SPEEDING UP INFORMATION COLLECTION BY COMBINING SEARCH AND SCRAP

A PROJECT REPORT

Submitted by

GUHAN M P S (20C024)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI – 15
(A Government Aided Autonomous Institution Affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

THIAGARAJAR COLLEGE OF ENGINEERING

MADURAI-15

(A Government Aided Autonomous Institution Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report "**Speeding up Information collection by combining search and scrap"** is the bonafide work of **"GUHAN M P S (20C024)"** who carried out the project work under my supervision during the Academic Year 2023-2024.

SIGNATURE

Dr.S. Mercy Shalinie, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE AND ENGINEERING
THIAGARAJAR COLLEGE OF ENGG.
MADURAI – 625 015

SIGNATURE

Mr.D.Nagendra Kumar, M.E.,
ASSISTANT PROFESSOR
COMPUTER SCIENCE AND ENGINEERING
THIAGARAJAR COLLEGE OF ENGG.
MADURAI – 625 015

Submitted for the VIVA VOCE Examination held at Thiagarajar College of Engineering on 06/05/2024

INTERNAL EXAMINER

EXTERNAL EXAMINER

Acknowledgement

I wish to express my deep sense of gratitude to **Dr.M. Palaninatharaja,** Principal of Thiagarajar College of Engineering for his support and encouragement throughout this project work.

I wish to express my sincere thanks to **Dr.S.Mercy Shalinie**, Head of the Department of Computer Science and Engineering for her support and ardent guidance.

I owe my special thanks and gratitude to **Mr.D.Nagendra Kumar**, Assistant Professor**,** Department of Computer Science and Engineering for his guidance and support throughout our project.

I am also indebted to all the teaching and non-teaching staff members of our college for helping us directly or indirectly by all means throughout the course of our study and project work.

I extremely thank my parents, family members and friends for their moral support and encouragement for my project.

Abstract

Website information collection is a process that plays a crucial role in web development and data analysis. It involves gathering and acquiring data from websites, which can then be utilized for a variety of purposes such as research, marketing, and content aggregation. The information gathered from websites can provide valuable insights and help businesses make informed decisions to stay competitive in the digital landscape.One of the essential steps in website information collection is extracting data from websites.

This process involves using specialized tools or techniques to retrieve the desired information from web pages. There are several approaches to extracting data, including manual extraction and automated scraping. Manual extraction involves manually visiting a website and copying the relevant data by selecting and copying the text or images from the webpage. While this method may be suitable for small-scale data collection or when dealing with websites that have limited data, it is time-consuming and not efficient for large-scale projects. Moreover, it requires human effort, which can lead to human errors and inconsistencies in the collected data.

Automated scraping, on the other hand, leverages software tools or scripts to extract data from websites automatically. These tools can navigate through the website's pages, interact with the content, and extract the desired information. Automated scraping is highly efficient and can handle large amounts of data. However, it is important to note that website scraping must be done ethically and within the boundaries of legal frameworks to respect website owners' terms of service and privacy policies.To effectively extract structured data from websites, several methodologies and techniques can be employed.

Some websites provide APIs, which are interfaces that allow developers to access and retrieve data from the website's backend directly. APIs often provide a more efficient and controlled way to access data compared to web scraping. Web scraping is more versatile and can extract data from any website, but it requires more technical expertise to implement and maintain.

This project work uses an orchestration mechanism to facilitate efficient fetching of scrapped data by combining search and scraping APIs so as to fasten information retrieval available in the world wide web.

Table of Contents

List of Figures

List of Abbreviations

Acronym	Abbreviation
CPU	Central Processing Unit
RAM	Random Access Memory
S3	Simple Storage Service
DDOS	Distributed Denial Of Service
URL	Uniform Resource Locator
CSV	Comma Separated Values
SDK	Software Development Kit

1. Introduction

1.1 Data Need:

In the current era of technology, information has become a vital resource for organizations in various sectors. By skillfully gathering, examining, and utilizing data, organizations can gain valuable information that informs their choices, improves their performance, and sparks creativity. With the growing importance of data in industries, the necessity for strong data collection systems has become more critical than ever.

Data collection is the basis for a variety of business processes, including market research, customer segmentation, product development and performance measurement. By aggregating data from a variety of sources, such as websites, sensors, social media, and internal systems, organizations can gain a comprehensive understanding of their operations, customers, and market landscape.

