Abstract

The project "Price Comparison using Web Scraping and Data Analysis" aims to develop a systematic approach for comparing prices of products across multiple e-commerce websites. With the increasing number of online retailers and the dynamic nature of product pricing, consumers often struggle to find the best deals. This project addresses this issue by leveraging web scraping techniques to gather pricing data from various websites and employing data analysis methods to compare and analyze the collected data.

The project begins with a comprehensive literature review, exploring the significance of price comparison and existing approaches in the field. It also covers the fundamentals of web scraping and data analysis, providing an overview of relevant techniques and tools. Building upon this foundation, a methodology is devised, outlining the steps involved in data collection, preprocessing, and analysis.

The implementation phase focuses on developing a software architecture that supports efficient web scraping and data analysis. Various programming languages and frameworks are utilized to construct the scraping algorithms and perform data preprocessing. Challenges encountered during implementation are addressed, and solutions are proposed to overcome website-specific restrictions and handle missing or incomplete data.

The results obtained from the project are presented and discussed in detail. The comparison of prices across different websites and products is illustrated using informative graphs and charts. Key findings and insights are highlighted, and the accuracy and reliability of the results are validated. Additionally, the project acknowledges its limitations and suggests potential areas for improvement and future work.

The project's contributions lie in providing a practical and automated solution for price comparison, empowering consumers to make informed purchasing decisions. By leveraging web scraping and data analysis techniques, this project demonstrates the effectiveness of extracting and analyzing pricing data from multiple sources. Overall, it serves as a valuable resource for individuals seeking to optimize their shopping experience and obtain the best possible prices for desired products.

I. Introduction

A. Background and Motivation:

- Explain the increasing popularity of online shopping and the vast number of e-commerce websites available.
- Discuss the importance of price comparison for consumers to make informed purchasing decisions and obtain the best deals.
- Highlight the challenges faced by consumers in manually searching and comparing prices across multiple platforms.

B. Problem Statement:

- Clearly state the problem addressed in the project: the lack of an efficient and automated mechanism for comparing prices across different e-commerce websites.
- Emphasize the need for a solution that saves time and effort for consumers and enables them to find the most cost-effective options.

C. Objectives and Scope of the Project:

- Present the main objectives of the project, such as developing a price comparison tool using web scraping and data analysis.
- Discuss the specific goals, such as collecting price data from multiple websites, performing data analysis and comparison, and presenting the results to users.
- Define the scope of the project, including the specific e-commerce websites considered and the types of products covered.

D. Research Questions:

- **List the research questions that the project aims to answer, such as:**
    1. How can web scraping be utilized to collect price data from various e-commerce websites?
    2. What data analysis techniques are suitable for comparing prices and identifying the best deals?
    3. How accurate and reliable are the results obtained through web scraping and data analysis?

E. Methodology Overview:

- Provide a brief overview of the methodology adopted in the project, including the steps involved in web scraping, data preprocessing, analysis, and visualization.
- Mention the programming languages and frameworks used in the implementation.
- Highlight the importance of ethical considerations, such as adhering to website terms of service and respecting data privacy.

II. System Requirements

A. Hardware Requirements

- **Processor:** The system should be equipped with a processor that meets the computational demands of web scraping and data analysis tasks. A multicore processor with a clock speed of at least 2 GHz or higher is recommended to handle concurrent scraping and analysis operations efficiently.

- **Memory (RAM):** Sufficient RAM is crucial for handling the data processing and analysis operations effectively. The amount of memory required depends on the size of the dataset being processed and the complexity of the analysis. As a general guideline, a minimum of 8 GB of RAM is recommended for small to medium-sized datasets. However, for larger datasets or complex analysis tasks, 16 GB or more may be necessary to ensure optimal performance.

- **Storage:** Adequate storage space is necessary to store the system components, web scraping scripts, and any collected data. The required storage capacity will depend on the size of the dataset and the frequency of data updates. It is recommended to have at least 100 GB of free disk space to accommodate the system and data storage needs.

- **Internet Connection:** A stable and reliable internet connection is essential for accessing the e-commerce websites and retrieving the required information through web scraping. A broadband connection with reasonable upload and download speeds is recommended to ensure efficient data retrieval.

B. Software Requirements

- **Operating System:** The system should be compatible with the chosen operating system. Common options include Windows, macOS, or Linux distributions. The specific version of the operating system should support the required software dependencies and libraries.

