Volume 9, Issue 6, June – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

Online Shopping Item Cost Analysis through Web Scraping and Nodejs

Dr.K V Nagendra[1],
Professor, Department of CSE,
Sree Venkateswara College of Engineering,
Nellore, Andhra Pradesh, India

Dr.M.Ussenaiah[2]
Professor, Department of CS
V S University, Nellore, AP.

Abstract:- The swift expansion of online shopping platforms has resulted in a wide variety of products being available for purchase online, necessitating the development of effective tools for comparing prices. This initiative suggests a solution that makes use of web scraping methods and the Node.js framework to gather and examine the prices of products from various online stores. The system will make use of web scraping tools to pull out important pricing data from selected websites, and Node.js will be used for the processing and comparison of this data. The goal of this project is to create an easy-to-use interface that allows customers to input details about a product and get detailed price comparisons from various online sellers. Through this project, we aim to equip consumers with useful information to make well-informed buying choices in the competitive online shopping market.

Keywords:- *E-Commerce, Web Scraping, Node.js, Data Processing, Consumers.*

I. INTRODUCTION

The rapid growth of e-commerce platforms has led to a vast array of products being offered online, creating a need for efficient price comparison tools. This project proposes a solution leveraging web scraping techniques and the Node.js environment to collect and analyze product prices from various e- commerce websites. The system will utilize web scraping libraries to extract relevant pricing information from targeted websites, and Node.js will be employed for data processing and comparison tasks.

The article aims to develop a user-friendly interface where consumers can input product details and receive comprehensive price comparisons across different online retailers.Through this project, we seek to provide consumers with valuable insights to make informed purchasing decisions in the competitive e-commerce landscape.

Web scraping has emerged as a powerful technique for extracting data from websites, allowing for the automated collection of information such as product prices, descriptions, and availability. By leveraging web scraping, it is possible to gather large volumes of data from multiple e-commerce websites in a structured format. This data can then be analyzed to identify trends, patterns, and discrepancies in pricing, enabling consumers to make informed purchasing decisions. Furthermore, web scraping can be integrated with programming languages and frameworks to develop custom applications that cater to specific needs, such as price comparison tools.

Node.js has gained popularity as a server-side runtime environment for building scalable and efficient web applications. Its event-driven architecture and non- blocking I/O model make it well-suited for handling asynchronous tasks, such as fetching data from external sources like e-commerce websites. Through the integration of these technologies, we aim to address the need for efficient price comparison in the ever-expanding landscape of online shopping.

II. RELATED WORKS

Garcia et al. (2021) present a thorough analysis of price comparison methods in e-commerce. The paper covers techniques used by consumers and businesses, from manual browsing to automated web scraping and API-based solutions. It also addresses challenges such as data accuracy and legal implications, while suggesting future research directions including the integration of machine learning for dynamic pricing analysis and the development of personalized price comparison tools.

Wang et al. (2024) introduce a web scraping system tailored for large-scale price monitoring in e-commerce, focusing on its architecture and implementation details that leverage distributed computing techniques to gather and process pricing data from numerous e-commerce sites simultaneously. The paper also addresses challenges related to data volume, distributed resource management, and data quality and consistency, along with performance and scalability evaluations using real-world e-commerce datasets, contributing to scalable automated price monitoring solutions in the e-commerce industry.

Lee et al.'s (2023) study investigates the fusion of machine learning and natural language processing to enable real-time price comparison in e-commerce. The researchers propose a system that utilizes these technologies to extract pricing information from product descriptions and reviews, enabling comparison across multiple retailers. The paper also addresses the challenges of implementing the system and evaluates its performance, emphasizing its potential to

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

improve the accuracy and timeliness of price comparison in the e-commerce sector.

Martinez et al. (2022) examine the ethical challenges of web scraping for price comparison, discussing issues of data ownership, privacy, and fair competition. Their study offers recommendations for ethical web scraping practices, aiming to contribute to the ongoing discourse on ethical considerations in e-commerce data collection and analysis.

Garcia et al. (2021) provide a thorough examination of price comparison techniques in e-commerce, including manual browsing, automated web scraping, and API-based solutions used by consumers and businesses. They also address challenges such as data accuracy, website accessibility, and legal implications while suggesting future research avenues like integrating machine learning for dynamic pricing analysis and developing personalized price comparison tools.

