graph)
Apple MacBook Air MQD32HN/A Laptop (Ci5/ 8GB/ 128GB/

Fig (28) Search the product to add

Now if you want to see the details and specification of laptops you can click on that laptop. All the software and hardware specification will be displayed.



Dell 3565 Notebook (7th Gen APU Dual Core A9/ 6GB/ 1TB/ Win10 Home)

Rs. 28990

Product features:

- ✓ AMD APU Dual Core A9 7th Gen A9-9400
- ✓ 2.4 GHz, Dual Core Clock Speed
- ✓ 6 GB DDR4 RAM
- ✓ 1 TB Hard Disk
- ✓ Integrated AMD Graphics
- ✓ 15.6 inches1366 x 768 pixels
- ✓ Windows 10 Home (64-bit) OS
- ✓ 1 Year Onsite Warranty

Full Specifications

General

Series:	Inspiron
Model:	A561226SIN9
Utility:	Everyday Use
OS:	Windows 10 Home (64-bit)
Warranty:	1 Year Onsite Warranty

Display

Type:	HD LED Backlit Truelife Display
Touch:	No
Size:	15.6 inches
Resolution:	1366 x 768 pixels
PPI:	~ 100
Aspect Ratio:	16

Connectivity

Ethernet:	10/100/1000 Mbps
WiFi:	IEEE 802.11ac
Bluetooth:	v4.1
Lan Port:	Yes
USB Ports:	1 x USB 2.0, 2 x USB 3.0
HDMI:	1 x HDMI Port (v1.4a)
Card Reader:	3-in-1 Card Reader (SD, SDHC, SDXC)
Microphone In:	Yes

Input

Camera:	Integrated HD Webcam
Keyboard:	English Non-Backlit Keyboard
Keyboard Backlit:	Yes
Pointer Device:	Touchpad
Inbuilt Microphone:	Single Digital Microphone
Speakers:	, Stereo Speakers
Sound:	2 x Tuned Speakers with Waves MaxxAudio Pro
Optical Drive:	Yes
Optical Drive Speed:	8x

Processor

Processor:	AMD APU Dual Core A9 7th Gen A9-9400
Speed:	2.4 GHz, Dual Core
Cache:	1 MB
Brand:	AMD
Series:	APU Dual Core
Model:	A9-9400

Graphics

GPU:	Integrated AMD Graphics
Brand:	AMD

Memory

RAM:	6 GB DDR4
RAM Bus Speed:	2400 MHz
Maximum RAM Supported:	Upto 8 GB
RAM Slots:	2
Hard Disk Capacity:	1 TB
Hard Disk Speed:	5400 RPM

