

Product Preeminent Price

A Senior Design Project Report

Prepared By

Md.Muttashim Mishel Mawn (ID: 1520800042)

Fatema Tasmim Tisha (ID: 1511996042)

Masud-Ur-Rahman (ID: 1411676642)

Subrena Islam (ID: 1620283042)

Supervised By

Mohammad Ashrafuzzaman Khan

Associate Professor

Electrical and Computer Engineering



Department of Electrical & Computer Engineering

North South University

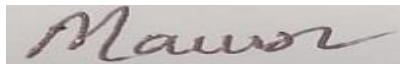
Dhaka 1229

Spring 2020

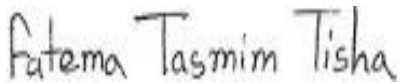
DECLARATION

We, hereby, declare that the work presented in this project report is the outcome of the design and development work performed by us under the supervision of Mohammad Ashrafuzzaman Khan, Associate Professor, Electrical & Computer Engineering, North South University as a course work of CSE/EEE 499A and CSE/EEE 499B (Capstone Senior Design Project). We also declare that no part of this report has been taken from other works without reference.

Signature of Students



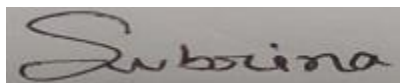
Md.Muttashim Mishel Mawn



Fatema Tasmim Tisha



Masud-Ur-Rahman



Subrena Islam

APPROVAL

This project report titled 'Product Preeminent Price' submitted by (i) Md. Muttashim Mishel Mawn ID: 1520800042 (ii) Fatema Tasmim Tisha ID: 1511996042 (iii) Masud-Ur-Rahman ID: 1411676642 & (iv) Subrena Islam ID: 1620283042 to the Department of Electrical and Computer Engineering, North South University, has been accepted as Senior Design Project Term Final Report.

Signatures

Mohammad Ashrafuzzaman Khan

Associate Professor, ECE

Dr. Mohammad Rezaul Bari

Associate Professor and Chairman, ECE

Abstract

We have proposed a client server based novel design for a product preeminent price from our country online based market. Because of now a days a lot of online sale service provides various product at our door but it's so difficult to find our desired product with minimum price. We know that people want their need but no one interest to spend much money or no one interest to spend extra money for a product everyone wants the cheapest one. And our software is the place where we pre scrap product data and search and find out those products that are cheap. After that we show up those preeminent data on a visual web site that people can know who provide the cheapest price and many more. This project use bs4 from python library for data scrap from various website to collect data. This paper also show the limitation of our application find when data are missing from its mother website. This paper also shows that our application can run in different device. There are lots of technical issue and technical challenges described in the report which we find when we try to do this research. However, the problems and solution are discussed smoothly so that future researchers are not faced any problem in that case.

Table of Contents

Introduction	8
Background	8
Problem Statement.....	9
Description of the problem being solved	9
Contribution.....	10
 Feasibility Study	 10
Problem Solution.....	10
Dataset Representation	10
 Proposed Solution	 11
Platform	11
Software.....	11
Packages.....	11
Proposed Solution Flowchart.....	12
Dataset Selection	13
Pre-Processing.....	13
Punctuation Removal.....	13
 Impacts	 14
Technical Analysis	14
Financial Analysis	14
Environmental Analysis.....	14
Societal Analysis.....	14
 Result Analysis.....	 15
Screenshots.....	15
Methodology.....	18
Discussion.....	19

Design Impacts & Cost.....	20
Time Cost	20
Improvement	21
Future Work	21
 Conclusion	 22
 References	 23
 Acknowledgement	 24
 Appendix	 25
Code	25-90

List of Figures

1. Proposed Solution Block Diagram
2. Application UI
3. Searching outcome (Solr)
4. Sample Dataset

Introduction

Background

Internet now consists of lots of data. People use this data for their daily works. Online data is a collection of facts that has been translated into a form that computer can process. The amount of online data is increasing day by day. Online marketplace is one of the biggest data inventories that contain a lot of similar data. People all over the world moves faster respective with time. So we can say that time is the ultimate world currency. So in this era people are mostly dependable on online marketplace. Buy sell or ride whatever you want you can find a lot of place in online. And one of the biggest parts is basic product purchase from online. Every one needs their demand at minimum price but best quality. One product can found in many shops. And the hardest part is to find out who provide the minimum price.

Day by day demand increase so that more people are interested to join online business. Everyone wants to convince you that they provide the best product with best price. So let's talk about electronics product. So Gigabyte making motherboard and they distribute their product all over the world. Every motherboard has different kind of feature and price tag. If you think about its quality then we should talk about another company who distribute motherboard. In this case we are thinking about single product from different place and different price tag. Both of the motherboard is same but why their pricing is different?

If we think about global market like Amazon, Alibaba then the price must be different because of government tax and shipping cost. So in this case we only consider our own marketplace. A lot of seller selling electronic device in Bangladesh under different company banner. Some of them provide almost same price but many more seller increase product price. So we try to scrape those data. Then build an application that shows who provide minimum price among them.

Problem Statement

In Bangladesh a lot of seller sells electronic products. Some of them are fully dependable of online web based applications and some of them use social media, blog and some of them are use both physical and online based application. And almost everyone sells same product but some of them has different pricing.

In this case we only scrap web based application data and organize them to in a same pattern so that we can use search operation easily. But one of the biggest obstacles is some of the seller use Bangla font in products title and some of then use code to their own way to manage their shop. So it is so hard to find out similar product among the all data we collected.

If we done it then the most awaiting question come in the front of us are what the accuracy is and how much data we are missing in searching operation. So that we categorized data as the online shops are provide. It will help us to maximum accuracy and less number of data missing in searching operation.

Description of the problem being solved

There is a lot of theoretical way to scrap data from web, but we use one of the strong libraries of python called BeautifulSoup 4. And the rest of them are Request, lxml, selenium and scrapy. We use bs4 because it's faster. After scraping data we use solr for data search from csv file, and maybe this one is not completely accurate to search a data but it's faster than the normal key to search algorithm. Every product title has 7-8 unique keywords and searching with multiple keywords is faster with solr.

Contributions

The proposed methodology may be among some first proposal. As there are not many researches done in this field in product title scraping and find the minimum price. Basically compare a single product with multiple providers. There are number of research done in comparison product but not in this particular way. Our objectives are to develop a platform where shows the providers or shop name who provide cheapest price all of them.

Feasibility Study

We discussed about possible solutions in this part.

Possible Solution

There may be many procedures for categorization and scraping data. There are some parts fixed for web scarping. In this section, we try to propose alternative solutions for each section.

Dataset Representation

Dataset selection is the most important for product title text. Data representation is the most unique and important part of this project. Firstly, data can have gathered in .txt format in each separate format. Reading directly from .txt and store them for separate class may be difficult. So, we can transfer those txt files into CSV format where title, price and category will be remained in the excel file.