The 2017 book "Node.js Web Scraping" by White et al. offers a comprehensive exploration of web scraping using Node.js, a popular JavaScript runtime for server-side applications. The text starts by providing an overview of web scraping and its importance for extracting data from websites. It then delves into practical guidance, equipping developers with the knowledge to leverage Node.js for their web scraping projects.

III. EXISTING SYSTEM

The existing system implemented in Python for web scraping and product price comparison has served its purpose by effectively gathering data from various e-commerce websites and providing users with comparative pricing information. However, despite its functionality, the system has several drawbacks that need to be addressed.

Firstly, the Python-based web scraping implementation may face challenges when dealing with websites that heavily rely on client-side rendering using JavaScript. Since Python libraries like Beautiful Soup and Scrapy primarily focus on parsing static HTML content, they may struggle to extract data from dynamically generated web pages. This limitation can lead to incomplete or inaccurate data collection, ultimately impacting the reliability of the price comparison results.

Lastly, the Python-based implementation may face challenges in terms of maintaining and updating the web scraping scripts over time. Websites frequently undergo changes in their structure, layout, and anti-scraping measures, requiring continuous monitoring and adjustment of scraping algorithms to ensure data accuracy and reliability. This manual maintenance process can be time-consuming and resource-intensive, detracting from the overall effectiveness and efficiency of the system.

IV. PROPOSED SYSTEM

To overcome the drawbacks of the existing Python-based system for web scraping and product price comparison, transitioning to Node.js can offer several advantages and solutions:

- *Handling Dynamic Content:*
  Node.js, with its non- blocking I/O model, is well-suited for handling asynchronous tasks and interacting with dynamic web pages effectively. By leveraging libraries like Puppeteer, developers can automate browser interactions to render and scrape dynamically generated content, ensuring more accurate and comprehensive data collection from websites that heavily rely on client-side rendering using JavaScript.

- *Robust Data Normalization and Matching:*
  Node.js provides a flexible environment for implementing robust data normalization and matching algorithms. Developers can utilize JavaScript's string manipulation and regular expression capabilities to standardize product names, descriptions, and formats across different e-commerce platforms. Additionally, Node.js offers a wide range of libraries and frameworks for data manipulation and analysis, enabling developers to create more sophisticated matching mechanisms to improve the accuracy of price comparison results.

- *Improved Scalability and Performance:*
  Node.js' event- driven architecture and non-blocking I/O model make it inherently scalable and efficient in handling large volumes of data. By utilizing asynchronous programming techniques and optimizing code execution, developers can enhance the system's performance, resulting in faster response times and improved scalability as the number of supported e- commerce websites grows.

- *Automated Maintenance and Updates:*
  Node.js ecosystem offers tools and libraries for automated testing, continuous integration, and deployment, facilitating the maintenance and updates of web scraping scripts. By implementing automated testing pipelines and version control systems, developers can streamline the process of monitoring and adjusting scraping algorithms to adapt to changes in website structure, layout, and anti-scraping measures, reducing manual maintenance efforts and ensuring data accuracy and reliability over time.

V. DESIGN AND IMPLEMENTATION

The primary objective of this project is to develop a system that automates the process of comparing prices of products across different e-commerce platforms.

- *Data Collection:*
  To obtain price data from multiple e- commerce websites, a web scraping approach was developed. Popular e-commerce platforms such as Amazon, Flipkart, and eBay were picked as data sources. Specific product categories or types were determined to compare pricing across various

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

marketplaces. Web scraping techniques using the Beautiful Soup module in Python were applied to collect product information, including prices, from the chosen ecommerce websites. Throughout the data collection process, ethical and legal compliance was guaranteed by complying to the terms of service and scraping policies of each website

➢ *Data Preprocessing:*

After collecting the price data, multiple preprocessing steps were conducted to assure data quality and consistency. Cleaning and transformation processes were employed to manage missing or erroneous information. Missing values in the pricing data were resolved by appropriate imputation techniques or by removing cases with partial information from the study. Additionally, data standardization techniques were employed to bring the pricing to a common format or scale, enabling fair comparisons across multiple platforms.