Battery

Battery:	4 Cell Battery
Battery Backup:	Upto 5 Hours

Extra

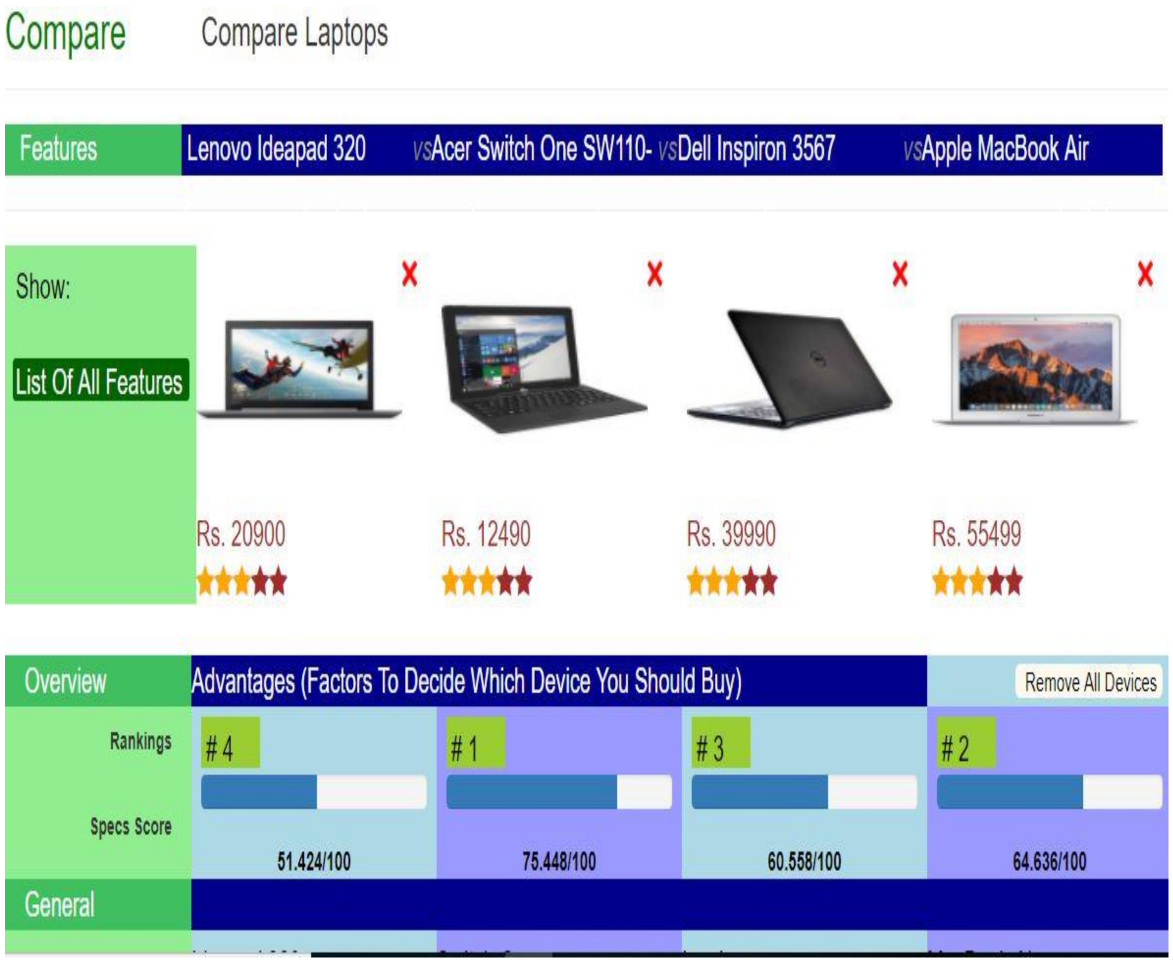
Sales Package:	Laptop, Battery, Power Adaptor, User Guide, Warranty Documents

© 2017-18 - prioritylappi.com, All rights reserved | Privacy Policy | Terms of service