- **Python:** The price comparison system is developed using Python programming language. Therefore, Python needs to be installed on the system. It is recommended to use the latest stable version of Python (e.g., Python 3.8 or higher). The required Python packages, such as BeautifulSoup and requests for web scraping, should be installed using a package manager like pip.

- **Integrated Development Environment (IDE):** An Integrated Development Environment (IDE) can greatly enhance the development process. Popular choices include PyCharm, Visual Studio Code, or Jupyter Notebook, which provide features like code editing, debugging, and version control integration. The chosen IDE should be compatible with the operating system.

- **Web Browsers:** The price comparison system may rely on web browsers to display the graphical user interface (GUI) or open product links for further analysis. Commonly used web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge should be

installed on the system.

Web Browser Automation Tools (optional): If web scraping involves interacting with web pages dynamically, web browser automation tools like Selenium WebDriver may be required. These tools enable the system to automate interactions with web elements, such as clicking buttons or filling forms, which may be necessary to retrieve the desired data.

Database Management System (if applicable): If the system incorporates a database for storing and retrieving data, a compatible Database Management System (DBMS) should be installed. Popular choices include MySQL, PostgreSQL, or MongoDB. The chosen DBMS should be compatible with the operating system and provide the necessary features for efficient data storage and retrieval.

III. Technologies

In the "Price Comparison using Web Scraping and Data Analysis" project, several technologies are employed to accomplish the desired functionality. These technologies play a crucial role in different aspects of the project, including data retrieval, web scraping, data analysis, and user interface development. The chosen technologies are outlined below:

- **Programming Language:** The project is implemented using Python programming language. Python is a versatile language widely used in web development, data analysis, and automation tasks. It provides a rich set of libraries and frameworks that facilitate web scraping, data manipulation, and statistical analysis.

- **Web Scraping:** Web scraping is a key component of the project, enabling data extraction from various e-commerce websites. The following libraries are utilized for web scraping: a. Requests: A popular library for making HTTP requests to retrieve web pages' content. b. BeautifulSoup: A powerful library for parsing HTML and XML documents, allowing easy navigation and extraction of relevant data.

- **GUI Development:** Graphical User Interface (GUI) is an essential part of the project to provide a user-friendly interface for interaction. The following technology is used for GUI development: a. Tkinter: Tkinter is a standard GUI toolkit for Python that provides a set of components and functions to create windows, buttons, labels, and other graphical elements.

- **Data Analysis:** Data analysis is a crucial aspect of the project to derive insights and make comparisons. The following libraries are utilized for data analysis: a. NumPy: A fundamental library for scientific computing with Python, providing support for large, multi-dimensional arrays and mathematical functions. b. Pandas: A powerful library for data manipulation and analysis, offering data structures and functions for efficient data handling and cleaning. c. Matplotlib: A widely used plotting library in Python for creating visualizations such as graphs, histograms, and scatter plots.

- **Web Browser Interaction:** Interacting with web browsers is necessary for opening product links. The following technology is used for this purpose: a. webbrowser: A Python library that provides a simple interface for opening web pages in the user's default web browser.

- **Integrated Development Environment (IDE):** An IDE is used for coding and development. The choice of IDE depends on personal preference, but popular options include PyCharm, Visual Studio Code, or Jupyter Notebook.

The selection of these technologies is based on their compatibility, ease of use, availability of libraries for web scraping and data analysis, and the ability to meet the project requirements efficiently. It is important to note that the choice of technologies may vary depending on individual preferences and project specifications.

IV. Architecture of Project

The architecture of the project provides an overview of the system's structure and how its components interact with each other. It outlines the high-level design and organization of the system, highlighting the key modules and their relationships. The architecture ensures that the project is scalable, maintainable, and efficient in achieving its objectives.

The architecture of the "Price Comparison using Web Scraping and Data Analysis" project can be divided into several components:

1. User Interface:

The user interface component is responsible for providing an interactive and intuitive interface for users to input product names, initiate the price comparison process, and view the results. It interacts with other modules to gather user inputs and display the outputs.

2. Web Scraping Modules:

These modules are responsible for extracting data from various e-commerce websites. The project utilizes web scraping techniques to collect information about product names and prices from websites such as Flipkart, GadgetsNow, and compareraja.in. Each website has its specific web scraping algorithm to navigate the site's structure, send requests, and extract relevant data.

