

E-COMMERCE PRICE COMPARISON TOOL

BY

AMRITESH SHRIVASTAVA: MCA/40034/20

ARNAB MAHATO: MCA/40039/20

PRATYUSH RANJAN: MCA/40008/20

SHIVAM KUMAR SINGH: MCA/40036/20



**BIRLA INSTITUTE OF TECHNOLOGY MESRA,
LALPUR CAMPUS-834001**

2021

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Professor Halweswar Prasad as well as our Co-ordinator Amrita Priyam who gave us the golden opportunity to do this wonderful project on the topic 'E-commerce price comparison tool', which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful to them.

We would also like to thank our parents and friends who helped us a lot in finishing this project within limited time.

ABSTRACT

E-Commerce Product Price Comparison Tool project proposed here gathers information on product prices from various websites (currently Amazon and Flipkart) & presents it to the users. The users can then choose to buy from the best options available. Even Ecommerce traders can use this price comparison website to study their competitors and form new strategies accordingly to attract new customers & stay ahead of their competitors.

CONTENTS

Overview.....	5
Minimum System Requirement.....	5
Methodology.....	5
Tkinter.....	7
Modules in Python.....	8
The Import Statement.....	8
Writing Modules.....	8
Source Code.....	11
GUI and Other Screen Shot.....	25

➤ **Overview:**

E-Commerce Product Price Comparison Tool project is used to gathers information on product prices from various websites & presents it to the users. This Price comparison tool is extremely helpful for frequent online shoppers to check prices on different online stores in one place, This system will show you the product prices from different retailers to show you where to buy the product at affordable price, Any two static websites classes are analysed to get the pricing details, To get the pricing details, the system visits the website based on user's search and downloads the html search page of that specific website, Once prices from both the websites are retrieved, it is displayed on our website in the form of price comparison.

The tool is created by using Python and its different libraries, packages as its backend, also on other side using Tinker as its frontend using its capabilities to display and logging of data.

➤ **Minimum System Requirement:**

- Processor: i3 3rd gen
- Ram: 4 Gb
- Windows 7 or higher
- 10 Gb storage

➤ **Methodology:**

This project uses the following tools and methods:

- **Python:**

It is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

- **Selenium package:**

Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way. Selenium Python bindings provide a convenient API to access Selenium Web Drivers like Firefox, Ie, Chrome, Remote etc. The current supported Python versions are 3.5 and above.

- **Tkinter:**

Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

- **Web scraping:**

It is the process of using bots to extract content and data from a website. Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML code and, with it, data stored in a database. The scraper can then replicate entire website content elsewhere. Web scraping is used in a variety of digital businesses that rely on data harvesting. Legitimate use cases include:

- a) Search engine bots crawling a site, analysing its content and then ranking it.
- b) Price comparison sites deploying bots to auto-fetch prices and product descriptions for allied seller websites.
- c.) Market research companies using scrapers to pull data from forums and social media (e.g., for sentiment analysis).

- **Data Logging:**

Data logging is the process of collecting and storing data over a period of time in order to analyse specific trends or record the data-based events/actions of a system, network or IT environment. It enables the

tracking of all interactions through which data, files or applications are stored, accessed or modified on a storage device or application.

➤ **TKINTER:**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter is the Python interface to Tk, which is the GUI toolkit for Tcl/Tk.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user
- Tk is an open-source, cross-platform widget toolkit used by many different programming languages to build GUI programs.
- Python implements the Tkinter as a module. Tkinter is a wrapper of C extensions that use Tcl/Tk libraries.
- Tkinter allows you to develop desktop applications. It's a very good tool for GUI programming in Python.

Tkinter is a good choice because of the following reasons:

- Easy to learn.
- Use very little code to make a functional desktop application.
- Layered design.
- Portable across all operating systems including Windows, macOS, and
- Pre-installed with the standard Python library.

➤ **Modules in Python:**

Modules enable you to split parts of your program in different files for easier maintenance and better performance. a module is a file consisting of Python code. It can define functions, classes, and variables, and can also include runnable code. Any Python file can be referenced as a module. A file containing Python code, for example: test.py, is called a module, and its name would be test.