Feedack | Report

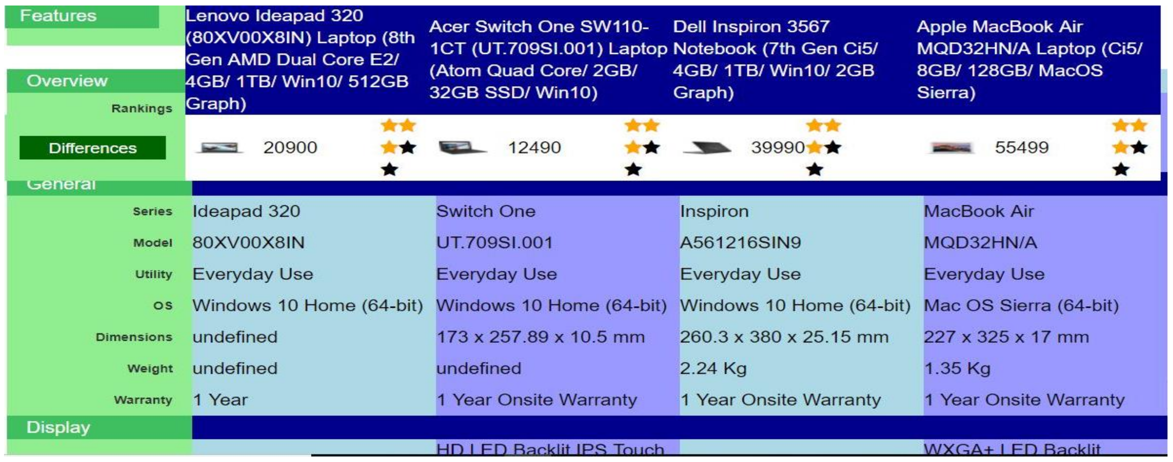
Fig (29) Specification of laptop

Before boosting the score, the following figure shows the specification score

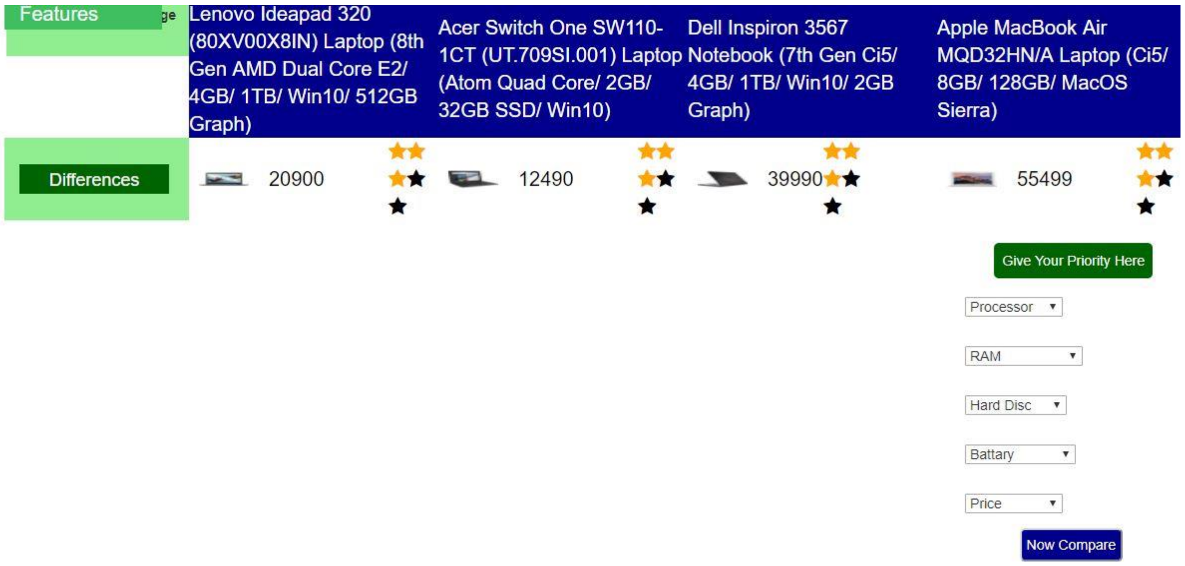


Compare Compare Laptops

Features	Lenovo Ideapad 320	vs Acer Switch One SW110-	vs Dell Inspiron 3567	vs Apple MacBook Air
Show: List Of All Features	Rs. 20900	Rs. 12490	Rs. 39990	Rs. 55499
Overview	**Advantages (Factors To Decide Which Device You Should Buy)**			Remove All Devices
Rankings	# 4	# 1	# 3	# 2
Specs Score	51.424/100	75.448/100	60.558/100	64.636/100
General				