Proposed Solution

Platform

Software

- *python 3.7.6

- *Laravel

- *PHP

- *Microsoft Excel

Packages

- *Beautiful Soup 4

- *Pandas

Proposed Solution Flowchart

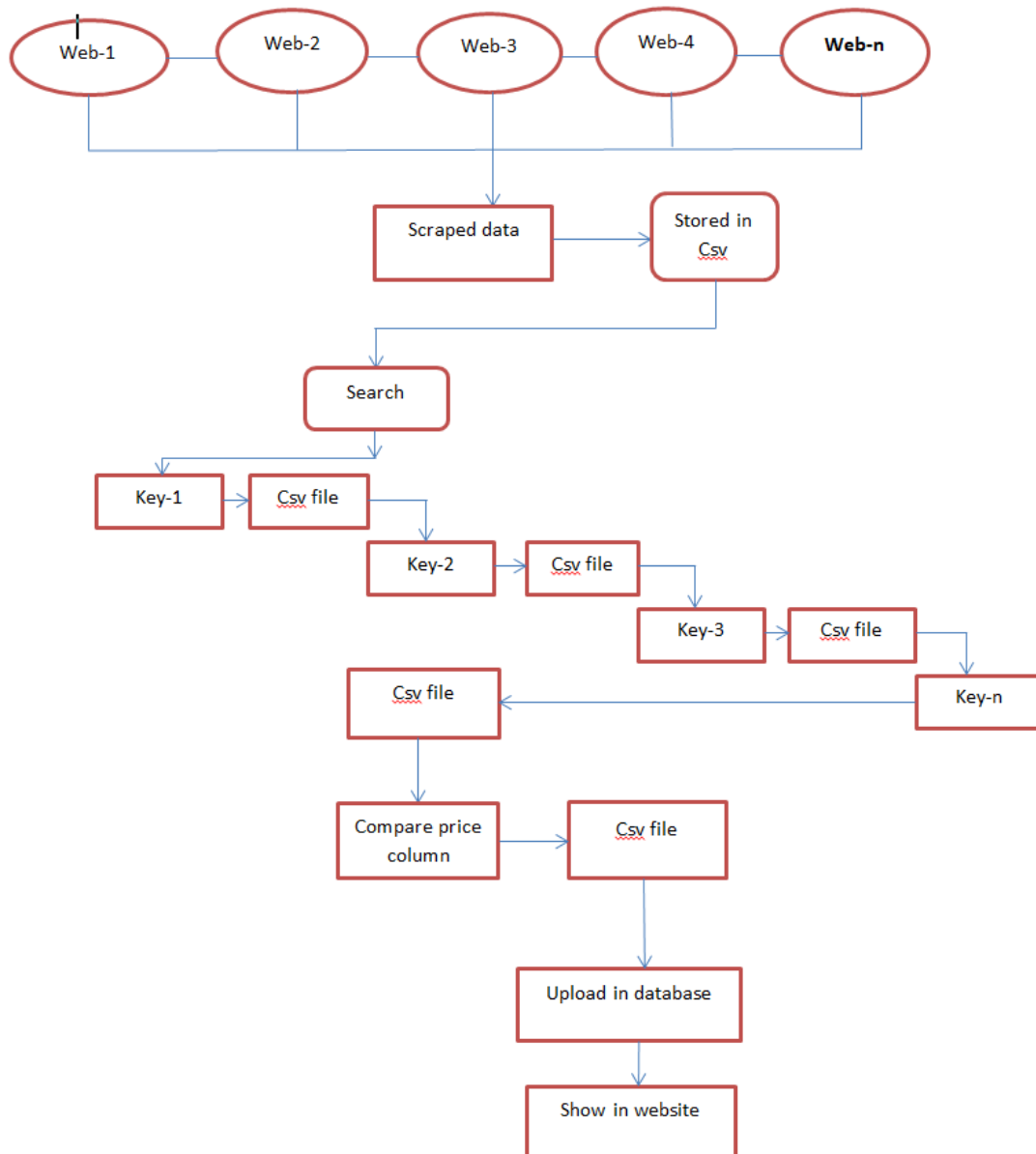


Fig 1: Proposed Solution Block Diagram

Data Set Selection

Dataset selection plays a very important role in supervised machine learning approaches to categorizing. The quality and the quantity of data shape the goal to predict a text to the right output. Web based shops is our main goal to scrap data but not all of the shops we choose digital device like motherboard, graphics cards, monitor and many more.

Pre-Processing

In this step, our goal is to remove noises from the data. As texts are a much unstructured way to represent information and consist noise that can make the classifier very difficult to do its work. There are some basic ways to preprocess text to extract only relevant pieces of information. We divided the preprocessing step into three steps. But in this project we are working in English language only so that for this application we need one pre-processing step that is

✓ Punctuation removal

Punctuation Removal

Since the punctuations play a very little role in order to contribute being a feature and also can be considered as noise we have removed all the punctuations used in the title of a product. We have also removed numerals and special characters for the same reason. Some of the examples of the punctuations, numerals, and special characters that we removed are given below

—, \, \, , , †, ‡, §, , s, S, , ` , ? , , % , , , , : , = , i , x , ! , \$, ¢ , ® , ¯ , ¬ , ´ , ¼ , å , è , ò , ö , ÷ , ø , ý , ý

Impacts

Impact analysis of this project consists of technical, financial and societal analysis.

Technical Analysis

Data scrap is one of the important parts of internet or web. Now a day everything is analyzed on web using text mining techniques. It's quite difficult for a person to check every product title and matched with other shops title and then store one of them. That's why data scraping and organize this data is so important for us.

Financial Analysis

They can use this data analysis to categorize reviews of a product and find out valuable information's from both seller and buyer. It's also beneficial for business farms or big companies to predict the market demand stock and customer taste.

Environmental Analysis

Data scraping from web based application and it's only used by people and developer. So we do not think there is any kind of environmental impact of this application.

Societal Analysis

There are impacts on societal analysis. By scraping data from online based market system will reduce fake seller, fake product and some of the high price provider sometimes make stock limit and made the market out of range it will be reduced in some ways.

Result Analysis

Screenshots

GIGABYTE AORUS GEFORCE RTX 2080 TI XTREME TIGB GRAPHICS CARD

From Techland

*Hover to zoom

Primeenent Price

Price: 122000 tk

shop link

[Shop Now \(Techland\)](#)

[Add to Wishlist](#)



Shop Name	Startech	Ryans	Computer-village
Price	122500 tk	122500 tk	122500 tk
Shop Link	Shop Now	Shop Now	Shop Now

Fig 2: Application UI

- Dashboard
- Logging
- Core Admin
- Java Properties
- Thread Dump
- directors
 - Overview
 - Analysis
 - Dataimport
 - Documents
 - Files
 - Ping
 - Plugins / Stats
 - Query
 - Replication
 - Schema
 - Segments Info

Request-Handler (qt)

/select

common

q

SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail

fq

sort

start, rows

0 10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

☐ indent off
☐ debugQuery

☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck

Execute Query

[http://localhost:8984/solr/directors/select?q=name:CORSAIR XMS 2GB \(2 x 1GB\) 184-Pin DDR SDRAM Unbuffered DDR 400 \(PC 3200\) Dual Channel Kit System Memory - Retail](http://localhost:8984/solr/directors/select?q=name:CORSAIR XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail)

```

{
  "responseHeader":{
    "status":0,
    "QTime":3,
    "params":{
      "q":"name:CORSAIR XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail",
      "_:":"1581104199337"}},
    "response":{"numFound":2,"start":0,"docs":[
      {
        "id":"V5169400C3",
        "name":["CORSAIR ValueSelect 1GB 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) System Memory - Retail"],
        "manu":["Corsair Microsystems Inc."],
        "manu_id_s":"corsair",
        "cat":["electronics",
          "memory"],
        "price":["74.99"],
        "popularity":["7"],
        "inStock":["true"],
        "store":["37.7752,-100.8232"],
        "manufacturedate_dt":"2006-02-13T15:26:37Z",
        "payloads":["electronics|4.0 memory|2.0"],
        "manu_str":["Corsair Microsystems Inc."],
        "_version_":"1657713075921354752",
        "cat_str":["electronics",
          "memory"],
        "name_str":["CORSAIR ValueSelect 1GB 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) System Memory - Retail"],
        "store_str":["37.7752,-100.8232"],
        "payloads_str":["electronics|4.0 memory|2.0"]},
      {
        "id":"TnINX2048-3200P80",
        "name":["CORSAIR XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail"],
        "manu":["Corsair Microsystems Inc."],
        "manu_id_s":"corsair",
        "cat":["electronics",
          "memory"],
        "features":["CAS latency 2, 2-3-3-6 timing, 2.75v, unbuffered, heat-spreader"],
        "price":["185.0"],
        "popularity":["5"],
        "inStock":["true"],
        "store":["37.7752,-122.4232"],
        "manufacturedate_dt":"2006-02-13T15:26:37Z",
        "payloads":["electronics|6.0 memory|3.0"],
        "manu_str":["Corsair Microsystems Inc."],
        "_version_":"1657713075919257800",
        "cat_str":["electronics",
          "memory"]}
    ]}
  }

```

Activate Windows

Go to Settings to activate Windows.

Fig 3: Searching outcome (Solr)