➤ **THE IMPORT STATEMENT**

When interpreter comes across an **IMPORT** statement, it imports the module to your current program. You can use the functions inside a module by using a dot(.) operator along with the module name When interpreter comes across an **IMPORT** statement, it imports the module to your current program. You can use the functions inside a module by using a dot(.) operator along with the module name.

```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from urllib.request import urlopen
import requests
```

Writing Modules

When the interpreter came across the import statement, it imported the module in your code and then by using the dot operator, you were able to access the add () function.

- **Python sleep:**

- The sleep () function suspends (waits) execution of the current thread for a given number of seconds. The sleep()function suspends execution of the current thread for a given number of

seconds. In case of single-threaded programs, `sleep ()` suspends execution of the thread and process. However, the function suspends a thread rather than the whole process in multithreaded programs.

- **Requests:** Requests is a simple, yet elegant, HTTP library.
 - Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your PUT & POST data.
- **Regular Expressions:** A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.
 - The Python module `re` provides full support for Perl-like regular expressions in Python. The `re` module raises the exception `re.error` if an error occurs while compiling or using a regular expression.
- **Python Packages Used:**
 - **Pandas:** Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data.
 - **NumPy:** NumPy is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.
 - **Beautifulsoup:** BeautifulSoup is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. BeautifulSoup supports the HTML parser included in Python's standard library, but it also supports a number of third-party Python parsers. One is the `lxml` parser.
 - **Urlopen:** The `urllib.request` module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more. It uses the `Urlopen` function and is able to fetch URLs using variety of different protocols. `Urllib` is a package that collects several modules for working with URLs.

urllib.request for opening and reading.

urllib.parse for parsing URLs

urllib.error for the exceptions raised

urllib. robotparser for parsing robot.txt files

- **OS:** OS provides a portable way of using operating system dependent functionality. The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The `*os*` and `*os.path*` modules include many functions to interact with the file system.
- **Time:** Python has a module named time to handle time-related tasks. To use functions defined in the module, we need to import the module first the time () function returns the number of seconds passed.
- **Smtp:** Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers. Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Source Code:

```
import tkinter as tk          # import the tkinter module as tk to the program
import time
import datetime
import smtplib

root = tk.Tk()                #the main windows is called root

#root window configuring
root.title("E-commerce price comparison tool")          #Gives the title of the
window
root.geometry("1920x1080")          #Gives the dimensions of
window
root.configure(bg="#6495ED")        #setting window color

#Individual product label
iplabel = tk.Label(
    root,
    text=" Individual products ",
    bg='white',
    fg='green'
)
iplabel.place(x=80, y=50)

#Individual output label
olabel = tk.Label(
    root,
```

```
text="      Output      ",
bg='white',
fg='green'
)
olabel.place(x=80, y=300)
```

```
#Individual part labels
```

```
# Amazon label
```

```
alabel = tk.Label(
    root,
    text=" Amazon Link ",
    bg='white',
    fg='red'
)
alabel.place(x=30, y=100)
```

```
# Flipkart label
```

```
flabel = tk.Label(
    root,
    text=" Flipkart Link ",
    bg='white',
    fg='blue'
)
flabel.place(x=30, y=150)
```

```
#Individual Output part labels
```

```
#Amazon label
```

```
flabel = tk.Label(  
    root,  
    text=" Amazon ",  
    bg='white',  
    fg='red'  
)  
flabel.place(x=30, y=350)
```

```
# Flipkart label
```

```
flabel = tk.Label(  
    root,  
    text=" Flipkart ",  
    bg='white',  
    fg='blue'  
)  
flabel.place(x=30, y=550)
```

```
#Creating textbox
```

```
amazonValue = tk.StringVar()  
flipkartValue = tk.StringVar()
```

```
#Taking string value
```

```
amazonEntry = tk.Entry(root, textvariable = amazonValue).place(x=150,y=100)
flipkartEntry = tk.Entry(root, textvariable = flipkartValue).place(x=150,y=150)
```

```
import requests
from bs4 import BeautifulSoup as bs
```

```
#Amazon code
import re
```

```
def get_converted_price(price):
    converted_price = float(re.sub(r"^\d.", "", price))
    return converted_price
```

```
def extract_url(url):
```

```
    if url.find("www.amazon.in") != -1:
```

```
        index = url.find("/dp/")
```

```
        if index != -1:
```

```
            index2 = index + 14
```

```
            url = "https://www.amazon.in" + url[index:index2]
```

```
else:
    index = url.find("/gp/")
    if index != -1:
        index2 = index + 22
        url = "https://www.amazon.in" + url[index:index2]
    else:
        url = None
else:
    url = None
return url
```

```
address_field = tk.Label(text="Enter your email address ").place(x=370,y=50)
address = tk.StringVar()
address_entry = tk.Entry(textvariable=address).place(x=370,y=100)
```

```
#Input desired price
```

```
#Label
```

```
adesire_field = tk.Label(text=" Enter your desire price ").place(x=580,y=50)
```

```
#Amazon
```

```
adesire_price = tk.StringVar()
```

```
adesire_entry = tk.Entry(textvariable=adesire_price).place(x=580,y=100)
```

```
#Flipkart
```

```
fdesire_price = tk.StringVar()
```

```
fdesire_entry = tk.Entry(textvariable=fdesire_price).place(x=580,y=150)
```

```

def agetValue():

    url = amazonValue.get()

    headers = {
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:68.0)
        Gecko/20100101 Firefox/68.0"
    }
    details = {"name": "", "price": 0, "deal": True, "url": ""}
    _url = extract_url(url)
    if _url is None:
        details = None
    else:
        page = requests.get(_url, headers=headers)
        soup = bs(page.content, "html.parser")
        title = soup.find(id="productTitle")
        price = soup.find(id="priceblock_dealprice")

        if price is None:
            price = soup.find(id="priceblock_ourprice")

        details["deal"] = False

    if title is not None and price is not None:
        details["name"] = title.get_text().strip()
        details["price"] = get_converted_price(price.get_text())