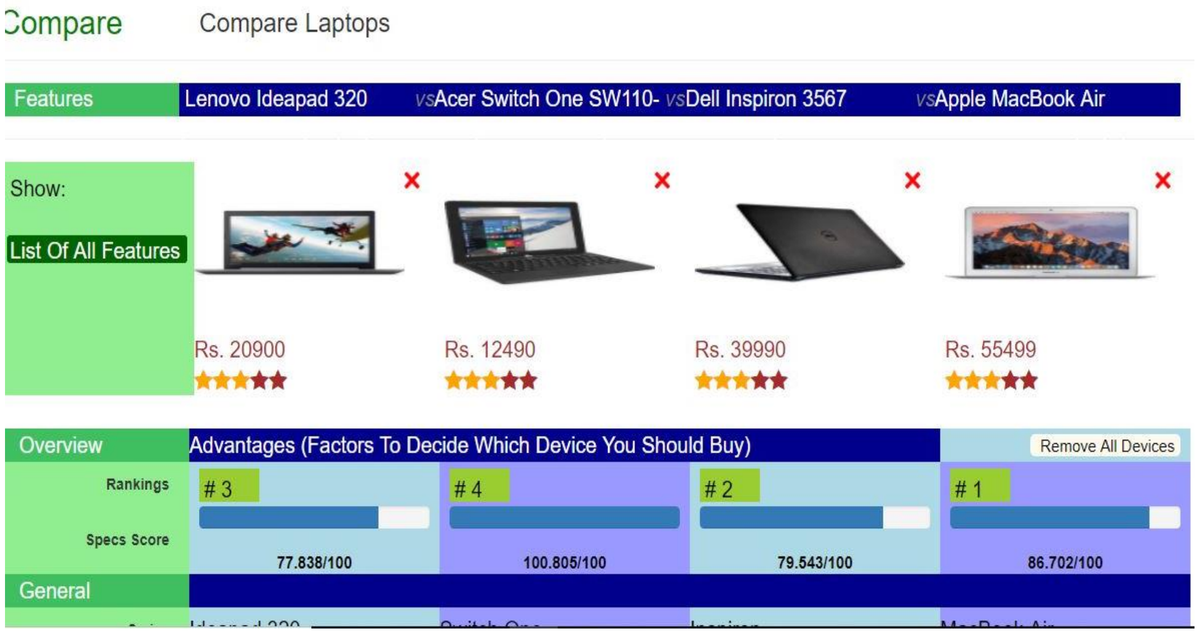
Features	Lenovo Ideapad 320 (80XV00X8IN) Laptop (8th Gen AMD Dual Core E2/ 4GB/ 1TB/ Win10/ 512GB Graph)	Acer Switch One SW110-1CT (UT.709SI.001) Laptop (Atom Quad Core/ 2GB/ 32GB SSD/ Win10)	Dell Inspiron 3567 Notebook (7th Gen Ci5/ 4GB/ 1TB/ Win10/ 2GB Graph)	Apple MacBook Air MQD32HN/A Laptop (Ci5/ 8GB/ 128GB/ MacOS Sierra)
Overview Rankings Differences	20900	12490	39990	55499
General				
Series	Ideapad 320	Switch One	Inspiron	MacBook Air
Model	80XV00X8IN	UT.709SI.001	A561216SIN9	MQD32HN/A
Utility	Everyday Use	Everyday Use	Everyday Use	Everyday Use
OS	Windows 10 Home (64-bit)	Windows 10 Home (64-bit)	Windows 10 Home (64-bit)	Mac OS Sierra (64-bit)
Dimensions	undefined	173 x 257.89 x 10.5 mm	260.3 x 380 x 25.15 mm	227 x 325 x 17 mm
Weight	undefined	undefined	2.24 Kg	1.35 Kg
Warranty	1 Year	1 Year Onsite Warranty	1 Year Onsite Warranty	1 Year Onsite Warranty
Display				
		HD LED Backlit IPS Touch		WXGA+ LED Backlit



Features	Lenovo Ideapad 320 (80XV00X8IN) Laptop (8th Gen AMD Dual Core E2/ 4GB/ 1TB/ Win10/ 512GB Graph)	Acer Switch One SW110-1CT (UT.709SI.001) Laptop (Atom Quad Core/ 2GB/ 32GB SSD/ Win10)	Dell Inspiron 3567 Notebook (7th Gen Ci5/ 4GB/ 1TB/ Win10/ 2GB Graph)	Apple MacBook Air MQD32HN/A Laptop (Ci5/ 8GB/ 128GB/ MacOS Sierra)
Differences	20900	12490	39990	55499