Clipboard		Font	Alignment	Number	Styles			
A1	Product_Title							
	A	B	C	D	E	F	G	H
1	Product_Title	Price	Shop	Category_Id				
2	COLORFUL GT710 2GD3 2GB GDDR3 GRAPHICS CARD	3800	DOLPHIN COMPUTERS	1				
3	GIGABYTE GEFORCE RTX 2060 WINDFORCE OC 6G 6GB DDR6 GRAPHICS CARD #GA-N2060WF2OC-6GD	38000	DOLPHIN COMPUTERS	1				
4	GIGABYTE GEFORCE RTX 2060 OC 6GB GDDR6 GRAPHICS CARD #GV-N2060OC-6GD	37200	DOLPHIN COMPUTERS	1				
5	GIGABYTE NVIDIA GEFORCE GT 710 2GB DDR3 GRAPHICS CARD #GV-N710D3-2GL	5200	DOLPHIN COMPUTERS	1				
6	MSI GT 710-2GD3H LP 2GB DDR3 GRAPHICS CARD	4500	DOLPHIN COMPUTERS	1				
7	GIGABYTE NVIDIA GEFORCE GT 1030 OC 2GB DDR5 GRAPHICS CARD	8500	DOLPHIN COMPUTERS	1				
8	ASUS GT 710 2GB DDR3 GRAPHICS CARD	4800	DOLPHIN COMPUTERS	1				
9	ASROCK PHANTOM GAMING X RADEON RX590 8G OC GRAPHICS CARD	23500	DOLPHIN COMPUTERS	1				
10	GIGABYTE GTX 1050 TI OC 4GB DDR5 GRAPHICS CARD	17500	DOLPHIN COMPUTERS	1				
11	ZOTAC GAMING GEFORCE RTX 2060 TWIN FAN 6GB GRAPHICS CARD	38000	BD STALL	1				
12	XFX AMD RADEON RX 570 RS XXX EDITION 8GB GRAPHICS CARD	15500	BD STALL	1				
13	GIGABYTE GEFORCE 1030 2GB DDR5 LOW PROFILE GRAPHICS CARD	8400	BD STALL	1				
14	AFOX NVIDIA GEFORCE G210 1GB DDR3 DESKTOP GRAPHICS CARD	2700	BD STALL	1				
15	ASUS EXPEDITION GEFORCE GTX 1050 TI 4GB GDDR5 GRAPHICS CARD	17999	BD STALL	1				
16	MSI GEFORCE GTX 1660 TI GAMING X 6GB GDDR6 GRAPHICS CARD	33500	BD STALL	1				
17	MSI RADEON RX 570 4GB GDDR5 GRAPHICS CARD	13100	BD STALL	1				
18	VIEW ONE NVIDIA GEFORCE GT 710 DDR3 1GB GRAPHICS CARD	2600	BD STALL	1				
19	GIGABYTE GEFORCE GTX 1050 TI G1 GAMING 4GB DDR5 GRAPHICS CARD	9800	BD STALL	1				
20	MSI ARMOR GEFORCE GTX 1070 8GB GDDR5 GAMING GRAPHICS CARD	26000	BD STALL	1				
21	ASUS NVIDIA GEFORCE GT 1030 2GB GDDR5 GRAPHICS CARD	8500	BD STALL	1				
22	GIGABYTE GEFORCE RTX 2080 WINDFORCE OC 8G GDDR6 GRAPHICS	78000	BD STALL	1				
23	GIGABYTE NVIDIA GEFORCE GT 710 2GB DDR5 GRAPHICS CARD	5400	BD STALL	1				
24	ASUS PHOENIX GEFORCE GTX 1050 3GB GDDR5 GRAPHICS CARD	15800	BD STALL	1				
25	GIGABYTE NVIDIA GEFORCE GT 1030 2GB GRAPHICS CARD	8000	BD STALL	1				
26	ASUS ROG STRIX RX-580 8GB GDDR5 PCI 3.0 GAMING GRAPHICS CARD	25500	BD STALL	1				
27	ASUS NVIDIA GEFORCE GT 710 2GB DDR5 GRAPHICS CARD	6200	BD STALL	1				
28	COLORFUL IGAME GEFORCE GTX 1050TI VULCAN U 4G EC VIDEO CARD	15500	BD STALL	1				
29	ASUS DUAL SERIES RADEON RX 580 OC 8GB GDDR5 GRAPHICS CARD	26000	BD STALL	1				
30	GIGABYTE GEFORCE GTX 1050 TI G1 GAMING 4GB VIDEO CARD	17900	BD STALL	1				
31	ASUS ROG STRIX GEFORCE RTX 2080 OC EDITION 8GB GDDR6 GRAPHICS CARD	91500	STARTECH	1				
32	ASUS TUF GAMING X3 GEFORCE GTX 1660 ADVANCED EDITION 6GB GDDR5 GRAPHICS CARD	27800	STARTECH	1				
33	ASUS TUF GAMING X3 RADEON RX 5700 XT OC 8GB GRAPHICS CARD	48300	STARTECH	1				
34	COOLER MASTER UNIVERSAL VGA HOLDER FOR ALL SIZE TOWER CHASSIS	1300	STARTECH	1				
35	GALAX GEFORCE GT 1030 EXOC WHITE 2GB GDDR5 GRAPHICS CARD	8000	STARTECH	1				
36	GIGABYTE GEFORCE RTX 2060 SUPER WINDFORCE OC 8GB GRAPHICS CARD	42500	STARTECH	1				
37	GIGABYTE GT 1030 2GB OC GRAPHICS CARD	8000	STARTECH	1				
38	SAPPHIRE PULSE RADEON RX 570 4GB GDDR5 HDMI DP GRAPHICS CARD	13100	STARTECH	1				
39	SAPPHIRE PULSE RADEON RX 570 8GB GDDR5 GRAPHICS CARD	16200	STARTECH	1				
40	SAPPHIRE PULSE RADEON RX 580 8GB GRAPHICS CARD	20300	STARTECH	1				
41	XFX AMD RADEON RX 570 RS 8GB XXX EDITION GRAPHICS CARD	15800	STARTECH	1				
42	XFX RADEON RX 5700 XT THICC II ULTRA 8GB GDDR6 GRAPHICS CARD	43000	STARTECH	1				
43	ZOTAC GEFORCE GT 710 2GB DDR3 GRAPHICS CARD	4200	STARTECH	1				

Fig 4: Sample Dataset

Methodology

We have made a website based on web server based architecture that offers more time essential and advance functionalities. For this project, we made actual time efficiency for client. First of all we made a flow chart that what we need and work flow steps. All we need those data sets that means in Bangladesh a lot of well-known online seller actively provide this service. So our first goal is to collect all those web sites product data. After that we reorganized this data set like remove unnecessary info from our data set. For this thing we use python data scraping technology. It's a python library and an algorithm that can store data from websites to csv or xml file. Then we use another python searching algorithm called python Solr for applying search query to find out selective data from data set. After this we just stack those searched data in another csv file for our database. One more thing we already done after we search, we already use python general algorithm to find out minimum price by comparing price column of our csv file. And the compared results stored in our final preeminent data set. After finalize our data set we started to design our application's database table and there internal relationships. And we use php and its one of the framework laravel for this database design. Then we started to design User interface (UI) by using HTML, Css, java script and media query. Some additional feather we implement here for our client like wish list and mail notifications for further query or reminder. So we can analyze our client movement and what they want.

Discussion

At the end of the project we completed our main theme of our project that we provide the cheapest product provider in our website. We face a lot of issue to make it happen but the main thing is we learned a lot from this project.

We face a lot of issue when we gather our data set for this project. Data scraping is easy with python library but data organization and data filtering bother us a lot. When we scrap data from a website by using python bs4 library we can store data but there is a lot of unnecessary character comes so that we can run our search query in solr that's why we convert data name in uppercase letter and remove all character except English alphabet. Another big issue that we face a lot is the price tag of a product. Every shop design their website at their own style some use Bangla font some use English so we need to convert it to a single style for our search and minimum price finding algorithm. After filtering our data set we faced a big problem in python solr data query process. Actually we didn't find our exact data after searching some key word. But solr documentation helps us a lot to figure out the main problem. Then we go for design our database and when we started designing our database we just face some relational problem. After completing our demo data base design we found out that issue and solve it. Then we started to implement our front end design and our teammate doing it very well and clean and then we put our data to complete user interface. In this process we use laravel & we face some common issues and we fixed it without trouble.

Design Impacts & Cost

In this section, we discuss about our time cost, improvements and about future work

Time Cost

Task	Working Hours
Theoretical Study	25 Hours
Data Scraping	15 Hours
Data analysis	4 Hours
Pre-processing	4 Hours
Solr implementation	6 Hours
Searching data and price comparison	6 Hours
Testing and debug	10 Hours
Database design and implementation	20 Hours
UI Design and Implementation	10 Hours
Total Time	100 Hours

Improvement

There are lots of data that has same name and tag and our project searching algorithm fully depends on products title. Because we are searching same products from different shop and it is so difficult to find out because every owner of the shop they put their product name in their own way. If the comparison is between 2 different products then it's easier to find out the output. As example: **GIGABYTE GEFORCE RTX 2060 WINDFORCE OC 6G 6GB DDR6 GRAPHICS CARD #GA-N2060WF2OC-6GD** & **GIGABYTE GEFORCE RTX 2060 6GB DDR6 GRAPHICS CARD #GA-N2060WF2OC-6GD** . This 2 title has 90% same words and both are not same product.

Future work

In our project everything is running good with front end and backend. The main object of our project is scraped data and search same product from different source and compare price. So we use url of a product so that client can re-direct to the main shop easily. But urls are only available if the products are in stock. Basically if the product going to stock out from that shop who gave the minimum prices then the url become unavailable. And some of the shop use Bangla language for product title, but in our project we only working on English language. These things we can add in our projects future work.

Conclusion

In this project we have made a product preeminent price which is based on a web server architecture that offers more essential and advanced functionalities. For this project, we made an actual data set for scraping and find out the cheapest product from a lot of data that we show up in our web based application product preeminent price. To control all the functionalities, we created a database in our local host system. The final result of our approach is to provide actual data that help the people who need to know who provide the cheapest price in Bangladesh online sales service or online market place like daraz, techlandbd, startech, Ryan's computers and many more. Our application working system is so simple, its pre search product result just show up our website and provide a bunch of information and the most important information is which seller provide the cheapest price of a particular product and what other seller offers then. We believe and hope that our web based application will be able to eradicate the obstacles that are faced by so many people who has not enough time to spend in market to find out to compare price to one another shop in their daily life and at the same time it will give mobility in their way of life.

Reference

[1] Reference 1: Beautiful Soup 4 (bs4). Beautiful Soup is a Python library for pulling data out of HTML and XML files

[2] Reference 2: Solrpy is a Python client for solr, an enterprise search server built on top of lucene. Solrpy allows you to add documents to a Solr instance, and then to perform queries and gather search results from Solr using Python.

[3] [Reference 3: Laravel](#) the PHP framework for web artisans.

[4] Reference 3: Bootstrap for web user interface design.

[5] Scrapy 2.4 documentation

[6] pandas powerful python data analysis toolkit

[7] Web Scraping in the Statistics and Data Science Curriculum: Challenges and Opportunities

Acknowledgement

First of all, we would like to thank Almighty for all the fate related to our studies and secondly to express our profound gratitude to our honorable course instructor **Mohammad Ashrafuzzaman Khan**, for his constant and meticulous supervision, valuable suggestions, his patience and encouragement to complete the project work.

We would also thank the ECE department of North South University for providing us with the opportunity to have an industrial level design experience as part of our curriculum for the undergraduate program.

Finally, we would like to thank our families, friends, classmates and everybody who supported us and provided with guidance for the completion of this project.