3. Data Processing and Analysis:

This component involves processing the scraped data and performing data analysis to compare prices across different websites and products. The extracted data is preprocessed to remove noise, normalize the format, and handle missing or incomplete information. Statistical analysis techniques are applied to identify trends, patterns, and calculate price differences and percentage variances. The results are then presented to the user.

4. Visualization:

The visualization component focuses on representing the collected data and analysis results in a visually appealing and understandable manner. It generates informative graphs, charts, or visual representations to facilitate better comprehension of the price comparison outcomes. The visualizations aid in comparing prices, identifying the cheapest option, and visualizing trends and insights.

5. Database (if applicable):

If the project incorporates a database, it serves as a storage mechanism for storing the collected data or any additional information related to the products and prices. The database architecture includes data schema design, table creation, and data storage and retrieval mechanisms.

The interaction between these components forms the overall architecture of the project. The user interface interacts with the web scraping modules to collect data from e-commerce websites. The extracted data is then processed, analyzed, and visualized to provide meaningful insights to the user. If a database is included, it serves as a storage mechanism for data persistence.

The architecture ensures a modular and scalable design, allowing for easy addition or modification of modules and functionalities. It promotes code reusability, separation of concerns, and efficient communication between the different components.

By following a well-defined architecture, the "Price Comparison using Web Scraping and Data Analysis" project can achieve its objectives effectively, providing users with accurate and valuable information for comparing prices across multiple e-commerce websites.

V. Modules Developed

This section provides a detailed description of each module developed as part of the "Price Comparison using Web Scraping and Data Analysis" project. Each module represents a distinct functionality or component of the system.

- **Web Scraping Module**

    1. Description: This module is responsible for retrieving data from various e-commerce websites. It utilizes web scraping techniques to extract product names and prices.

    2. Functionality: Implements web scraping algorithms for each targeted website (e.g., Flipkart, GadgetsNow, compareraja.in).

    3. Handles website-specific challenges, such as handling pagination, navigating through search results, and dealing with different HTML structures.

    4. Extracts relevant information, such as product names and prices, from the HTML response. Provides methods for searching and retrieving data for a given product.

- **Data Preprocessing Module**

    1. Description: This module focuses on preprocessing and cleaning the scraped data before analysis.

    2. Functionality: Extracts numeric prices from the scraped data using regular expressions and converts them to a numerical format.

    3. Handles missing or incomplete data, applying suitable techniques for data imputation or removal.

    4. Normalizes and standardizes the data, ensuring consistency and comparability across different websites.

- **Data Analysis Module**

    1. Description: This module performs data analysis on the preprocessed data to gain insights and compare prices.

    2. Functionality: Conducts statistical analysis, such as calculating mean, median, and standard deviation of prices.

    3. Identifies trends and patterns in price data, using techniques like time series analysis or regression analysis.

    4. Calculates price differences and percentage variances between different websites and products.

    5. Generates informative visualizations, graphs, and charts to present the analysis results.

- **Graphical User Interface (GUI) Module**

    1. Description: This module provides a user-friendly interface for interacting with the system.

    2. Functionality: Implements a graphical user interface using Tkinter, a Python GUI toolkit.

    3. Allows users to enter product names, initiate price comparison, and view the results.

    4. Displays the results of price comparison from different websites in a clear and organized manner.

    5. Provides options to open the product links in the web browser for further exploration.

- **Database Module (if Applicable)**

    1. Description: This module handles the storage and retrieval of data, if a database is used in the project.

    2. Functionality: Establishes a connection to the database and creates the necessary tables or collections.

    3. Stores the scraped data in the database for future reference or analysis.

    4. Retrieves data from the database for analysis or presentation purposes.

Each module plays a crucial role in the overall functionality of the price comparison system. They work together to collect, preprocess, analyze, and present the data, enabling users to make informed decisions based on the comparison results.

VI. Graphical User Interface & Source Code

In this section, we will discuss the Graphical User Interface (GUI) and present relevant portions of the source code for your "Price Comparison using Web Scraping and Data Analysis" project.

A. **Graphical User Interface (GUI)**

The GUI serves as the user-facing part of your application, allowing users to interact with the system and view the results of the price comparison. It provides a user-friendly and intuitive interface to input product names, display search results, and visualize the compared prices.