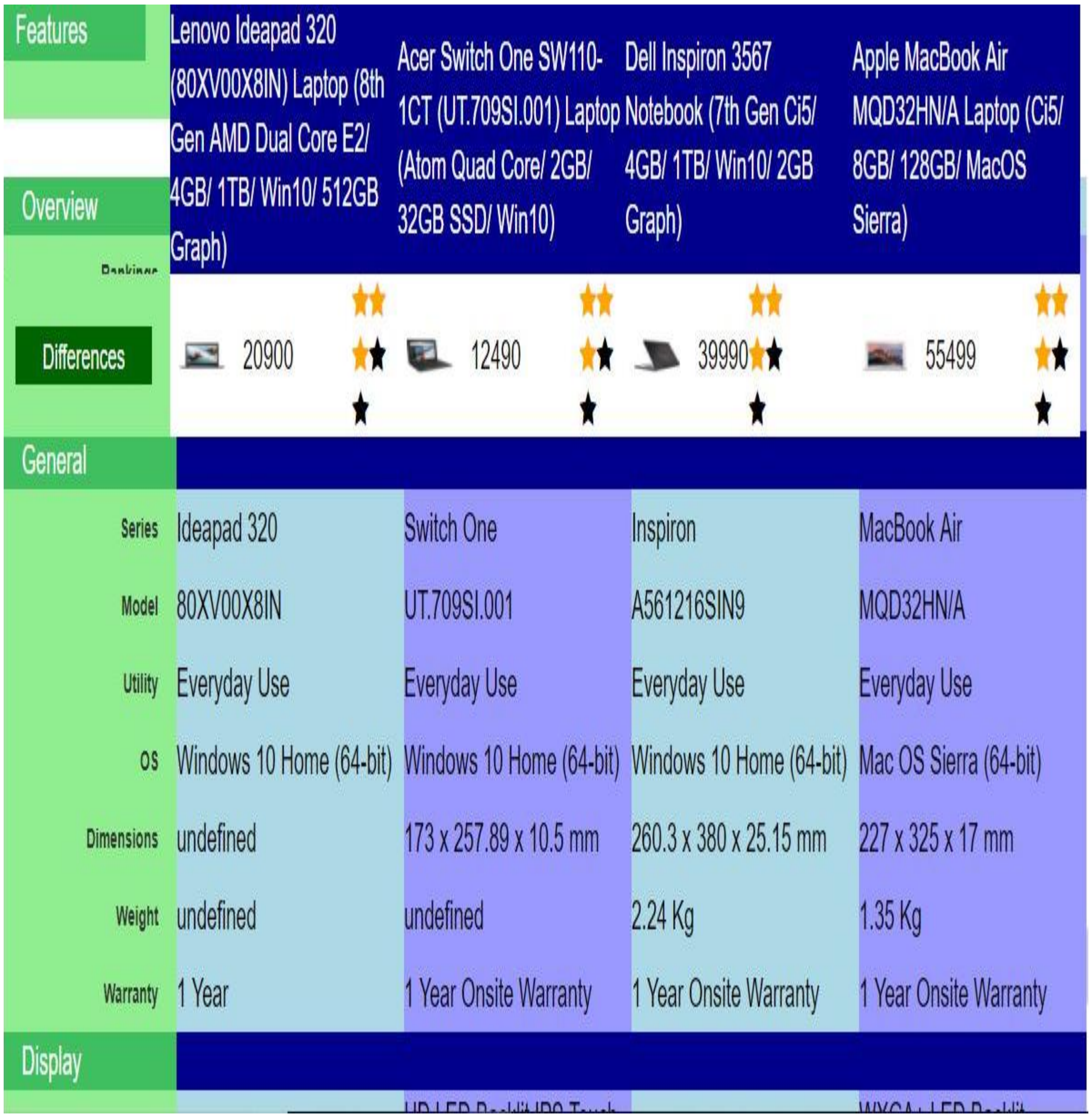
Fig (30) Before boost

Now, we will give our priority according to which our required products will be displayed.