Appendix

Code

Python Data Scraping Code

```
from bs4 import BeautifulSoup
import requests
import time
import threading

# output csv file declared here
filename = "data.csv"

f = open(filename, 'w', encoding='utf-8-sig')

headers = "Product_Title,Price,Shop_Name,URL,Image_Url,Product_Code,Category_Id\n"

f.write(headers)

def webscrapBdstall():
    pages = [1,2]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.bdstall.com/graphics-card/{}'.format(page)).text

        # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html
```

```

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

product_thumb = body.find_all('div', class_='row product-cat-box s-top')

# using loop for grabbing whole page data

for product in product_thumb:

    product_name = product.find(
        'div', class_='six columns product-cat-box-text')

    title = product_name.a.text.upper()

    product_price = product.find('div', class_='product-price')

    tk = product_price.text.replace(",","").replace("₹","")

    product_link = product.find('div',class_='six columns product-cat-box-text')

    link = product_link.a['href']

    image_url = product.find('div', class_='seven columns')

    url = image_url.img['src']

    data =title + "," + tk + "," + "BDSTALL" + "," + link + "," + url + "," + "1" + "," + "1" + "\n"

```

```
f.write(data)
```

```
time.sleep(5)
```

```
def webscrapCvillage():
```

```
    pages = [1,2,3,4,5,6,7,8,9]
```

```
    # declared the url directory and store it in a variable
```

```
    # techland gpu section
```

```
    for page in pages:
```

```
        source_link = requests.get('https://www.village-bd.com/graphics-card/{}'.format(page)).text
```

```
        soup = BeautifulSoup(source_link,'lxml')
```

```
        # search element from specified url html
```

```
        body = soup.find('body')
```

```
        # here product-thumb is a css class so that i used it as a variable for better understanding
```

```
        productInfo = soup.find_all('li',class_='col-sm-3')
```

```
        # using loop for grabing whole page data
```

```
        for product in productInfo:
```

```

product_name = product.find ('div',class_='pro-name')

title = product_name.a.text.upper()

product_price = product.find('div',class_='price')

tk = product_price.text.replace(",","").replace("৳","")

product_link = product.find('div', class_='pro-name')

link = product_link.a['href']

image_url = product.find('div', class_='img-box')

url = image_url.img['src']

data =title + "," + tk + "," + "COMPUTER VILLAGE" + "," + link + "," + url + "," + "1" + "," + "1" + "\n"

f.write(data)

time.sleep(5)

```

```
def webscrapDolphin():
```

```
    source_link = requests.get('http://dolphin.computer/products?category=graphics-card').text
```

```
    # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
```

```

soup = BeautifulSoup(source_link, 'lxml')

# search element from specified url html

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

product_thumb = body.find_all('a', class_='product-card')

# using loop for grabbing whole page data

for product in product_thumb:

    product_name = product.find('span', class_='product-name')

    title = product_name.text.upper()

    product_price = product.find('span', class_='product-price')

    tk = product_price.text.replace(",","").replace("Tk", "")

    link = product['href']

    image_url = product.find('div',class_='image-holder')

    url = image_url.img['src']

    data =title + "," + tk + "," + "DOLPHIN" + "," + link + "," + url + "," + "1" + "," + "1" + "\n"

```

```
f.write(data)
```

```
time.sleep(5)
```

```
def webscrapRyans():
```

```
    pages = [1,2,3,4,5,6,7]
```

```
    for page in pages:
```

```
        source_link = requests.get(
```

```
            'https://ryanscomputers.com/grid/desktop-component-graphics-  
card?page={}'.format(page)).text
```

```
        soup = BeautifulSoup(source_link, 'xml')
```

```
        # search element from specified url html
```

```
        body = soup.find('body')
```

```
        # here product-thumb is a css class so that i used it as a variable for better understanding
```

```
        product_info = body.find_all('div', class_='product-box')
```

```
        # using loop for grabbing whole page data
```

```
        for product in product_info:
```

```
            product_name = product.find('div', class_='product-content-info')
```

```
            title = product_name.a.text.replace(",","").upper()
```

```

product_price = product.find('span', class_='price')

tk = product_price.text.replace("Tk", "").replace(",","").replace("BDT", "").strip()

product_link = product.find('div', class_='product-content-info')

link = product_link.a['href']

image_url = product.find('div', class_='product-thumb')
url = image_url.img['src']

data =title + "," + tk + "," + "RYANS" + "," + link + "," + url + "," + "1" + "," + "1" + "\n"

f.write(data)
time.sleep(5)

def webscrapStartech():
    pages = [1,2,3,4,5]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.startech.com.bd/component/graphics-card?page={}'.format(page)).text

        soup = BeautifulSoup(source_link, 'lxml')

```

```

# search element from specified url html

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

productInfo = body.find_all('div', class_='col-xs-12 col-md-4 product-layout grid')

# using loop for grabbing whole page data

for product in productInfo:

    product_name = product.find('h4', class_='product-name')

    title = product_name.a.text.upper().replace(",","")

    product_price = product.find('div', class_='price space-between')

    tk = product_price.span.text.replace(",","").replace("₹","")

    product_link = product.find('h4', class_='product-name')

    link = product_link.a['href']

    image_url = product.find('div',class_ = 'img-holder')

    url = image_url.img['src']

    data =title + "," + tk + "," + "STARTECH" + "," + link + "," + url + "," + "1" + "," + "1" + "\n"

```



```

        f.write(data)
    time.sleep(5)

def webscrapTechland():

    pages = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page={}'.format(page)).text

        # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
        soup = BeautifulSoup(source_link, 'xml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        product_thumb = soup.find_all('div', class_='product-thumb')

        # using loop for grabbing whole page data

        for product in product_thumb:

            product_name = product.find('div', class_='name')

```

```

title = product_name.a.text.strip()

product_price = product.find('div', class_='price')

tk = product_price.span.text

product_link = product.find('div', class_='name')

link = product_link.a['href']

image_url = product.find('a', class_='product-img')

url = image_url.img['src']


data = (title + "," + tk.replace("&", "").replace(",","") + "," + "TECHLAND" + "," + link + "," + url + "," +
"1" + "," + "1"+"\n")

f.write(data)

time.sleep(5)

def bdstallKeyboard():
    pages = [1,2]

    # declared the url directory and store it in a variable
    # techland gpu section

```

for page in pages:

```
source_link = requests.get('https://www.bdSTALL.com/computer-keyboard/{}'.format(page)).text
```

```
# source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
```

```
soup = BeautifulSoup(source_link, 'xml')
```

```
# search element from specified url html
```

```
body = soup.find('body')
```

```
# here product-thumb is a css class so that i used it as a variable for better understanding
```

```
product_thumb = body.find_all('div', class_='row product-cat-box s-top')
```

```
# using loop for grabbing whole page data
```

```
for product in product_thumb:
```

```
    product_name = product.find(  
        'div', class_='six columns product-cat-box-text')
```

```
    title = product_name.a.text.upper()
```

```
    product_price = product.find('div', class_='product-price')
```

```
    tk = product_price.text.replace(",","").replace("৳","")
```

```
    product_link = product.find('div',class_='six columns product-cat-box-text')
```

```
    link = product_link.a['href']
```

```

image_url = product.find('div', class_='seven columns')

url = image_url.img['src']

data =title + "," + tk + "," + "BDSTALL" + "," + link + "," + url + "," + "1" + "," + "2" + "\n"

f.write(data)
time.sleep(5)

def bdstallMouse():

    pages = [1,2]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.bdstall.com/mouse/{}'.format(page)).text

        # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        product_thumb = body.find_all('div', class_='row product-cat-box s-top')

```

```
# using loop for grabbing whole page data
```

```
for product in product_thumb:
```

```
    product_name = product.find(
        'div', class_='six columns product-cat-box-text')
```

```
    title = product_name.a.text.upper()
```

```
    product_price = product.find('div', class_='product-price')
```

```
    tk = product_price.text.replace(",","").replace("€","")
```

```
    product_link = product.find('div',class_='six columns product-cat-box-text')
```

```
    link = product_link.a['href']
```

```
    image_url = product.find('div', class_='seven columns')
```

```
    url = image_url.img['src']
```

```
    data =title + "," + tk + "," + "BDSTALL" + "," + link + "," + url + "," + "1" + "," + "3" + "\n"
```

```
    f.write(data)
```

```
time.sleep(5)
```

```
def bdstallMonitor():
```

```

pages = [1,2,3,4]

# declared the url directory and store it in a variable
# techland gpu section
for page in pages:
    source_link = requests.get('https://www.bdstall.com/monitor/{}'.format(page)).text

    # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
    soup = BeautifulSoup(source_link, 'lxml')

    # search element from specified url html

    body = soup.find('body')

    # here product-thumb is a css class so that i used it as a variable for better understanding

    product_thumb = body.find_all('div', class_='row product-cat-box s-top')

    # using loop for grabbing whole page data

    for product in product_thumb:

        product_name = product.find(
            'div', class_='six columns product-cat-box-text')

        title = product_name.a.text.upper()

        product_price = product.find('div', class_='product-price')

```

```

tk = product_price.text.replace(",","").replace("৳","")

product_link = product.find('div',class_='six columns product-cat-box-text')

link = product_link.a['href']

image_url = product.find('div', class_='seven columns')

url = image_url.img['src']

data =title + "," + tk + "," + "BDSTALL" + "," + link + "," + url + "," + "1" + "," + "4" + "\n"

f.write(data)

time.sleep(5)

def computervillageKeyboard():

    pages = [1,2,3,4,5,6,7]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.village-bd.com/keyboard/{}'.format(page)).text

        soup = BeautifulSoup(source_link,'lxml')

        # search element from specified url html