The following components are included in the GUI:

```
# Create Tkinter window
window = tk.Tk()
window.title("Price Comparison")
window.geometry("800x600")

# Create search section
search_frame = tk.Frame(window)
search_frame.pack(pady=20)

search_label = tk.Label(search_frame, text="Enter Product Name:")
search_label.grid(row=0, column=0)

search_entry = tk.Entry(search_frame, width=40)
search_entry.grid(row=0, column=1)

search_button = tk.Button(search_frame, text="Search", command=compare_prices)
search_button.grid(row=0, column=2)

# Create Flipkart section
flipkart_frame = tk.Frame(window)
flipkart_frame.pack(pady=20)

flipkart_label = tk.Label(flipkart_frame, text="Flipkart", font=("Helvetica", 16, "bold"))
flipkart_label.pack()

flipkart_result = tk.Label(flipkart_frame, text="", font=("Helvetica", 12))
flipkart_result.pack()

# Create GadgetsNow section
gadgetsnow_frame = tk.Frame(window)
```

```
gadgetsnow_frame.pack(pady=20)

gadgetsnow_label = tk.Label(gadgetsnow_frame, text="GadgetsNow",
font=("Helvetica", 16, "bold"))
gadgetsnow_label.pack()

gadgetsnow_result = tk.Label(gadgetsnow_frame, text="", font=("Helvetica", 12))
gadgetsnow_result.pack()

# Create compareraja section
compareraja_frame = tk.Frame(window)
compareraja_frame.pack(pady=20)

compareraja_label = tk.Label(compareraja_frame, text="compareRaja",
font=("Helvetica", 16, "bold"))
compareraja_label.pack()

compareraja_result = tk.Label(compareraja_frame, text="", font=("Helvetica", 12))
compareraja_result.pack()

# Create cheapest price section
cheapest_frame = tk.Frame(window)
cheapest_frame.pack(pady=20)

cheapest_label = tk.Label(cheapest_frame, text="Cheapest Price", font=("Helvetica",
16, "bold"))
cheapest_label.pack()

cheapest_price = tk.Label(cheapest_frame, text="", font=("Helvetica", 12))
cheapest_price.pack()

# Create Get Product section
get_product_frame = tk.Frame(window)
get_product_frame.pack(pady=20)

get_product_button = tk.Button(get_product_frame, text="Get Product",
command=open_product_link)
get_product_button.pack()
```

```
# Start the Tkinter event loop
window.mainloop()
```

The GUI is built using the Tkinter library in Python. It includes the following components:

- **Search Section:**
    1. A text label prompting the user to enter the product name.
    2. An entry field where the user can input the product name.
    3. A search button to trigger the price comparison process.

- **Result Display Areas:**
    1. Three separate areas for displaying the search results from each website (Flipkart, GadgetsNow, and compareRaja).
    2. Each result area shows the product name and its corresponding price.

- **Cheapest Price Section:**
    1. A section that displays the website with the cheapest price for the searched product.
    2. The website name is displayed as text.

- **Get Product Button:**
    1. A button that allows the user to open the product page of the website with the cheapest price in their web browser.

B. Source Code

1. **Web Scraping Functions:** These code snippets implement web scraping for each e-commerce website by extracting the relevant information (product name and price) from the HTML content.

```
def search_compareraja(product_name):
    search_query = product_name.replace(' ', '-')
    url = f"https://www.compareraja.in/search?c=all&q={search_query}"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser')

    if soup:
        product_name_element = soup.find("a", class_="link")
        product_price_element = soup.find("b")

        if product_name_element and product_price_element:
            product_name = product_name_element.text.strip()
```

```
            product_price = product_price_element.text.strip()
            numeric_price = extract_numeric_price(product_price)
            if numeric_price:
                compareraja_result['text'] = f"Product Name: {product_name}\nProduct
Price: {numeric_price}"
                return numeric_price
            else:
                compareraja_result['text'] = "Product not found."
        else:
            compareraja_result['text'] = "Product not found."
    else:
        compareraja_result['text'] = "Product not found."
    return None


def search_flipkart(product_name):
    search_query = product_name.replace(' ', '%20')
    url =
f"https://www.flipkart.com/search?q={search_query}&otracker=search&otracker1=s
earch&marketplace=FLIPKART&as-show=on&as=off"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser')
    product_name_element = soup.find("div", class_="_4rR01T")
    product_price_element = soup.find("div", class_="_30jeq3 _1_WHN1")

    if product_name_element and product_price_element:
        product_name = product_name_element.text.strip()
        product_price = product_price_element.text.strip()
        numeric_price = extract_numeric_price(product_price)
        if numeric_price:
            flipkart_result['text'] = f"Product Name: {product_name}\nProduct Price:
{numeric_price}"
            return numeric_price
        else:
            flipkart_result['text'] = "Product not found."
    else:
```

```
        flipkart_result['text'] = "Product not found."
    return None


def search_gadgetsnow(product_name):
    search_query = product_name.replace(' ', '+')
    url =
f"https://shop.gadgetsnow.com/mtkeywordsearch?SEARCH_STRING={search_query}"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser')

    if soup:
        product_name_element = soup.find("span", class_="product-name")
        product_price_element = soup.find("span", class_="offerprice")

        if product_name_element and product_price_element:
            product_name = product_name_element.text.strip()
            product_price = product_price_element.text.strip()
            numeric_price = extract_numeric_price(product_price)
            if numeric_price:
                gadgetsnow_result['text'] = f"Product Name: {product_name}\nProduct
Price: {numeric_price}"
                return numeric_price
            else:
                gadgetsnow_result['text'] = "Product not found."
        else:
            gadgetsnow_result['text'] = "Product not found."
    else:
        gadgetsnow_result['text'] = "Product not found."
    return None
```