Industries such as retail, finance, healthcare, and manufacturing use data collection to gain competitive advantage. For example, retailers use data to analyze consumer behavior and preferences to deliver personalized marketing campaigns and inventory management. In finance, data collection enables risk assessment, fraud detection and algorithmic trading. In healthcare, data-driven insights support clinical decision-making, patient care and disease management. Also, data collection in manufacturing helps with predictive maintenance, supply chain optimization and quality control.

The emergence of technologies such as the Internet of Things (IoT), artificial intelligence (AI) and big data analytics have further increased the need for data collection. These technologies allow

organizations to instantly collect and analyze large amounts of data and discover patterns and trends that were previously unavailable.

While there are many benefits to collecting data, organizations must contend with challenges such as data privacy, security, and ethical considerations. To mitigate these risks, it is important to ensure compliance with regulations such as the General Data Protection Regulation (GDPR) and implement strong data management practices.

1.2 Web Scraping:

Web scraping has become a powerful tool for extracting and transforming data from websites, providing valuable insights for businesses, researchers, and individuals. In today's digital age, when information is abundant but often scattered across the web, web crawling provides an efficient way to collect and analyze data.

Web scraping is the process of extracting information from websites. This involves sending an HTTP request to the website, downloading the HTML content of the page, and then parsing the HTML to extract the required information. Python is a popular web scripting language due to its simplicity and the availability of powerful libraries and frameworks.

Web scraping involves automatically extracting data from websites, often using programming languages such as Python and specialized libraries such as BeautifulSoup and Scrapy. BeautifulSoup allows you to parse HTML and XML files, navigate parse trees, and extract data based on tags, attributes, and other criteria. It provides a simple and intuitive HTML interface to easily extract data from web pages. This process allows users to collect a wide range of data from different websites, including text, images and structured information.

Additionally, Scrapy is a Python framework specifically designed for web scraping. Scrapy is a web crawler and web scraping system that provides a powerful set of tools for extracting data from websites. It allows you to use XPath or CSS selectors to define the structure of the pull-out material, and then automatically crawls the site to pull the material according to your specifications. It provides a set of tools and utilities for extracting data from websites, including a powerful web crawler that can crawl multiple pages and extract information based on specific rules.

In addition to the beautiful group and sandpaper, Python also offers other library and network scraping tools such as requests, LXML and selenium. Demand is a library that sends HTTP request Python, commonly used with Beautifulsoup or other parsing libraries. Lxml is a fast and efficient library for parsing XML and HTML files and can be used as a replacement for BeautifulSoup. Selenium is a web automation tool that can be used to crawl websites that require JavaScript to render their content.

The applications of web scraping are varied and influential in different industries. In e-commerce, companies use web scraping to monitor competitor prices, collect product reviews, and analyze market trends. In finance, web scraping can collect real-time data on stock prices, financial news, and economic indicators. In the academic world, researchers use web scraping to collect data for research and analysis in fields such as social science, economics, and health.

Implementing web scraping involves several important steps. First, the user must identify the target site and the specific data elements to be extracted. Users then use programming languages and related libraries to develop scraping scripts to specify how the data should be retrieved and

processed. The cultivated data is then typically stored in a structured format such as a CSV file or database for further analysis or use.

While web copying offers many advantages, including data collection, automation and efficiency, it also raises ethical and legal concerns. Users must comply with the site's terms of service, respect the robots.txt file and avoid overloading the site with too many requests. In addition, the collection of personal or sensitive data requires consent and compliance with applicable data protection laws.

1.3 Web Search:

Web browsing is the process of retrieving information from the vast and interconnected network of web pages on the Internet. This is a complex and dynamic field that uses specialized algorithms and techniques to index, rank, and retrieve relevant content in response to user queries. Web search is based on the principles of information retrieval (IR), the science of finding and retrieving information from collections of data.

The primary goal of web search is to provide users with relevant, quality information in a timely manner. This is achieved through search engines, which are software systems designed to index web pages, process user queries and return a list of matching results. Search engines use a variety of technologies to rank web pages, including site content analysis, indicating the number and quality and reputation of its links on the site.

One of the biggest challenges in web search is the size and complexity of the web. With billions of web pages and new content being added all the time, search engines need to be able to index and retrieve relevant information quickly and efficiently. To deal with this problem,

search engines use crawling and indexing algorithms to discover and catalog web pages, and ranking algorithms to determine the order in which results are presented to users.