```

```
body = soup.find('body')
```

```
# here product-thumb is a css class so that i used it as a variable for better understanding
```

```
productInfo = soup.find_all('li',class_='col-sm-3')
```

```
# using loop for grabbing whole page data
```

```
for product in productInfo:
```

```
    product_name = product.find ('div',class_='pro-name')
```

```
    title = product_name.a.text.upper()
```

```
    product_price = product.find('div',class_='price')
```

```
    tk = product_price.text.replace(",","").replace("₹","")
```

```
    product_link = product.find('div', class_='pro-name')
```

```
    link = product_link.a['href']
```

```
    image_url = product.find('div', class_='img-box')
```

```
    url = image_url.img['src']
```

```
    data =title + "," + tk + "," + "COMPUTER VILLAGE" + "," + link + "," + url + "," + "1" + "," + "2" + "\n"
```



```

f.write(data)

time.sleep(5)

def computervillageMouse():

    pages = [1,2,3,4,5,6]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link = requests.get('https://www.village-bd.com/mouse/{}'.format(page)).text

        soup = BeautifulSoup(source_link,'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        productInfo = soup.find_all('li',class_='col-sm-3')

        # using loop for grabbing whole page data

        for product in productInfo:

```

```

product_name = product.find ('div',class_='pro-name')

title = product_name.a.text.upper()

product_price = product.find('div',class_='price')

tk = product_price.text.replace(",","").replace("₹","")

product_link = product.find('div', class_='pro-name')

link = product_link.a['href']

image_url = product.find('div', class_='img-box')

url = image_url.img['src']

data =title + "," + tk + "," + "COMPUTER VILLAGE" + "," + link + "," + url + "," + "1" + "," + "3" + "\n"

f.write(data)

time.sleep(5)

def computervillageMonitor():

    pages = [1,2,3,4,5,6,7,8,9,10,11,12]

    # declared the url directory and store it in a variable

```

```

# techland gpu section
for page in pages:
    source_link = requests.get('https://www.village-bd.com/monitor/{}'.format(page)).text

    soup = BeautifulSoup(source_link,'lxml')

    # search element from specified url html

    body = soup.find('body')

    # here product-thumb is a css class so that i used it as a variable for better understanding

    productInfo = soup.find_all('li',class_='col-sm-3')

    # using loop for grabbing whole page data

    for product in productInfo:

        product_name = product.find ('div',class_='pro-name')

        title = product_name.a.text.upper().replace(",","")

        product_price = product.find('div',class_='price')

        tk = product_price.text.replace(",","").replace("৳","").replace("Please call at 01713240766","0")

        product_link = product.find('div', class_='pro-name')

```

```

link = product_link.a['href']

image_url = product.find('div', class_='img-box')

url = image_url.img['src']

data =title + "," + tk + "," + "COMPUTER VILLAGE" + "," + link + "," + url + "," + "1" + "," + "4" + "\n"

f.write(data)

time.sleep(5)

def ryansKeyboard():
    pages = [1,2,3,4,5]
    for page in pages:
        source_link = requests.get(
            'https://ryanscomputers.com/grid/desktop-component-keyboard?page={}'.format(page)).text

        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        product_info = body.find_all('div', class_='product-box')

```

```
# using loop for grabbing whole page data
```

```
for product in product_info:
```

```
    product_name = product.find('div', class_='product-content-info')
```

```
    title = product_name.a.text.replace(",","").upper()
```

```
    product_price = product.find('span', class_='price')
```

```
    tk = product_price.text.replace("Tk","").replace(",","").replace("BDT","").strip()
```

```
    product_link = product.find('div', class_='product-content-info')
```

```
    link = product_link.a['href']
```

```
    image_url = product.find('div', class_='product-thumb')
```

```
    url = image_url.img['src']
```

```
    data =title + "," + tk + "," + "RYANS" + ","+ link + "," + url + "," + "1" + "," + "2" + "\n"
```

```
    f.write(data)
```

```
    time.sleep(5)
```

```
def ryansMouse():
```

```
    pages = [1,2,3,4,5,6,7]
```

```
    for page in pages:
```

```
        source_link = requests.get(
```

```
            'https://ryanscomputers.com/grid/desktop-component-mouse?page={}'.format(page)).text
```

```
soup = BeautifulSoup(source_link, 'lxml')

# search element from specified url html

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

product_info = body.find_all('div', class_='product-box')

# using loop for grabbing whole page data

for product in product_info:

    product_name = product.find('div', class_='product-content-info')

    title = product_name.a.text.replace(", ", "").upper()

    product_price = product.find('span', class_='price')

    tk = product_price.text.replace("Tk", "").replace(", ", "").replace("BDT", "").strip()

    product_link = product.find('div', class_='product-content-info')

    link = product_link.a['href']

    image_url = product.find('div', class_='product-thumb')

    url = image_url.img['src']
```

```

data =title + "," + tk + "," + "RYANS" + ","+ link + "," + url + "," + "1" + "," + "3" + "\n"

f.write(data)

time.sleep(5)

def ryansMonitor():

    pages = [1,2,3,4,5,6,7]
    for page in pages:
        source_link = requests.get(
            'https://ryanscomputers.com/grid/all-monitor-all-brand?page={}'.format(page)).text

        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        product_info = body.find_all('div', class_='product-box')

        # using loop for grabing whole page data

        for product in product_info:

            product_name = product.find('div', class_='product-content-info')

            title = product_name.a.text.replace(",","").upper()

```

```

product_price = product.find('span', class_='price')

tk = product_price.text.replace("Tk", "").replace(",","").replace("BDT", "").strip()

product_link = product.find('div', class_='product-content-info')

link = product_link.a['href']

image_url = product.find('div', class_='product-thumb')
url = image_url.img['src'].replace(",","")

data =title + "," + tk + "," + "RYANS" + "," + link + "," + url + "," + "1" + "," + "4" + "\n"

f.write(data)
time.sleep(5)

def startechKeyboard():

    pages = [1,2,3,4,5,6,7,8]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link =
requests.get('https://www.startech.com.bd/accessories/keyboards?page={}'.format(page)).text

    soup = BeautifulSoup(source_link, 'lxml')

```



```

# search element from specified url html

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

productInfo = body.find_all('div', class_='col-xs-12 col-md-4 product-layout grid')

# using loop for grabbing whole page data

for product in productInfo:

    product_name = product.find('h4', class_='product-name')

    title = product_name.a.text.upper().replace(",","")

    product_price = product.find('div', class_='price space-between')

    tk = product_price.span.text.replace(",","").replace("₹","")

    product_link = product.find('h4', class_='product-name')

    link = product_link.a['href']

    image_url = product.find('div',class_ = 'img-holder')

    url = image_url.img['src']

    data =title + "," + tk + "," + "STARTECH" + "," + link + "," + url + "," + "1" + "," + "2" + "\n"

```

```

        f.write(data)
    time.sleep(5)

def startechMouse():
    pages = [1,2,3,4,5,6,7,8,9]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:
        source_link =
requests.get('https://www.startech.com.bd/accessories/mouse?page={}'.format(page)).text

        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        productInfo = body.find_all('div', class_='col-xs-12 col-md-4 product-layout grid')

        # using loop for grabbing whole page data

        for product in productInfo:

            product_name = product.find('h4', class_='product-name')

            title = product_name.a.text.upper()

```

```

product_price = product.find('div', class_='price space-between')

tk = product_price.span.text.replace(",","").replace("৳","")

product_link = product.find('h4', class_='product-name')

link = product_link.a['href']

image_url = product.find('div',class_ = 'img-holder')

url = image_url.img['src']

data =title + "," + tk + "," + "STARTECH" + ","+ link + "," + url + "," + "1" + "," + "3" + "\n"

f.write(data)

time.sleep(5)

def startechMonitor():

    pages = [1,2,3,4,5,6,7,8,9,10]

    # declared the url directory and store it in a variable
    # techland gpu section
    for page in pages:

        source_link = requests.get('https://www.startech.com.bd/monitor?page={}'.format(page)).text

        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

```

```

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

productInfo = body.find_all('div', class_='col-xs-12 col-md-4 product-layout grid')

# using loop for grabbing whole page data

for product in productInfo:

    product_name = product.find('h4', class_='product-name')

    title = product_name.a.text.upper().replace(",","")

    product_price = product.find('div', class_='price space-between')

    tk = product_price.span.text.replace(",","").replace("₹","")

    product_link = product.find('h4', class_='product-name')

    link = product_link.a['href']

    image_url = product.find('div',class_ = 'img-holder')

    url = image_url.img['src']

    data =title + "," + tk + "," + "STARTECH" + "," + link + "," + url + "," + "1" + "," + "4" + "\n"

    f.write(data)

```

```
time.sleep(5)
```

```
def techlandKeyboard():
```

```
    pages = [1,2,3,4,5,6,7,8,9,10]
```

```
    for page in pages:
```

```
        source_link = requests.get(
```

```
            'https://www.techlandbd.com/accessories/computer-keyboard?page={}'.format(page)).text
```

```
        soup = BeautifulSoup(source_link, 'lxml')
```

```
        # search element from specified url html
```

```
        body = soup.find('body')
```

```
        # here product-thumb is a css class so that i used it as a variable for better understanding
```

```
        product_thumb = soup.find_all('div', class_='product-thumb')
```

```
        # using loop for grabbing whole page data
```

```
        for product in product_thumb:
```

```
            product_name = product.find('div', class_='name')
```

```
            title = product_name.a.text.replace(",","").upper()
```

```
            product_price = product.find('span', class_='price-normal')
```

```

if(product_price):
    tk = product_price.text.replace("Tk","").replace(",","").replace("₹","").strip()
else:
    tk = '0000'

product_link = product.find('div',class_='name')

link = product_link.a['href']

image_url = product.find('a',class_='product-img')

url = image_url.img['src'].replace(",","")

data = (title + "," + tk.replace("₹","").replace(",","") + "," + "TECHLAND" + "," + link + "," + url + "," +
"1" + "," + "2" + "\n")

f.write(data)

time.sleep(5)

def techlandMouse():

pages = [1,2,3,4,5,6,7,8,9,10,11,12]
for page in pages:

source_link = requests.get(
    'https://www.techlandbd.com/accessories/shop-computer-mouse?page={}'.format(page)).text
soup = BeautifulSoup(source_link, 'lxml')