Features	Lenovo Ideapad 320	vs Acer Switch One SW110-	vs Dell Inspiron 3567	vs Apple MacBook Air
Show: List Of All Features	Rs. 20900	Rs. 12490	Rs. 39990	Rs. 55499

Overview	Advantages (Factors To Decide Which Device You Should Buy)			Remove All Devices
Rankings	# 3	# 4	# 2	# 1
Specs Score	77.838/100	100.805/100	79.543/100	86.702/100
General				



Features	Lenovo Ideapad 320 (80XV00X8IN) Laptop (8th Gen AMD Dual Core E2/ 4GB/ 1TB/ Win10/ 512GB Graph)	Acer Switch One SW110-1CT (UT.709SI.001) Laptop (Atom Quad Core/ 2GB/ 32GB SSD/ Win10)	Dell Inspiron 3567 Notebook (7th Gen Ci5/ 4GB/ 1TB/ Win10/ 2GB Graph)	Apple MacBook Air MQD32HN/A Laptop (Ci5/ 8GB/ 128GB/ MacOS Sierra)
Overview				
Rankings				
Differences	20900 ★★★★★	12490 ★★★★★	39990 ★★★★★	55499 ★★★★★
General				
Series	Ideapad 320	Switch One	Inspiron	MacBook Air
Model	80XV00X8IN	UT.709SI.001	A561216SIN9	MQD32HN/A
Utility	Everyday Use	Everyday Use	Everyday Use	Everyday Use
OS	Windows 10 Home (64-bit)	Windows 10 Home (64-bit)	Windows 10 Home (64-bit)	Mac OS Sierra (64-bit)
Dimensions	undefined	173 x 257.89 x 10.5 mm	260.3 x 380 x 25.15 mm	227 x 325 x 17 mm
Weight	undefined	undefined	2.24 Kg	1.35 Kg
Warranty	1 Year	1 Year Onsite Warranty	1 Year Onsite Warranty	1 Year Onsite Warranty
Display				
		HD LED Backlit IPS Touch		WXGA+ LED Backlit

Fig (31) After Boost

Now, after the boosting the score will be as per in the given above diagram.

Chapter 6

PLANNING OF WORK

Module1: Requirements Analysis, Feasibility Study (For deciding the parameters)
Module2: Synopsis presentation
Module3: Data Collection (Scraping)
Module4: Database Design and scraped data insert into database
Module5: Front End Design
Module6: Back End Design (Algorithm design for filtering, sorting and comparison)
Module7: Implementation and validation
Module8: Testing
Module9: Project demonstration, going live and final report submission