```



```

        details["price"] = int(float(details["price"]))
#converting float price to int
        details["url"] = _url

    else:
        details = None
ct = datetime.datetime.now()
converted_ct = str(ct)

#Making Amazon.txt and storing details
with open('Amazon.txt', 'a+') as a:
    a.write('Product Name = '+details["name"])
    a.write("\n\n")
    a.write("Price = "+str(details["price"]))
    a.write("\n")
    a.write("Time = '+converted_ct)
    a.write("\n\n")
    a.close()

#Amazon output labels
allabel = tk.Label(
root,
text=" Product name ",
bg='white',
fg='blue'
)
allabel.place(x=30, y=400)

```

```
alabelpn = tk.Label(  
    root,  
    text=details["name"],  
    bg='white',  
    fg='blue'  
)  
alabelpn.place(x=150, y=400)
```

```
#label for price  
a1label = tk.Label(  
    root,  
    text="    Price    ",  
    bg='white',  
    fg='blue'  
)  
a1label.place(x=30, y=430)
```

```
labelpp = tk.Label(  
    root,  
    text=details["price"],  
    bg='white',  
    fg='blue'  
)  
labelpp.place(x=150, y=430)  
print("\nAmazon details:\n\n"+"Product: "+details["name"]+"\n\nPrice: ")
```

```
print(details["price"])
```

```
if address.get() != "":
```

```
    aprice = int(details["price"])
```

```
    adesire_int = int(adesire_price.get())
```

```
    if(aprice <= adesire_int):
```

```
        address_info = address.get()
```

```
        email_body = "Amazon - Your desired value has been reached."
```

```
        print(address_info)
```

```
        sender_email = "arnabmahato91@gmail.com"
```

```
        sender_password = "Excitedsky1!"
```

```
        server = smtplib.SMTP('smtp.gmail.com',587)
```

```
        server.starttls()
```

```
        server.login(sender_email,sender_password)
```

```
        print("Login successful")
```

```
        server.sendmail(sender_email,address_info,email_body)
```

```

        print("Message sent")

    else:
        print("sab khatam")

    return details["price"]
#Flipkart code
def fgetValue():

    url = flipkartValue.get() #To get the current text of a
    #Entry widget as a string, we use the get() method

    request = requests.get(url)
    soup = bs(request.content, 'html.parser')

    product_name = soup.find("span", {"class": "B_NuCI"}).get_text()
    price = soup.find("div", {"class": "_30jeq3 _16Jk6d"}).get_text()
    price = price.replace("₹", "")
    price = price.replace(",", "")

    ct = datetime.datetime.now()
    converted_ct = str(ct)

    #Making flipkart.txt and storing details
    with open('flipkart.txt', 'a+') as f:
        f.write('Product Name = '+product_name)

```

```
f.write('\n\n')
f.write('Price = '+price)
f.write('\n')
f.write('Time = '+converted_ct)
f.write('\n\n')
f.close()
```