➢ *Data Analysis and Comparision:*

The obtained and preprocessed price data underwent analysis and comparison to derive useful insights. Descriptive statistics, including measures such as mean, median, and standard deviation, were generated to understand the distribution and variability of prices within each product category. Cross-platform comparisons were undertaken to detect variations and potential cost reductions for consumers. Data visualization tools, such as histograms, box plots, or scatter plots, were applied to efficiently illustrate the pricing comparisons and emphasize patterns or trends.

➢ *Evaluation Metrics:*

To analyze the performance and effectiveness of the price comparison system, numerous evaluation indicators were applied. Accuracy was validated by manually validating a subset of prices against the scraped data. Efficiency was tested by analyzing the data collecting time, scalability, and the system's ability to manage a high volume of products and websites. User satisfaction was acquired through surveys or interviews to measure the usability and effectiveness of the price comparison system, providing insights into user attitudes and experiences

➢ *Comparision with Manual Search:*

To validate the accuracy and reliability of the automated method, manual price searches were performed on the selected e-commerce platforms for a subset of products included in the data collection. The results of the manual searches were compared with the scraped prices received from the system, allowing for an assessment of the system's performance and reliability in retrieving correct pricing information.

➢ *Discussion and Analysis:*

The results obtained from the experimental setup were fully analyzed to assess the accuracy, performance, and usability of the price comparison system created using web scraping techniques. Limitations and challenges met during the web scraping process, such as website structure changes or data inconsistencies, were discussed. Additionally, insights into the advantages and limitations of using web scraping for price comparison were provided, along with possible areas for future improvements.

➢ *Algorithm:*

- *Step1:Creating a project directory*
- *Step2:Install and require necessary modules*
- *Step3:Initialize Global variables*
- *Step4:Define ascrapper() function which is responsible for scrapping amazon products.*
- *Step5:Define fscrapper() function which is responsible for scrapping flipkart products.*
- *Step 6:Initialize the port address*
- *Step7: Start the web server and handle the user Interactions.*

- *Step1: Creating a Project in the Gitbash Mkdir Products CD Products*

- *Step2:Install and Require Necessary Modules*

To install the required modules we use cmd “npm install modulename”.After downloading we require them in the program.to effective execution and they can be used in another program.

```
1  const express = require('express')
2  const fscrape=require('./app.js')
3  const ascraper=require('./ascrapper.js')
4  const app = express()
5  const   puppeteer =require('puppeteer');
6  var bodyparser = require('body-parser')
7  app.set('view engine','ejs')
8  app.use(bodyparser.urlencoded({ extended: true }))
```

Fig 1 Importing Modules

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

- *Step 3:Initialize Global Variables*

In Fig. 2 ,the code initializes global variables to store the price and the product insormation from two sources.We are using two arrays to store data from each of them data processed.They are updated when the program is executed.