```

```
# search element from specified url html
```

```
body = soup.find('body')
```

```
# here product-thumb is a css class so that i used it as a variable for better understanding
```

```
product_thumb = soup.find_all('div', class_='product-thumb')
```

```
# using loop for grabbing whole page data
```

```
for product in product_thumb:
```

```
    product_name = product.find('div', class_='name')
```

```
    title = product_name.a.text.replace(",","").upper()
```

```
    product_price = product.find('span', class_='price-normal')
```

```
    if(product_price):
```

```
        tk = product_price.text.replace("Tk","").replace(",","").replace("₹ ","").strip()
```

```
    else:
```

```
        tk = '0000'
```

```
    product_link = product.find('div',class_='name')
```

```
    link = product_link.a['href']
```

```
    image_url = product.find('a',class_='product-img')
```

```
    url = image_url.img['src'].replace(",","")
```

```
data = (title + "," + tk.replace(" ", "").replace(",","") + "," + "TECHLAND" + "," + link + "," + url + "," +  
"1" + "," + "3" + "\n")
```

```
f.write(data)
```

```
time.sleep(5)
```

```
def techlandMonitor():
```

```
pages = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
```

```
for page in pages:
```

```
source_link = requests.get(
```

```
'https://www.techlandbd.com/computer-monitor?page={}'.format(page)).text
```

```
soup = BeautifulSoup(source_link, 'lxml')
```

```
# search element from specified url html
```

```
body = soup.find('body')
```

```
# here product-thumb is a css class so that i used it as a variable for better understanding
```

```
product_thumb = soup.find_all('div', class_='product-thumb')
```

```
# using loop for grabbing whole page data
```

```
for product in product_thumb:
```



```

product_name = product.find('div', class_='name')

title = product_name.a.text.replace(",","").upper()

product_price = product.find('span', class_='price-normal')
if(product_price):
    tk = product_price.text.replace("Tk","").replace(",","").replace("₹ ","").strip()
else:
    tk = '0000'

product_link = product.find('div',class_='name')

link = product_link.a['href']

image_url = product.find('a',class_='product-img')

url = image_url.img['src'].replace(",","")

data = (title + "," + tk.replace("₹","").replace(",","") + "," + "TECHLAND" + "," + link + "," + url + "," +
"1" + "," + "4" + "\n")

f.write(data)

time.sleep(5)

def dolphinKeyboard():

    source_link = requests.get('https://dolphin.computer/products?category=keyboard').text

```

```
# source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
soup = BeautifulSoup(source_link, 'xml')

# search element from specified url html

body = soup.find('body')

# here product-thumb is a css class so that i used it as a variable for better understanding

product_thumb = body.find_all('a', class_='product-card')

# using loop for grabbing whole page data

for product in product_thumb:

    product_name = product.find('span', class_='product-name')

    title = product_name.text.upper()

    product_price = product.find('span', class_='product-price')

    tk = product_price.text.replace(",","").replace("Tk", "")

    link = product['href']

    image_url = product.find('div',class_='image-holder')

    url = image_url.img['src']
```

```

data =title + "," + tk + "," + "DOLPHIN" + "," + link + "," + url + "," + "1" + "," + "2" + "\n"

f.write(data)

time.sleep(5)

def dolphinMonitor():

    pages = [1,2,3]
    for page in pages:
        source_link =
requests.get('https://dolphin.computer/products?category=monitor&sort=latest&page={}'.format(page)
).text

        # source_link = requests.get('https://www.techlandbd.com/shop-graphics-card?page=1').text
        soup = BeautifulSoup(source_link, 'lxml')

        # search element from specified url html

        body = soup.find('body')

        # here product-thumb is a css class so that i used it as a variable for better understanding

        product_thumb = body.find_all('a', class_='product-card')

        # using loop for grabbing whole page data

        for product in product_thumb:

            product_name = product.find('span', class_='product-name')

```

```

title = product_name.text.upper().replace(", ", "")

product_price = product.find('span', class_='product-price')

tk = product_price.text.replace(", ", "").replace("Tk", "").replace(".00 34000.00", "").replace(".00
6500.00", "")

link = product['href']

image_url = product.find('div', class_='image-holder')

url = image_url.img['src']

data = title + ", " + tk + ", " + "DOLPHIN" + ", " + link + ", " + url + ", " + "1" + ", " + "4" + "\n"

f.write(data)

time.sleep(5)

threading.Thread(target = webscrapBdstall).start()

threading.Thread(target = webscrapCvillage).start()

threading.Thread(target = webscrapDolphin).start()

```

```
threading.Thread(target = webscrapRyans).start()
```

```
threading.Thread(target = webscrapStartech).start()
```

```
threading.Thread(target = webscrapTechland).start()
```

```
threading.Thread(target = bdstallKeyboard).start()
```

```
threading.Thread(target = bdstallMouse).start()
```

```
threading.Thread(target = bdstallMonitor).start()
```

```
threading.Thread(target = computervillageKeyboard).start()
```

```
threading.Thread(target = computervillageMouse).start()
```

```
threading.Thread(target = computervillageMonitor).start()
```

```
threading.Thread(target = ryansKeyboard).start()
```

```
threading.Thread(target = ryansMouse).start()
```

```
threading.Thread(target = ryansMonitor).start()
```

```
threading.Thread(target = startechKeyboard).start()
```

```
threading.Thread(target = startechMouse).start()
```

```
threading.Thread(target = startechMonitor).start()
```

```
threading.Thread(target = techlandKeyboard).start()
```

```
threading.Thread(target = techlandMouse).start()
```

```
threading.Thread(target = techlandMonitor).start()
```

```
threading.Thread(target = dolphinKeyboard).start()
```

```
threading.Thread(target = dolphinMonitor).start()
```

Laravel Back-End Code

Web.php

```
<?php
```

```
/*
```

```
|-----
```

```
| Web Routes
```

```
|-----
```

```
|
```

```
| Here is where you can register web routes for your application. These
```

```
| routes are loaded by the RouteServiceProvider within a group which
```

```
| contains the "web" middleware group. Now create something great!
```

```
|
```

```
*/
```

```
// Admin dashboard all controller start here
```

```
// dashboard login routes
```

```
Route::get('/admin', 'dashboardController@index');
```

```
Route::get('/dashboard', 'superadminController@index');
```

```
Route::post('/admindashboard', 'dashboardController@dashboard');
```

```
Route::get('/logout', 'superadminController@logout');
```

```
// add and show all category routes
```

```
Route::get('/addcategory', 'categoryController@index');
```

```
Route::get('/allcategory', 'categoryController@allCategory');
```

```
Route::post('/savecategory', 'categoryController@addCategory');
```

```
// CRUD category routes
```

```
Route::get('/deactivecategory/{category_id}', 'categoryController@deactiveCategory');
```

```
Route::get('/activecategory/{category_id}', 'categoryController@activeCategory');
```

```
Route::get('/editcategory/{category_id}','categoryController@editCategory');

Route::post('/updatecategory/{category_id}','categoryController@updateCategory');

Route::get('/deletecategory/{category_id}','categoryController@deleteCategory');


// add and show all products routes


Route::get('/addproduct','productController@index');

Route::post('/uploadproduct','productController@addProducts');


Route::get('/allproduct','productController@allProduct');

Route::get('/activate/{product_id}','productController@activateProduct');

Route::get('/deactivate/{product_id}','productController@deactivateProduct');

Route::get('/deleteproduct/{product_id}','productController@deleteProduct');

Route::get('/editproduct/{product_id}','productController@editProduct');

Route::post('/updateproduct/{productId}','productController@updateProduct');


// Admin dashboard all controller end here
```


Layout.php

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<!-- Mirrored from bootstrapmaster.com/live/metro/index.html by HTTrack Website Copier/3.x  
[XR&CO'2014], Mon, 08 Jan 2018 16:56:12 GMT -->
```

```
<head>
```

```
<!-- start: Meta -->
```

```
<meta charset="utf-8">
```

```
<title>Metro Admin Template - Metro UI Style Bootstrap Admin Template</title>
```

```
<meta name="description" content="Metro Admin Template.">
```

```
<meta name="author" content="Łukasz Holeczek">
```

```
<meta name="keyword" content="Metro, Metro UI, Dashboard, Bootstrap, Admin, Template, Theme,  
Responsive, Fluid, Retina">
```

```
<!-- end: Meta -->
```

```
<!-- start: Mobile Specific -->
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<!-- end: Mobile Specific -->
```

```

<!-- start: CSS -->

<link id="bootstrap-style" href="{{ asset('adminAssets/css/bootstrap.min.css') }}" rel="stylesheet">

<link href="{{ asset('adminAssets/css/bootstrap-responsive.min.css') }}" rel="stylesheet">

<link id="base-style" href="{{ asset('adminAssets/css/style.css') }}" rel="stylesheet">

<link id="base-style-responsive" href="{{ asset('adminAssets/css/style-responsive.css') }}"
rel="stylesheet">