- **search_flipkart(product_name):** This function performs a search on Flipkart using the provided product name. It scrapes the website's HTML content to find and extract the product name and price elements.

- **search_gadgetsnow(product_name):** This function searches for the product on GadgetsNow. It utilizes web scraping techniques to extract the product name and price from the website's HTML.

- **search_compareraja(product_name):** This function searches for the product on compareRaja. It uses web scraping methods to extract the product name and price information from the HTML structure.

2. **Data Analysis and Comparison:** The compare_prices function collects the price data, performs statistical analysis, and determines the website with the cheapest price.

```
def compare_prices():
    product_name = search_entry.get()
    flipkart_price = search_flipkart(product_name)
    gadgetsnow_price = search_gadgetsnow(product_name)
    compareraja_price = search_compareraja(product_name)

    if flipkart_price and gadgetsnow_price and compareraja_price:
        min_price = min(flipkart_price, gadgetsnow_price, compareraja_price)
        if min_price == flipkart_price:
            cheapest_price['text'] = "Flipkart"
        elif min_price == gadgetsnow_price:
            cheapest_price['text'] = "GadgetsNow"
        else:
            cheapest_price['text'] = "compareRaja"
    else:
        cheapest_price['text'] = "Product not found."
```

The compare_prices() function collects the price data, performs statistical analysis, and determines the website with the cheapest price.

- The function retrieves the product name entered by the user.
- It calls the web scraping functions for each website to obtain the prices.
- The collected prices are compared to find the minimum value.
- The website associated with the cheapest price is displayed in the GUI.

3. **Link Retrieval Functions:** These code snippets generate the links to the product pages on the respective websites.

```
def get_flipkart_product_link(product_name):
    search_query = product_name.replace(' ', '%20')
    url =
f"https://www.flipkart.com/search?q={search_query}&otracker=search&otracker1=s
earch&marketplace=FLIPKART&as-show=on&as=off"
    return url


def get_gadgetsnow_product_link(product_name):
    search_query = product_name.replace(' ', '+')
    url =
f"https://shop.gadgetsnow.com/mtkeywordsearch?SEARCH_STRING={search_query}"
    return url


def get_compareraja_product_link(product_name):
    search_query = product_name.replace(' ', '-')
    url = f"https://www.compareraja.in/search?c=all&q={search_query}"
    return url
```

- **get_flipkart_product_link(product_name):** This function takes the product name as input and generates the URL for the product page on Flipkart.
- **get_gadgetsnow_product_link(product_name):** This function generates the URL for the product page on GadgetsNow based on the provided product name.
- **get_compareraja_product_link(product_name):** This function generates the URL for the product page on compareRaja using the given product name.

The code snippets presented here illustrate the essential parts of the GUI and source code for your project.

VII. Output

A. Presentation of Collected Data:

1. The output will display the product name and its corresponding price for each website queried (Flipkart, GadgetsNow, and compareRaja).

2. The collected data will be presented in a structured format, allowing for easy comparison and analysis.

B. Comparison of Prices:

1. The output will include a detailed comparison of prices for the same product across different e-commerce websites.

2. For each product, the prices obtained from Flipkart, GadgetsNow, and compareRaja will be displayed.

3. The comparison may include the minimum, maximum, and average prices for each product, allowing users to make informed decisions based on their budget and preferences.