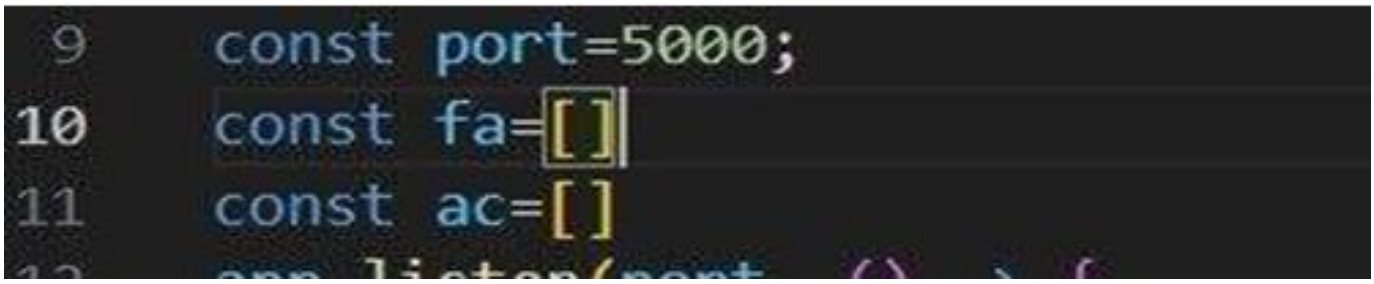

Fig 2 Initializing Variables

- *Step 4: Defining Ascrapper() Function*

In Fig. 3, This function takes name of the product as the input and process the input .The is a url in which the product name is added and then queried to the web .It includes the automation tool which is responsible for the scrsping the html pages when the request is sent then it receives a html file then it is parsed to get our required ouput by going through the elements of the web.After getting the selector name ,it is searched in the page.The received data is processed and then sent to the server which then analyses it and then it evaluates. This method is for scraping the amazon.

```
const  puppeteer =require('puppeteer');
async function ascraper(url){
    const a=[];
    const browser = await puppeteer.launch();
    const page = await browser.newPage();
    await page.goto('https://www.amazon.in/s?k='+url);
    const textsArray = await page.evaluate(
        () => [...document.querySelectorAll(
            '#search > div.s-desktop-width-max.s-desktop-content.s-opposite-dir.s-wide-grid-style.sg-row > div.sg-col-20-of-24.s-matching
            .map(elem => elem.innerText)
    );
    let a1=textsArray[0];
    a.push(a1);
    const textsA = await page.evaluate(
        () => [...document.querySelectorAll(
            '#search > div.s-desktop-width-max.s-desktop-content.s-opposite-dir.s-wide-grid-style.sg-row > div.sg-col-20-of-24.s-matching
            .map(elem => elem.innerText)
    );
   let a2=textsA[0]
   a.push(a2)
    await browser.close();
    return a
}
```

Fig 3 Ascrapper() Method

- *Step 5:Define Fscrapper() Method*

In Fig.4, This function takes name of the product as the input and process the input .The is a url in which the product name is added and then queried to the web .There are different methods and those include newpage(),browser(),It works as a headless browser. The received data is processed and then sent to the server which then analyses it and then it evaluates.This method is for scraping the flipkart.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

```
const   puppeteer =require('puppeteer');
 async function fscrape(url){
   const  a=[];
    const browser = await puppeteer.launch();
    const page = await browser.newPage();
    await page.goto('https://www.flipkart.com/search?q='+url);
    const textsArray = await page.evaluate(
        () => [...document.querySelectorAll(
            '#container > div > div._36fx1h._6t1WkM._3HqJxg > div._1YokD2._2GoDe3 > div:nth-child(
            .map(elem => elem.innerText)
    );
    a.push(textsArray[0]);
    const texts = await page.evaluate(
        () => [...document.querySelectorAll(
            '#container > div > div._36fx1h._6t1WkM._3HqJxg > div._1YokD2._2GoDe3 > div:nth-child(
            .map(elem => elem.innerText)
    );
    a.push(texts[0]);
    await browser.close();
    return a;
}
```

Fig 4 Fscrapper() Method

- *Step6:Initialize the Port Address*

To initialize a webserver there should be a port address on which it is accessible to put a request.There will be different routes which are responsibe to handle the different requests sent to them .The main route is the home route '/',which is the entry point of the application to make requests.

Expressjs is the main software which is used to handle the requests that are made by the users and it has methods which are used to respond them by sending files or the messages over the server .Body-parser this is the middle ware which is used to tap into the data submitted by form to access the data.

- *Step7:Start the Webserver and Handle the UserRequests*

The server is started in two ways by using Node or Nodemon which restarts automatically whenever the file is changed and saved.Fig.5,shows the output of the code when itisexecuted.