Web search is also a highly interdisciplinary field, drawing on insights and techniques from computer science, information science, linguistics, and other fields. As technology continues to evolve, web search is likely to continue to evolve as new algorithms, technologies and applications emerge to improve the search experience for users around the world.

Web search using the Python Software Development Kit (SDK) harnesses the power of programming to interact with search engines and retrieve information from the web. Python is a popular programming language that offers several SDKs and libraries that simplify the process of building web search applications. BeautifulSoup is one such library that allows developers to parse HTML and XML files, extract relevant information, and navigate web page structures.

Another powerful Python tool for web browsing is Selenium, a web automation tool that allows developers to simulate user interactions with web pages. Selenium can be used to automate tasks such as filling out forms, clicking buttons, and browsing websites, making it ideal for web maintenance and data mining.

Using these Python SDKs, developers can build complex web search applications that retrieve, parse, and analyze information from the web, allowing them to build intelligent search engines, data mining tools, and more. Python's simplicity and versatility make it an excellent choice for web search projects, allowing developers to quickly and easily build powerful web search applications.

1.4 Orchestration:

Orchestration in computing is a systematic approach to automating, coordinating, and managing complex systems, workflows, or services. This is essential in today's computing environments, especially cloud computing and distributed systems, where multiple components must work seamlessly. Orchestration involves sequencing tasks, managing dependencies, handling errors, and ensuring efficient use of resources to achieve specific goals or outcomes.

Essentially, orchestration abstracts away the complexity of the underlying system, allowing users to focus on defining high-level workflows and business logic. This abstraction is facilitated by orchestration tools and frameworks that provide a single interface to manage and coordinate resources across environments. These tools typically include functionality to define workflows, manage dependencies, monitor execution, and handle errors.

One of the main benefits of orchestration is the ability to increase system scalability, reliability, and efficiency. By automating the deployment and management of resources, orchestration reduces manual intervention, reduces human error, and ensures consistent and repeatable results. This is especially important in cloud computing, where applications often consist of multiple services that must be dynamically deployed, scaled, and managed.

There are several orchestration approaches, including imperative and declarative models. In imperative orchestration, the orchestration engine specifies the exact steps to take to achieve the desired state. In contrast, declarative orchestration focuses on defining a desired system state, and the orchestration engine decides the necessary steps to achieve

that state. Declarative approaches are often preferred because of their simplicity and flexibility.

Orchestration is closely related to other concepts such as automation, DevOps, and continuous integration/continuous deployment (CI/CD). By automating application deployment and management, orchestration enables organizations to adopt DevOps practices that emphasize collaboration, automation, and integration between development and operations teams. CI/CD pipelines automate the development, testing, and deployment of applications, often relying on orchestration tools to coordinate the various stages of the pipeline.

Orchestration is a key element of today's computing environment, enabling organizations to effectively manage complex systems and workflows, abstract the complexity of underlying systems, and automate the deployment and management of their applications. As computing environments become more distributed and complex, orchestration will continue to be critical to the reliability, scalability, and efficiency of these systems.

IT teams manage various servers, systems and applications in private data centers, cloud and edge locations. As IT environments become more complex, automating tasks can increase efficiency and make processes more manageable, but scaling automation has its own challenges. Most IT processes involve many separate tasks that need to be automated. To fully automate a process, the tasks that make up the process must also work together, when a task is completed, the corresponding next tasks must start.

Some automation solutions can connect tasks to logical workflows, eliminating manual work to trigger actions at the right time. Structuring these workflows is an element of orchestration. However, each part of the task may involve multi-step workflows that depend on communicating

with third-party systems. For example, system provisioning typically includes orchestrating with hypervisors to create virtual machines, communicating with the network to configure and establish connectivity, and confirming that all necessary firewall policies are in place. Here comes a comprehensive orchestration solution that can coordinate tasks across systems, allowing IT teams to create fully automated workflows that span use cases across the enterprise.