#labels for product name

```
f1label = tk.Label(root,
text=" Product name ",
bg='white',
fg='blue')
f1label.place(x=30, y=600)
```

```
labelpn = tk.Label(root,
text=product_name,
bg='white',
fg='blue')
labelpn.place(x=150, y=600)
```

#label for price

```
f1label = tk.Label(root,
text="      Price      ",
bg='white',
fg='blue'
```

```
)
```

```
f1label.place(x=30, y=630)
```

```
labelpp = tk.Label(  
root,  
text=price,  
bg='white',  
fg='blue'  
)  
labelpp.place(x=150, y=630)
```

```
print("\n\nFlipkart details:\n\n"+"Product: "+product_name+"\n\nPrice: ")  
print(price)
```

```
if address.get() != "":  
    print(fdesire_price)  
    fprice = int(price)  
    fdesire_store = fdesire_price.get()  
    fdesire_int = int(fdesire_store)
```

```
    if(fprice <= fdesire_int):  
#if desired_price becomes equal to or greater than flipkart price
```

```
address_info = address.get()  
    email_body = "Flipkart - Your desired value has been reached."
```

```
print(address_info)

sender_email = "arnabmahato91@gmail.com"

sender_password = "Excitedsky1!"

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login(sender_email,sender_password)

print("Login successful")

server.sendmail(sender_email,address_info,email_body)

print("Message sent")

else:
    print("sab khatam")
return price
```

```
def fauto():#Flipkart Automation
```

```
    while True:
```

```
        time.sleep(10)
```

```
        f = fgetValue()#line 252
```

```
def aauto(): #Amazon Automation
```

```
    while True:
```

```
        time.sleep(10)
```

```
        a = agetValue()#line 137
```

```
#Automate button
```

```
tk.Button(text="Automate", command=fauto).place(x=760,y=145)
```

```
tk.Button(text="Automate", command=aauto).place(x=760,y=95)
```

```
def compare():#One time price check function
```

```
    fgetValue()#Line 252
```

```
    agetValue()#Line 137
```

```
tk.Button(text="Compare", command=compare).place(x=120,y=250)
```

```
root.mainloop()          #mainloop() keeps the window visible on the screen.
```

If you don't call the mainloop() method, the windows will disappear

Individual products

Amazon Link

Flipkart Link

Enter your email address

Enter your desire price

Automate

Open log folder

Automate

Compare

Output

Amazon

Flipkart

Home Screen

Individual products

Amazon Link

s=electronics&sr=1-4

Flipkart Link

&cid=VTD69CFDGGP2

Enter your email address

example@test.com

Enter your desire price

30000

Automate

Open log folder

Automate

Compare

Output

Amazon

Product name

HP Chromebook 14-inch (35.56 cms) Thin & Light Touchscreen Laptop (Celeron N4020/4GB/64GB eMMC + 256GB Expandable Storage/Chrome OS/1.46 kgs Light/ Mineral Silver), 14a-na0003TU

Price

26990

Flipkart

Product name

Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price

25999

Compared Prices

Flipkart details:

Product: Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price:
25999

Amazon details:

Product: HP Chromebook 14-inch (35.56 cms) Thin & Light Touchscreen Laptop (Celeron N4020/4GB/64GB eMMC + 256GB Expandable Storage/Chrome OS/1.46 kgs Light/ Mineral Silver), 14a-na0003TU

Price:
26990

Amazon details:

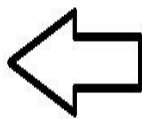
Product: HP Chromebook 14-inch (35.56 cms) Thin & Light Touchscreen Laptop (Celeron N4020/4GB/64GB eMMC + 256GB Expandable Storage/Chrome OS/1.46 kgs Light/ Mineral Silver), 14a-na0003TU

Price:
26990

pratyushrn3019@gmail.com


Login successful

Message sent



USES SMTP TO SEND NOTIFICATION TO THE USER WHEN THE DESIRED PRICE IS REACHED

SMTP for notification

(no subject)  Inbox x



[REDACTED]


to ▾

Amazon - Your desired value has been reached.