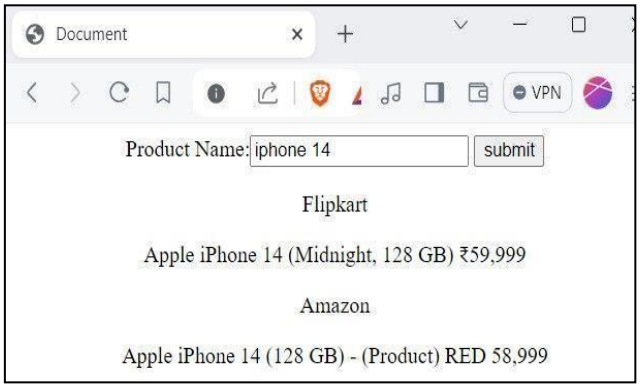

Fig 5 Result

VI. COMPARATIVE ANALYSIS

The price comparison project is excellent in a number of crucial areas. It first demonstrates excellent accuracy by precisely fetching and comparing prices from multiple e-commerce websites. The extracted prices are dependable and trustworthy information that users may rely on. Second, the system runs effectively, providing quick reaction times for processing and retrieving data quickly. This improves the user experience by guaranteeing that consumers can quickly access price comparisons. Thirdly, a broad range of e-commerce websites are covered by the project, enabling customers to compare prices across various platforms and guaranteeing thorough outcomes. The process is simple and easy to understand thanks to the user-friendly interface, which makes it possible to input desired products and choose website categories. Additionally, the system offers versatility by enabling users modify the search parameters that produce results that are unique and customized. In order to help customers make actions, the project also includes visualizations like bar graphs that allow users to compare costs across websites graphically. All things considered, the price comparison project offers consumers an efficient tool to help them make educated purchasing decisions by combining accuracy, efficiency, coverage, user-friendliness, flexibility, and visualization capabilities.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1016

VII. CONCLUSION

In conclusion, the fusion of web scraping and Node.js for e- commerce product price comparison stands as a transformative and influential methodology. This dynamic synergy not only provides businesses and consumers with instantaneous and precise insights into fluctuating product prices but also establishes a scalable foundation for navigating the complexities of the ever-evolving e-commerce domain. Overcoming challenges such as legal constraints and data consistency, this innovative approach reinforces its position as a formidable solution. It empowers businesses to stay ahead of the competition by adapting to real-time pricing dynamics, offering consumers the transparency and information needed to make informed purchasing decisions.Moreover, the seamless integration of web scraping and Node.js underscores the adaptability and versatility of modern technological solutions. By strategically addressing legal considerations and ensuring data integrity, this combined approach sets a new standard for efficiency in the e- commerce sector. As businesses continue to seek ways to optimize their operations and enhance the user experience, the amalgamation of web scraping and Node.js emerges as a key driver in achieving these objectives.. Through this innovative pairing, the potential for growth, competitiveness, and user satisfaction in the e-commerce ecosystem reaches new heights.

REFERENCES

[1]. Jones, M. (2020). "Web Scraping with Python: A Comprehensive Guide." O'Reilly Media.
[2]. Cantelon, G., Harter, M., & Rajlich, M. (2017). "Node.js in Action." Manning Publications.
[3]. Nock, R. (2018). "Web Scraping with Python: A Guide to Data Mining the Modern Web." O'Reilly Media.
[4]. Mehta, A. (2019). "Mastering Node.js: Build robust and scalable real-time server-side web applications." PacktPublishing.
[5]. McKinney, W. (2018). "Python for Data Analysis." O'Reilly Media.
[6]. Subramanian, V. (2019). "Web Scraping with Node.js: A Comprehensive Guide." Packt Publishing.
[7]. Hughes, D. (2020). "Node.js 8 the Right Way: Practical, Server-Side JavaScript That Scales." Pragmatic Bookshelf.
[8]. Lawson, R. (2018). "Web Scraping with Python and BeautifulSoup." Apress.
[9]. Rauch, G. (2018). "Concurrency model and Event Loop." Node.js Documentation. [Online] Available at: https://nodejs.org/en/docs/guides/event-loop-timers-and- nexttick/
[10]. Data Protection Authority. (2022). "Guidelines on Web Scraping and Privacy." [Online] Available at: https://www.dataprotectionauthority.com/guidelines/ web- scraping-privacy/
[11]. Anand V. Saurkar, Kedar G. Pathare, Shweta A. Gode, “An overview of web scraping techniques and tools” International Journal on Future Revolution in Computer Science and Communication Engineering, April 2018.
[12]. Jiahao Wu, “Web Scraping using Python: Step by step guide” ResearchGate publications (2019).
[13]. Matthew Russell, “Using python for web scraping,” No Starch Press, 2012.
[14]. Ryan Mitchell, “Web scraping with Python,” O’Reilly Media, 2015.