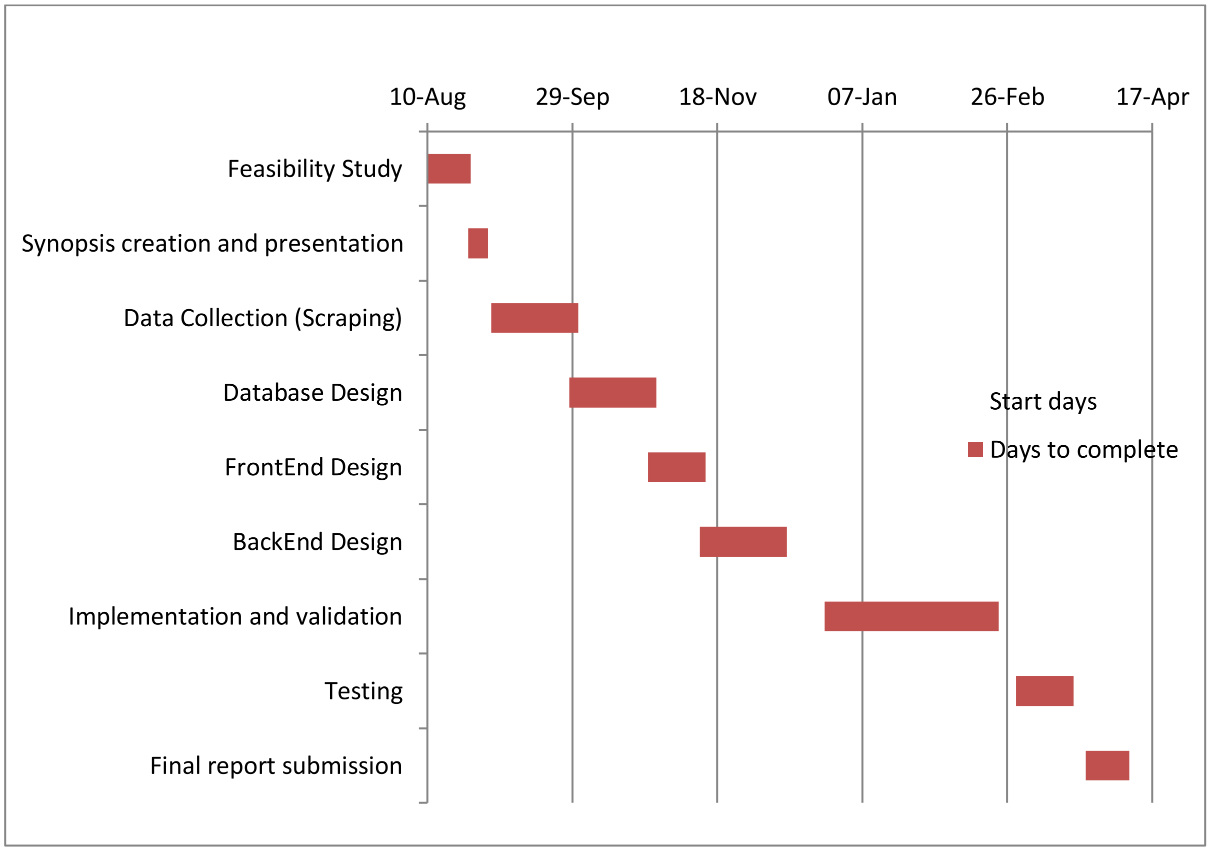


Fig (32) Gantt Chart

Chapter 7
CONCLUSION

PBCS was mainly focused on the prioritization and comparison of products. Prioritization or sentiments has been researching area for many years. With the help of scraping, we are able to store the data and compare them later on. Priority of the user is the main concern, because whenever a user shops he/she wants the best product in market. And for that he need to compare the products according to their priorities. A specification score helps the user for his comparison. The algorithm is written for comparison as well as for prioritization. Priority Based Comparison System provides a platform where one can view prices, specifications & prioritize and compare product from different E-commerce websites on a single platform. PBCS has been developing by using MEAN technology which has emerged in the market at a very fast rate. Most of the industry is moving towards MEAN stack technology because it's several advantages over other technologies. With the mixture of MEAN and text classification, PBCS covers the major need for a better comparison system for the products. Earlier dispersion of data at different platform led to customers' problems like more time and more data. With PBCS customer gets one common platform for better comparison study.

REFERENCES

[1] htttps://www.smartprix.com

[2] https://www.smartprix.com/laptops

[3] https://www.smartprix.com/laptops/compare.php

[4] https://www.smartprix.com/laptops/dell-inspiron-3467-laptop-6th-gen-ci3-4gb-1tb-win10

[5] https://stackoverflow.com/questions/49600975/how-can-i-filter-my-collection-according-to-the-data-in-array

[6] https://stackoverflow.com/questions/50031060/can-i-use-jquery-for-performing-operation-the-html-elements-made-inside-an-ajax

[7] https://stackoverflow.com/questions/49847352/how-can-i-query-on-the-result-i-get-from-another-query-in-mongodb

[8] https://stackoverflow.com/questions/49729162/how-can-i-return-the-result-of-my-node-controller-to-my-application-through-ajax

[9] https://stackoverflow.com/questions/49721427/how-can-i-select-the-element-of-html-by-combing-both-class-and-id-attribute

[10] https://stackoverflow.com/questions/49931108/can-i-render-the-response-of-the-success-block-of-ajax-to-the-ejs-file

[11] https://www.w3schools.com/php/php_ajax_intro.asp

[12] https://docs.mongodb.com/manual/reference/operator/query/or/

[13] https://docs.mongodb.com/manual/reference/operator/query/or/

[14] https://api.mongodb.com/python/current/

[15] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[16] https://readthedocs.org/projects/beautiful-soup-4/

[17] https://www.cpubenchmark.net/

[18] https://www.videocardbenchmark.net/

[19] https://www.harddrivebenchmark.net/

[20] https://www.memorybenchmark.net/