 Reply

 Forward

Email Received for Amazon's notification

(no subject)  Inbox x



[REDACTED]

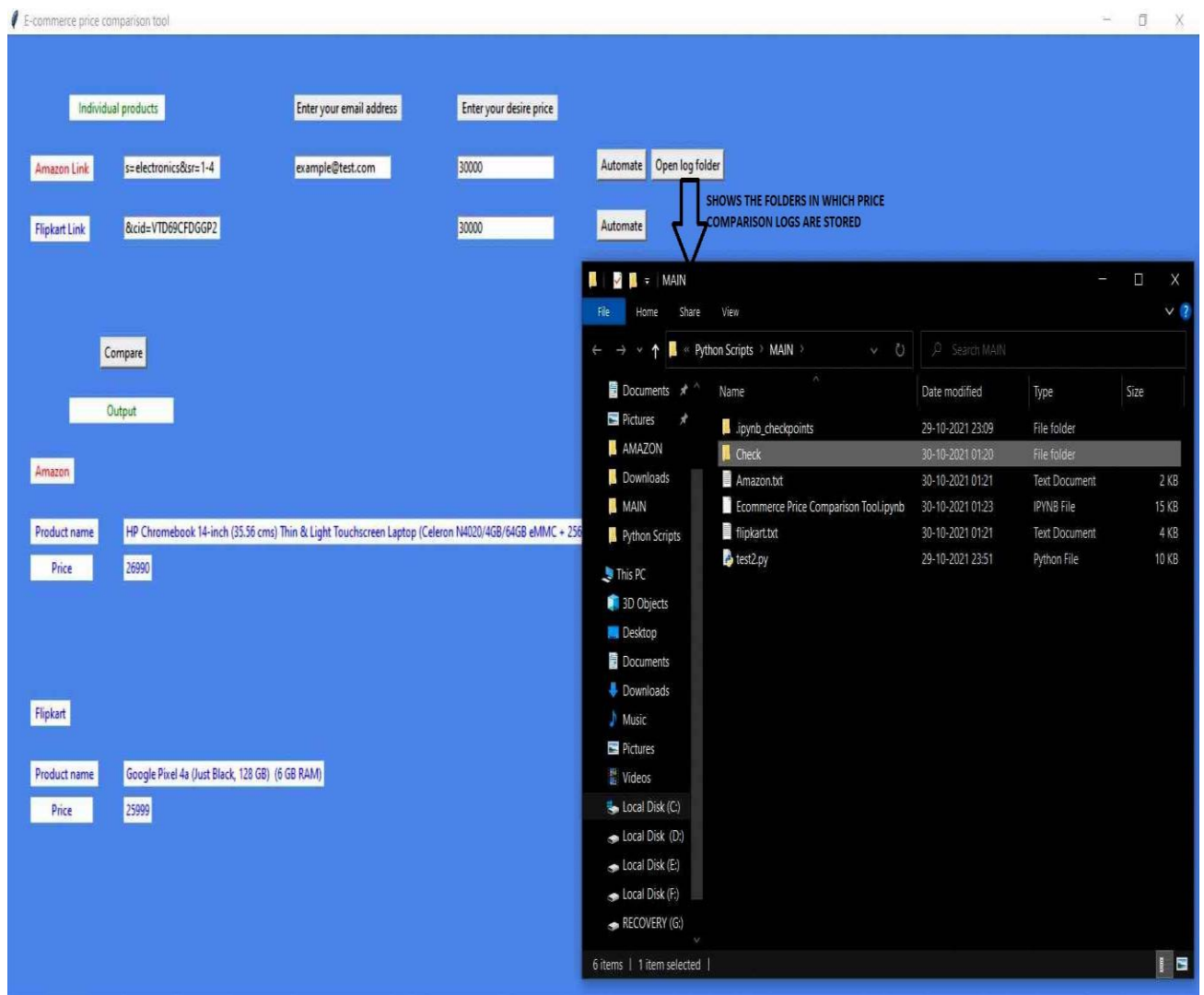
to ▾

Flipkart - Your desired value has been reached.