<link
href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700italic,800italic,
400,300,600,700,800&subset=latin,cyrillic-ext,latin-ext' rel='stylesheet' type='text/css'>

<!-- <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous"> -->

<!-- end: CSS -->


<!-- The HTML5 shim, for IE6-8 support of HTML5 elements -->

<!--[if lt IE 9]>

    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>

    <link id="ie-style" href="css/ie.css" rel="stylesheet">

<![endif]-->


<!--[if IE 9]>

    <link id="ie9style" href="css/ie9.css" rel="stylesheet">

<![endif]-->


<!-- start: Favicon -->

<link rel="shortcut icon" href="img/favicon.ico">

<!-- end: Favicon -->

```

```
</head>
```

```
<body>
```

```
  <!-- start: Header -->
```

```
  <div class="navbar">
```

```
    <div class="navbar-inner">
```

```
      <div class="container-fluid">
```

```
        <a class="btn btn-navbar" data-toggle="collapse" data-target=".top-nav.nav-collapse,.sidebar-nav.nav-collapse">
```

```
          <span class="icon-bar"></span>
```

```
          <span class="icon-bar"></span>
```

```
          <span class="icon-bar"></span>
```

```
        </a>
```

```
        <a class="brand" href="index.html"><span></span></a>
```

```
  <!-- start: Header Menu -->
```

```
  <div class="nav-no-collapse header-nav">
```

```
    <ul class="nav pull-right">
```

```
      <!-- start: User Dropdown -->
```

```
      <li class="dropdown">
```

```
        <a class="btn dropdown-toggle" data-toggle="dropdown" href="#">
```

```
          <i class="halflings-icon white user"></i>
```

```
          {{ Session::get('name') }}
```

```
          <span class="caret"></span>
```

```
        </a>
```

```
        <ul class="dropdown-menu">
```

```

        <li class="dropdown-menu-title">
            <span>Account Settings</span>
        </li>
        <!-- <li><a href="#"><i class="halflings-icon user"></i> Profile</a></li> -->
        <li><a href="{{ url('/logout') }}"><i class="halflings-icon off"></i> Logout</a></li>
    </ul>
</li>
<!-- end: User Dropdown -->
</ul>
</div>
<!-- end: Header Menu -->

</div>
</div>
</div>
<!-- start: Header -->

<div class="container-fluid-full">
    <div class="row-fluid">

        <!-- start: Main Menu -->
        <div id="sidebar-left" class="span2">
            <div class="nav-collapse sidebar-nav">
                <ul class="nav nav-tabs nav-stacked main-menu">
                    <li><a href="{{ URL::to('/dashboard') }}"><i class="icon-bar-chart"></i><span
class="hidden-tablet">
                        Dashboard</span></a></li>
                    <li>
                        <a class="dropdown" href="#"><i class="icon-folder-close-alt"></i><span class="hidden-
tablet"> Category</a>
                    <ul>

```

```

        <li><a class="submenu" href="{ URL::to('/addcategory') }"><i class="icon-file-
alt"></i><span class="hidden-tablet">Add Category</span></a></li>

        <li><a class="submenu" href="{ URL::to('/allcategory') }"><i class="icon-file-
alt"></i><span class="hidden-tablet">All Category</span></a></li>

    </ul>

</li>

<li>

    <a class="dropdown" href="#"><i class="icon-folder-close-alt"></i><span class="hidden-
tablet"> Products</span></a>

    <ul>

        <li><a class="submenu" href="{ URL::to('/addproduct') }"><i class="icon-file-
alt"></i><span class="hidden-tablet">Add Products</span></a></li>

        <li><a class="submenu" href="{ URL::to('/allproduct') }"><i class="icon-file-
alt"></i><span class="hidden-tablet">All Products</span></a></li>

    </ul>

</li>

<li>

    <a class="dropdown" href="#"><i class="icon-folder-close-alt"></i><span class="hidden-
tablet"> Slider</span></a>

    <ul>

        <li><a class="submenu" href="addSlider.html"><i class="icon-file-alt"></i><span
class="hidden-tablet">Add Slider</span></a></li>

        <li><a class="submenu" href="allSlider.html"><i class="icon-file-alt"></i><span
class="hidden-tablet">All Slider</span></a></li>

    </ul>

</li>

<li><a href="manageOrder.html"><i class="icon-folder-open"></i><span class="hidden-
tablet">

    Manage Order</span></a></li>

```

```
        </ul>
    </div>
</div>
<div id="content" class="span10">

    @yield('adminContent')

    @yield('addCategory')

    @yield('allCategory')

    @yield('updateCategory')

    @yield('addProducts')

    @yield('allProducts')

    @yield('updateProduct')

</div>
</div>
</div>
```

```
<footer>
```

```
<p>
```

```
    <span style="text-align:left;float:left ">&copy; 2019 all rights reserved <a href=" " alt="Bootstrap
Themes ">minibazarbd.com</a></span>
```

```
    <span class="hidden-phone " style="text-align:right;float:right ">Powered by: <a
href="http://admintemplates.co/ " alt="Bootstrap Admin Templates ">Mishel</a></span>
```

</p>

</footer>

<!-- start: JavaScript-->

<script src="{{ asset('adminAssets/js/jquery-1.9.1.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery-migrate-1.0.0.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery-ui-1.10.0.custom.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.ui.touch-punch.js') }}"></script>

<script src="{{ asset('adminAssets/js/modernizr.js') }}"></script>

<script src="{{ asset('adminAssets/js/bootstrap.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.cookie.js') }}"></script>

<script src="{{ asset('adminAssets/js/fullcalendar.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.dataTables.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/excanvas.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.flot.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.flot.pie.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.flot.stack.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.flot.resize.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.chosen.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.uniform.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.cleditor.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.noty.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.elfinder.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.raty.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.iphone.toggle.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.uploadify-3.1.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.gritter.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.imagesloaded.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.masonry.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.knob.modified.js') }}"></script>

<script src="{{ asset('adminAssets/js/jquery.sparkline.min.js') }}"></script>

<script src="{{ asset('adminAssets/js/counter.js') }}"></script>

<script src="{{ asset('adminAssets/js/retina.js') }}"></script>

<script src="{{ asset('adminAssets/js/custom.js') }}"></script>

<!-- end: JavaScript-->

</body>

<!-- Mirrored from bootstrapmaster.com/live/metro/index.html by HTTrack Website Copier/3.x
[XR&CO'2014], Mon, 08 Jan 2018 16:56:47 GMT -->

</html>

Category.php

@extends('adminLayout')

@section('addCategory')

<div class="row-fluid sortable">

<div class="box span12">

@if (\$message = Session::get('msg'))

{{ \$message }}

{{ Session::put('msg',null) }}

@else

@endif

<div class="box-header" data-original-title>

<h2><i class="halflings-icon edit"></i>Add Category</h2>

<div class="box-icon">

<i class="halflings-icon wrench"></i>

```

        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
</div>
<div class="box-content">
    <form class="form-horizontal" action="{{ url('/savecategory') }}" method="post">
        {{ csrf_field() }}
        <fieldset>
            <div class="control-group">
                <label class="control-label" for="typeahead">Category Name</label>
                <div class="controls">
                    <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->
                    <input type="text" class="form-control" id="exampleInputEmail1"
name="categoryName" required="" placeholder="Enter Category Name">
                </div>
            </div>

            <div class="control-group">
                <label class="control-label">Publication Status</label>
                <div class="controls">
                    <label class="checkbox inline">
                        <input type="checkbox" id="inlineCheckbox1" name="categoryStat" required=""
value="1">
                    </label>
                </div>
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-primary">Add Category</button>
                <button type="reset" class="btn">Cancel</button>
            </div>

```

```

        </fieldset>
    </form>

</div>
</div>
<!--/span-->

</div>

@endsection
@extends('adminLayout')

@section('addProducts')

<div class="row-fluid sortable">
    <div class="box span12">
        <span style="color:lightcoral;">
            @if ($message = Session::get('msg'))
                {{ $message }}
                {{ Session::put('msg',null) }}

            @else

            @endif
        </span>
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon edit"></i><span class="break"></span>Add Products</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>

```

```

        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
</div>
<div class="box-content">
    <form class="form-horizontal" action="{{ url('/uploadproduct') }}" method="post"
    enctype="multipart/form-data">
        {{ csrf_field() }}
        <fieldset>
            <div class="control-group">
                <label class="control-label" for="typeahead">Product Name</label>
                <div class="controls">
                    <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
    provide="typeahead"> -->
                    <input type="text" class="form-control" id="exampleInputEmail1" required=""
    name="productName" placeholder="Enter Product Name">
                </div>
            </div>

            <div class="control-group">
                <label class="control-label" for="selectError3">Product Category</label>
                <div class="controls">
                    <select id="selectError3" name="productCategory" required="">
                        {{ $dropdownCategory = DB::table('tbl_category')
                        ->get()}}

                        @foreach($dropdownCategory as $vCategory)

                            <option value="{{ $vCategory->category_id }}"> {{ $vCategory->category_name
    }}</option>

                        @endforeach
                    </select>
                </div>
            </div>
        </fieldset>
    </form>
</div>

```

```

        </select>
    </div>
</div>

<div class="control-group">
    <label class="control-label" for="typeahead">Product Price</label>

    <div class="controls">
        <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->

        <input type="text" class="form-control" id="exampleInputEmail1" required=""
name="productPrice" placeholder="Enter Product Price">
    </div>
</div>

<div class="control-group">
    <label class="control-label" for="typeahead">Product Color</label>