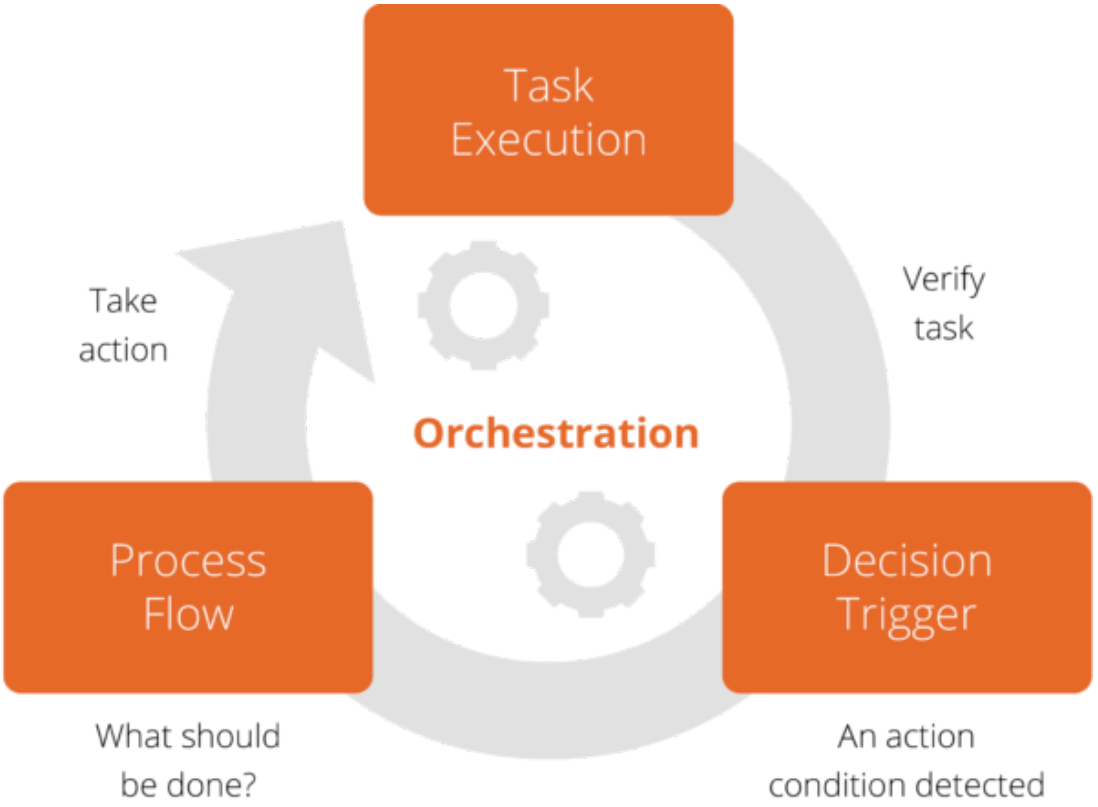


Fig 1.1 Orchestration flow

1.5 Metadata handling:

Metadata refers to data that describes other data. In a computing environment, metadata provides information about the properties of a material, such as its structure, format, location, and purpose. Metadata

management is critical in today's computing environments because it helps organize, manage, and interpret data effectively.

One of the key requirements for metadata processing is data management. Metadata provides basic information about the material, such as its origin, ownership and rights of use, which is essential to ensure its quality, safety and compliance. For example, metadata can help identify sensitive data that needs to be protected or trace data lineage to ensure its accuracy and reliability.

Metadata processing is also important for data discovery and retrieval. By adding metadata to data, organizations can easily search, filter and retrieve relevant information, even in large and complex datasets. For example, search engines use metadata to display web pages and provide users with relevant search results.

Another critical need for metadata management is data integration. In today's computing environments, data is often stored and managed across multiple systems and platforms. Metadata provides a common framework for describing data in these systems, facilitating the integration and analysis of data from different sources. For example, metadata can help map data fields between different databases or systems, ensuring seamless data integration.

Metadata management is also essential for data governance and compliance. Metadata provides visibility into how data is managed and used within an organization and helps ensure that data is stored, accessed and used in accordance with regulatory requirements and internal policies. For example, metadata can help identify data subject to specific regulations, such as personal data protected by GDPR or HIPAA.

Metal data processing plays an important role in modern computers, allowing the organization to organize, manage and explain data effectively. By providing basic information on data, metadata can

help ensure data quality, security and conformity and promote data disclosure, giving, integration and management. Metadata management is essential to data governance as it provides the information needed to ensure that data is properly managed throughout its lifecycle. This includes defining data ownership, access permissions, retention policies and data lineage. By effectively managing metadata, organizations can ensure responsible use of data and compliance with legal requirements.

2. Literature Survey

An overview of related research that examines web scraping in real world contexts is provided in this section. In the context of the publications described above, we also take into account employing blockchain technology for authentication reasons.