4. The output will highlight the website offering the lowest price for each product.

C. Determining the Cheapest Price:

1. The output will indicate the website offering the cheapest price for the searched product.

2. If the prices from Flipkart, GadgetsNow, and compareRaja are available, the output will state which website provides the lowest price.

3. If the product is not found on any of the websites, the output will indicate that the product is not available.

D. Opening Product Links:

1. The output will provide the option to open the product link from the website offering the cheapest price.

2. If the cheapest price is from Flipkart, the output will include a button to open the product link on the Flipkart website.

3. If the cheapest price is from GadgetsNow, the output will include a button to open the product link on the GadgetsNow website.

4. If the cheapest price is from compareRaja, the output will include a button to open the product link on the compareRaja website.

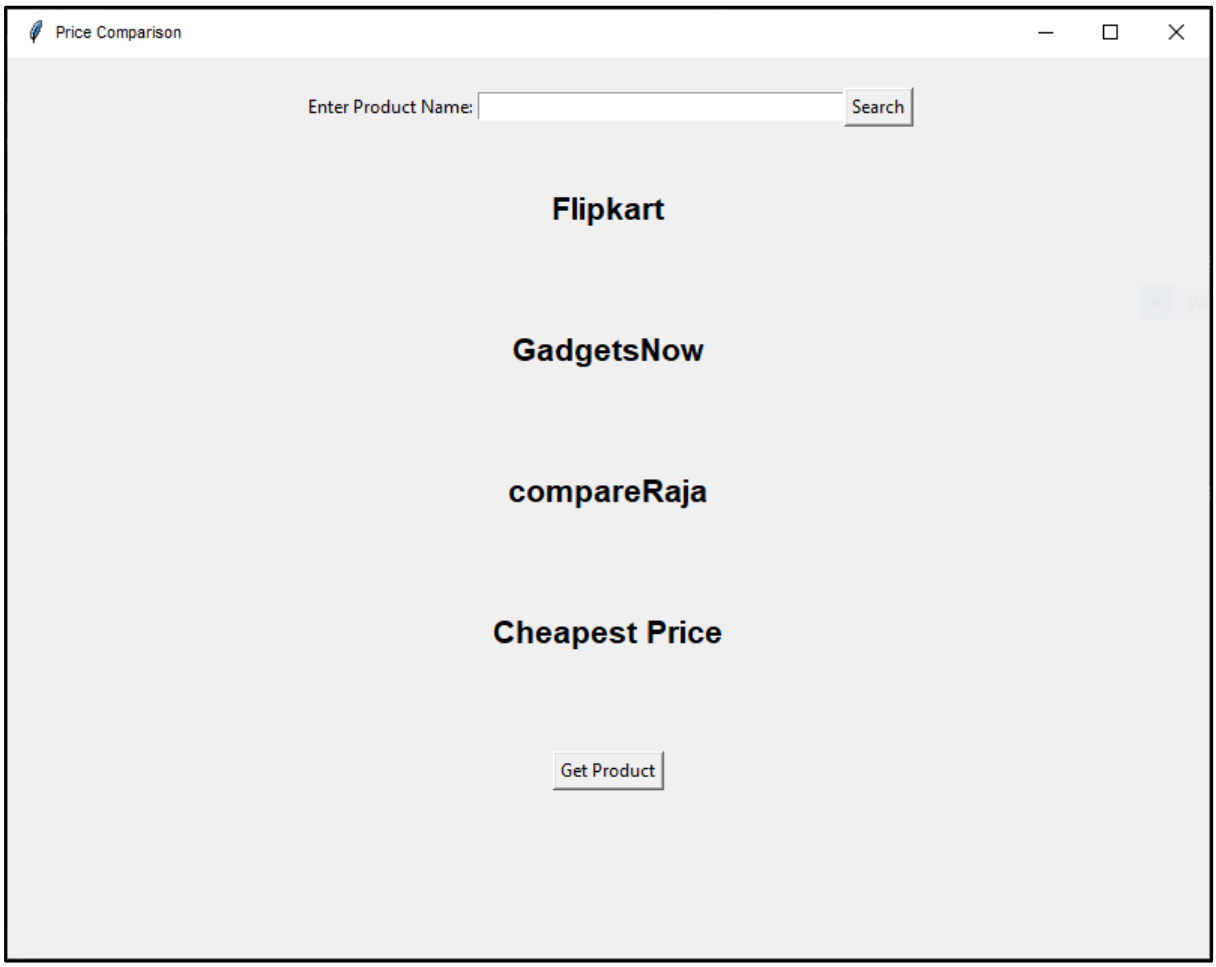

Figure 1 User Interface

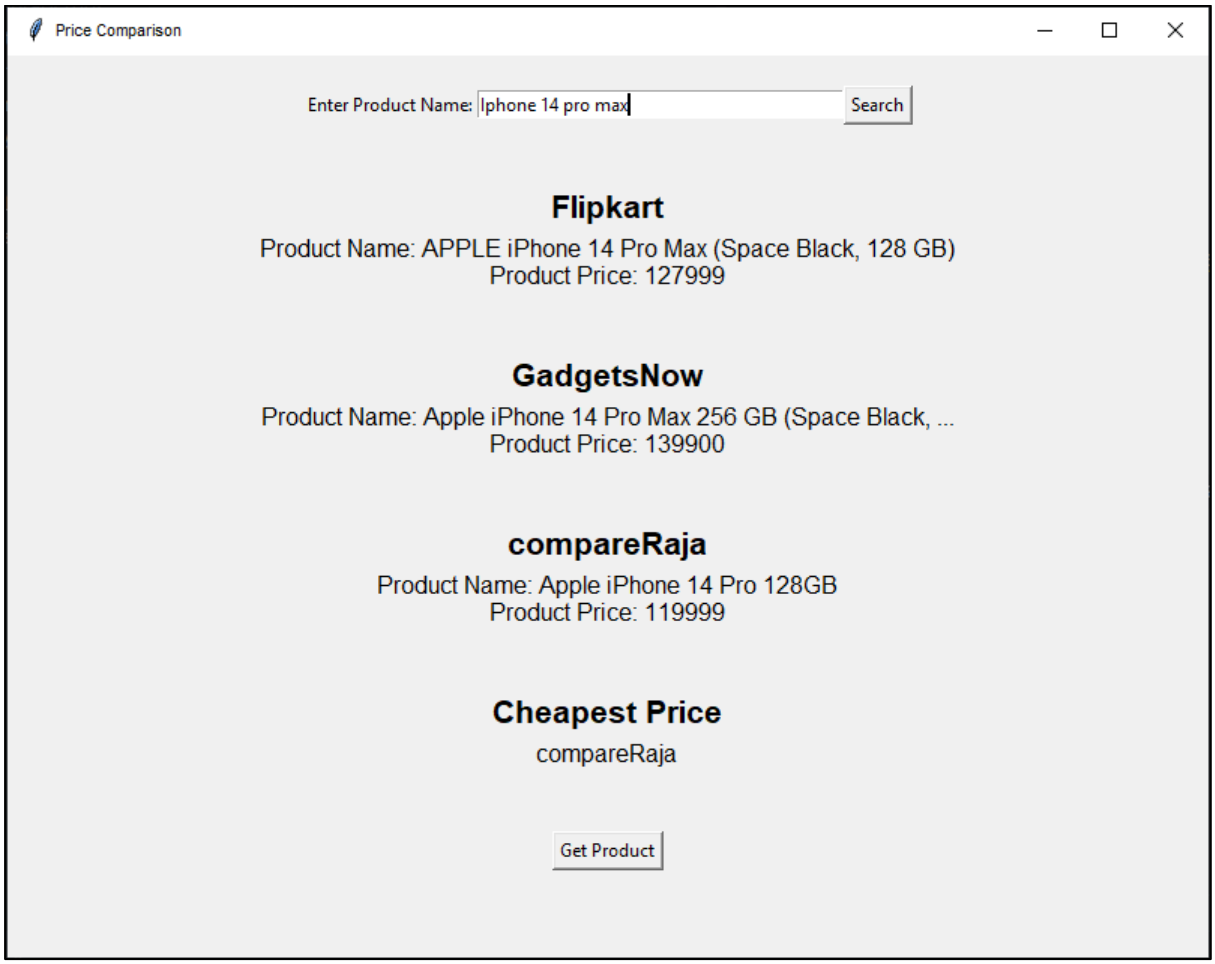

Figure 2 Running Phase

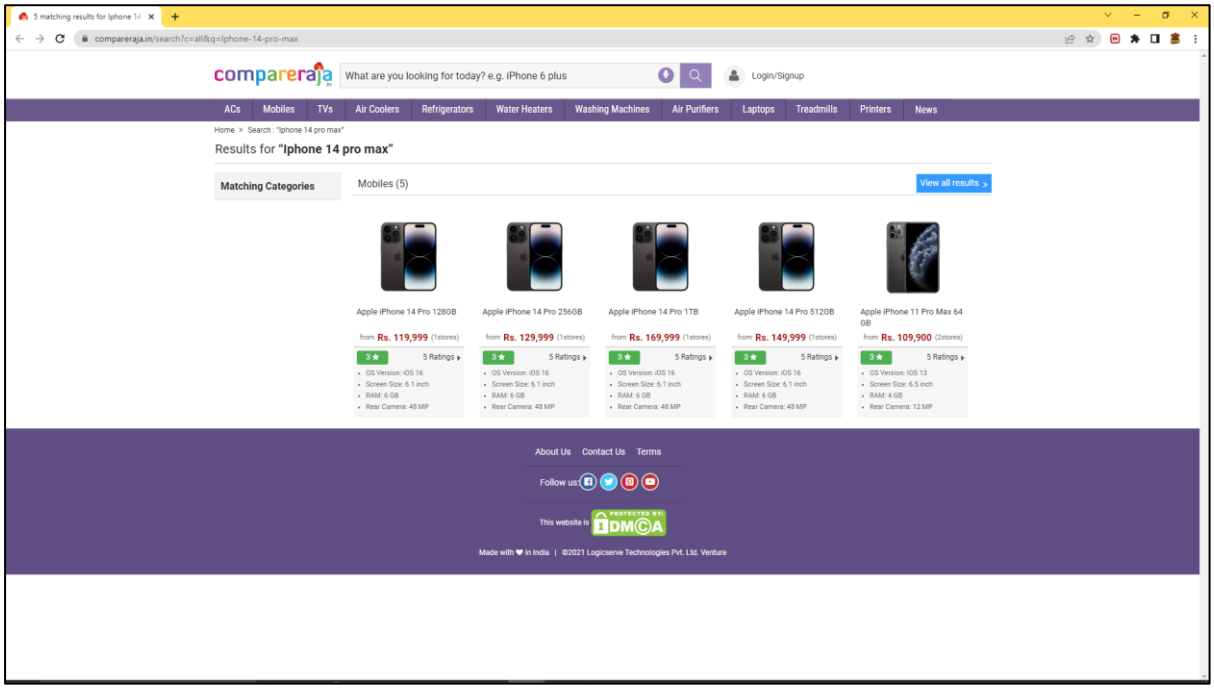

Figure 3 Output Website

VIII. Advantages and Disadvantages

A. Advantages