 Reply

 Forward

Email Received for Flipkart's notification



Opens Log Folder

```
Amazon.txt - Notepad
File Edit Format View Help
Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:11:42.054312

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:12:32.600954

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:12:48.853922

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:26:51.048970

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:28:25.220477

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:28:41.474882

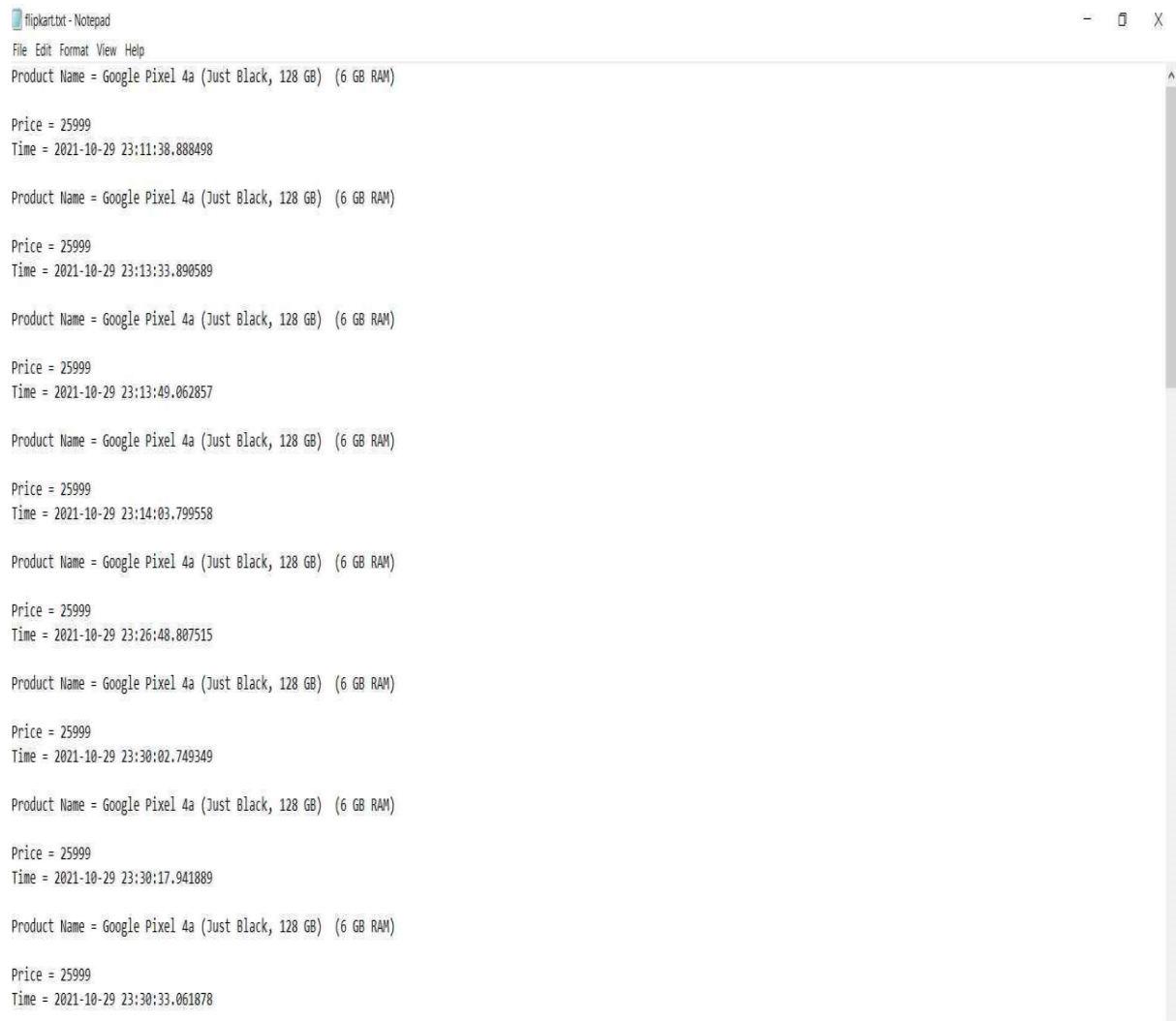
Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:41:00.504824

Product Name = OnePlus Nord CE 5G (Blue Void, 8GB RAM, 128GB Storage)

Price = 24999
Time = 2021-10-29 23:44:06.177298
```

Log file for amazon



```
flipkart.txt - Notepad
File Edit Format View Help
Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:11:38.888498

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:13:33.890589

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:13:49.062857

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:14:03.799558

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:26:48.807515

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:30:02.749349

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:30:17.941889

Product Name = Google Pixel 4a (Just Black, 128 GB) (6 GB RAM)

Price = 25999
Time = 2021-10-29 23:30:33.061878
```

Log file for flipkart