    <div class="controls">
        <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->

        <input type="text" class="form-control" id="exampleInputEmail1" required=""
name="productColor" placeholder="Enter Product Color">
    </div>
</div>

<div class="control-group">
    <label class="control-label">Image Upload</label>

    <div class="controls">
        <input type="file" name="productImage">
    </div>
</div>

```

```
<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Description</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productDescription" id="textarea2" required="" rows="3"></textarea>
  </div>
</div>
```

```
<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Overview</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productOverview" id="textarea2" required="" rows="3"></textarea>
  </div>
</div>
```

```
<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Specification</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productSpecification" id="textarea2" required="" rows="3"></textarea>
  </div>
</div>
```

```
<div class="control-group">
  <label class="control-label">Publication Status</label>
  <div class="controls">
    <label class="checkbox inline">
```

```

        <input type="checkbox" id="inlineCheckbox1" name="productStat" required=""
value="1">
        </label>
    </div>
</div>
<div class="form-actions">
    <button type="submit" class="btn btn-primary">Add Product</button>
    <button type="reset" class="btn">Cancel</button>
</div>
</fieldset>
</form>

</div>
</div>
<!--/span-->

</div>
<!--/row-->

```

```
@endsection
```

```
@extends('adminLayout')
```

```
@section('allCategory')
```

```
<div class="row-fluid sortable">
```

```

<div class="box span12">
  <span style="color:lightcoral;">
    @if ($message = Session::get('msg'))
      {{ $message }}
      {{ Session::put('msg',null) }}

    @else

    @endif
  </span>
<div class="box-header" data-original-title>
  <h2><i class="halflings-icon user"></i><span class="break"></span>All Category</h2>
  <div class="box-icon">
    <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
    <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
    <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
  </div>
</div>
<div class="box-content">
  <table class="table table-striped table-bordered bootstrap-datatable datatable">
    <thead>
      <tr>
        <th>Category ID</th>
        <th>Category Name</th>
        <th>Status</th>
        <th>Actions</th>
      </tr>
    </thead>
    @foreach($allCategory as $vAllCategory)
      <tbody>

```



```

<tr>
  <td>{{ $vAllCategory->category_id }}</td>
  <td class="center">{{ $vAllCategory->category_name }}</td>

  @if($vAllCategory->category_status == 1)
  <td class="center">
    <span class="label label-success">Active</span>
  </td>

  @else
  <td class="center">
    <span class="label label-success" style="color:red">Deactivate</span>
  </td>
  @endif

  <td class="center">
    @if($vAllCategory->category_status == 1)
    <a class="btn btn-success" href="{{ URL::To('/deactivecategory/'.$vAllCategory-
>category_id) }}">
      <i class="halflings-icon white thumbs-up"></i>
    </a>

    @else
    <a class="btn btn-success" href="{{ URL::To('/activecategory/'.$vAllCategory-
>category_id) }}">
      <i class="halflings-icon white thumbs-down"></i>
    </a>
    @endif

    <a class="btn btn-info" href="{{ URL::to('/editcategory/'.$vAllCategory->category_id) }}">
      <i class="halflings-icon white edit"></i>

```

```

        </a>
        <a class="btn btn-danger" href="{ URL::to('/deletecategory/'. $vAllCategory-
>category_id) }}">
            <i class="halflings-icon white trash"></i>
        </a>
    </td>
</tr>
</tbody>
@endforeach
</table>
</div>
</div>
<!--/span-->

```

```

</div>

```

```

@endsection

```

All product .php

```

@extends('adminLayout')

```

```

@section('allProducts')

```

```

<div class="row-fluid sortable">

```

```

    <div class="box span12">

```

```

        <span style="color:lightcoral;">

```

```

            @if ($message = Session::get('msg'))

```

```

{{ $message }}
{{ Session::put('msg',null) }}

@else

@endif

</span>

<div class="box-header" data-original-title>
  <h2><i class="halflings-icon user"></i><span class="break"></span>All Products</h2>
  <div class="box-icon">
    <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
    <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
    <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
  </div>
</div>
<div class="box-content">
  <table class="table table-striped table-bordered bootstrap-datatable datatable">
    <thead>
      <tr>
        <th>Product ID</th>
        <th>Product Name</th>
        <th>Product Category</th>
        <th>Product Price</th>
        <th>Product Image</th>
        <th>Product Color</th>
        <th>Product Description</th>
        <th>Product Overview</th>
        <th>Product Spacification</th>
        <th>Status</th>
      </tr>
    </thead>
  </table>
</div>

```

```

        <th>Actions</th>
    </tr>
</thead>
@foreach($allProduct as $vProduct)
<tbody>
    <tr>
        <td>{{ $vProduct->product_id }}</td>
        <td class="center">{{ $vProduct->product_name }}</td>
        <td>{{ $vProduct->category_name }}</td>
        <td>{{ $vProduct->product_price }}</td>
        <td></td>
        <td>{{ $vProduct->product_color }}</td>
        <td>{{ $vProduct->product_description }}</td>
        <td>{{ $vProduct->product_overview }}</td>
        <td>{{ $vProduct->product_specification }}</td>
        <td class="center">
            @if($vProduct->product_status == 1)
                <span class="label label-success">Active</span>
            @else
                <span class="label label-danger">Deactivate</span>
            @endif
        </td>
        <td class="center">
            @if($vProduct->product_status ==1)
                <a class="btn btn-success" href="{{ URL::to('/deactivate/'.$vProduct->product_id) }}">
                    <i class="halflings-icon white thumbs-down"></i>
                </a>
            @else
                <a class="btn btn-success" href="{{ URL::to('/activate/'.$vProduct->product_id) }}">

```

```

        <i class="halflings-icon white thumbs-up"></i>
    </a>
@endif
    <a class="btn btn-info" href="{{ URL::to('/editproduct/'. $vProduct->product_id) }}">
        <i class="halflings-icon white edit"></i>
    </a>
    <a class="btn btn-danger" href="{{ URL::to('/deleteproduct/'. $vProduct->product_id)
}}">

        <i class="halflings-icon white trash"></i>
    </a>
</td>
</tr>
</tbody>
@endforeach
</table>
</div>
</div>
<!--/span-->
</div>
<!--/row-->
@endsection

```

updateproduct.php

```

@extends('adminLayout')
@section('updateProduct')
<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon edit"></i><span class="break"></span>Update Product</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
            </div>
        </div>
    </div>
</div>

```

```

        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
</div>
<div class="box-content">
    <form class="form-horizontal" action="{{ URL::to('/updateproduct/'.$productInfo->product_id)
}}" method="post" enctype="multipart/form-data">
        {{ csrf_field() }}
        <fieldset>
            <div class="control-group">
                <label class="control-label" for="typeahead">Product Name</label>
                <div class="controls">
                    <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->
                    <input type="text" class="form-control" id="exampleInputEmail1"
name="productName" value="{{ $productInfo->product_name }}">
                </div>
            </div>
            <div class="control-group">
                <label class="control-label" for="selectError3">Product Category</label>
                <div class="controls">
                    <select id="selectError3" name="productCategory">
                        <option>{{ $productInfo->category_name }}</option>
                        {{ $dropdownCategory = DB::table('tbl_category')
->get()}}
                        @foreach($dropdownCategory as $vCategory)
                            <option value="{{ $vCategory->category_id }}"> {{ $vCategory->category_name
}}</option>
                        @endforeach
                    </select>
                </div>
            </div>
        </div>
    </div>

```

```

<div class="control-group">
  <label class="control-label" for="typeahead">Product Price</label>
  <div class="controls">
    <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->
    <input type="text" class="form-control" id="exampleInputEmail1" name="productPrice"
value="{{ $productInfo->product_price }}">
  </div>
</div>

<div class="control-group">
  <label class="control-label" for="typeahead">Product Color</label>
  <div class="controls">
    <!-- <input type="text" class="span6 typeahead" id="typeahead" data-
provide="typeahead"> -->
    <input type="text" class="form-control" id="exampleInputEmail1" name="productColor"
value="{{ $productInfo->product_color }}">
  </div>
</div>

<div class="control-group">
  <label class="control-label">Image Upload</label>
  <div class="controls">
    <input type="file" name="productImage">
  </div>
</div>

<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Description</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productDescription" id="textarea2" rows="3"> {{ ($productInfo->product_description) }}
</textarea>
  </div>
</div>

```

```

<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Overview</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productOverview" id="textarea2" rows="3">{{ $productInfo->product_overview }}</textarea>
  </div>
</div>

<div class="control-group hidden-phone">
  <label class="control-label" for="textarea2">Products Specification</label>
  <div class="controls">
    <textarea class="form-control" style="width: 909px;height: 145px;"
name="productSpecification" id="textarea2" rows="3">{{ $productInfo->product_specification
}}</textarea>
  </div>
</div>

<div class="control-group">
  <label class="control-label">Publication Status</label>
  <div class="controls">
    <label class="checkbox inline">
      <input type="checkbox" id="inlineCheckbox1" name="productStat" value=1>
    </label>
  </div>
</div>

<div class="form-actions">
  <button type="submit" class="btn btn-primary">Save changes</button>
  <button type="reset" class="btn">Cancel</button>
</div>
</fieldset>
</form>
</div>
</div>

```



```
<!--/span-->
</div>
<!--/row-->
@endsection
Dashboard.php
@extends('adminLayout')
@section('adminContent')
<div class="login-head">
    <h4 style="color:black;">Welcome to bestprice.com dashboard</h4>
</div>
@endsection
```