- **Cost Savings:** Price comparison using web scraping and data analysis can help users find the best deals and discounts across multiple e-commerce websites. This can result in significant cost savings for consumers.

- **Time Efficiency:** By automating the process of gathering and comparing prices, users can save time that would otherwise be spent manually visiting each website and searching for the desired products.

- **Comprehensive Comparison:** Web scraping allows for the collection of extensive data from various sources, enabling a comprehensive comparison of prices across different websites and products. This provides users with a broader understanding of market prices and trends.

- **Data-Driven Decisions:** With the availability of accurate and up-to-date price data, users can make informed purchasing decisions based on real-time information. They can identify the best offers and make optimal choices based on their preferences and budget.

- **Price Tracking:** By regularly scraping and analyzing price data, users can track changes in prices over time. This information can be valuable for predicting price fluctuations and identifying the best time to make a purchase.

B. Disadvantages

- **Website Restrictions:** Some e-commerce websites may have anti-scraping measures in place, making it challenging to extract data or even block access to scraping bots. This can limit the effectiveness and reliability of the price comparison system.

- **Data Accuracy:** Web scraping relies on the structure and formatting of websites, and any changes to the website's layout or HTML structure can break the scraping process and lead to inaccurate or missing data.

- **Legal and Ethical Considerations:** Web scraping activities may infringe on the terms of service or copyright of the targeted websites. It is essential to respect the website's policies and ensure compliance with legal and ethical guidelines while scraping data.

- **Limited Product Coverage:** Depending on the websites included in the scraping process, the system may not cover all available products or niche markets. This can result in incomplete or biased price comparisons, especially for specialized or less popular products.

- **Dependency on Website Availability:** The reliability of the price comparison system is contingent on the availability and accessibility of the target websites. If a website goes offline or experiences technical issues, it can affect the scraping process and the availability of up-to-date price data.