In [1] the following methodology was followed; after identifying relevant URLs and HTML elements, a crawler, parser, and data processor were developed using Python 3.10 and the ``request'' and ``beautifulsoup4'' libraries in the PyCharm 2022.2.2 environment. Web scrapers also include error handling mechanisms to prevent potential problems such as connection errors when a URL does not exist. The purpose of this study is twofold. First, they outline the implementation of the WeTMS approach through a case study that creates a dataset containing contextual variables of health tools that can improve social connectedness of older adults in different health jurisdictions in Catalonia, Spain. Second, they analyze these datasets to describe trends in the characteristics and registration of these health assets by local stakeholders.

The web crawler requested 50,000 URLs from websites related to the property and health platform, including 25,000 activities and 25,000

resources, to capture scheme reference numbers between July 2015 and 23 December 2022. Scratched. The "parser" program then parses the HTML code of each existing URL and extracts the required elements, strips the text and automatically saves it to 2 CSV datasets for activities and resources. The encoding of the data set has been changed to avoid a mismatch between the character set used to represent text data and the character set used to traverse text, which could cause some characters to appear as symbols. A preliminary review of discontinued health care measures was performed to exclude irrelevant observations. Before proceeding to the text processing and extraction steps, they filtered out properties that were registered outside of Catalonia or were only aimed at children and young people.

They then use these new datasets to extract properties that may promote social connectedness for older adults and conduct descriptive analysis to explore their characteristics and property registration trends across jurisdictions. This analysis shows the potential application of this approach in project evaluation.

In [2] they are developing MULARS (Multi-Language Review Grabber), a hotel customer review collection tool designed to effectively review multilingual online reviews. In their previous study, they have enabled the search of multilingual data. However, crawling data at scale requires faster speeds and more efficient crawling systems. That's why they were looking for ways to improve the efficiency and effectiveness of the browsing process. They proposed data scraping using multithreading to fasten the information fetching through web scraping. They implemented a multi-threaded crawl system and tested its performance to demonstrate improvements in speed and efficiency. Finally, the article discusses improvements in the use of multithreading, the advantages of

the tool and the possibilities of further application of the tool in various fields.

In [3] was mostly based on privacy concerns that can be handled by web scraping and how that data can be prevented from leakage through scraping. Cookies are ubiquitous on today's web and are an important tool used by website developers, marketers and businesses to improve user experience and obtain useful information. The ethical use of cookies has been scrutinized due to concerns about user privacy and data security. They present a new approach to cookie analysis. The main goal of this paper is to develop a method for analyzing cookies on websites to assess their compliance with privacy regulations, potential security risks, and overall impact on user experience. This article describes how to effectively use web crawlers and scrapers to collect and test cookies.

In [4] the researchers provide more insights of how scrapped data can be useful in real life. First, they examine existing retail applications using network data. Second, they demystify the use of web data by discussing the value of web data in the context of existing retail data sets and larger web data sets to come. Last, they also provide a practical guide to help retailers incorporate web data collection into their research routines. To guide novices in collecting web data, they even developed a mock-up retailing platform called music-to-scrape.org that offers data via web scraping and APIs. They implemented it using libraries such as BeautifulSoup, Scrapy. They even used a LLM version of web scrapers to retrieve data. But it involves higher cost and longer time. They guide on the ways a web scraping infrastructure can be made for the required usage, so that the researchers can use their work as a guideline on which other researchers who want to efficiently set up a web scraping infrastructure for their work can use upon.

In [5] the project tries not only to improve the methods of web extraction related to the raw material (lexical-semantic groups of Russian-sounding verbs, semantically transformed at the level of word formation), but also to systematize the search results in a convenient and easily understandable way. -use the format. is very effective. The main task of this research phase is to collect, summarize (merge the results), analyze and present a summary index of all possible sound verb forms recorded in the electronic version of the Russian dictionary and the State Library, Archive of the Russian Federation. The possibility of combinatorial optimization in the processing of open and closed language databases is especially important when obtaining information from various sources of digital dictionaries (in one or more languages), national language corpora and digital text collections. The results can contribute to the development of various web applications for searching, aggregating and visualizing linguistic materials of different volumes and complexity.

In [6] they specifically meant to show the way web scraping can be useful to geographic researchers and how it seems to be more helpful for them when using it within the legal limitations of the data that they are allowed to scrap. Web scraping faces are more unique challenges, some of which are location related. Ethical and legal issues mainly concern intellectuals Property Rights, Informed Consent and (Geo)Privacy, and Site Integrity. This also affects the practice of open science. In addition, there are technical and statistical problems related to reliability and incompleteness, data inconsistency and biases and limitations on historical reports. In addition, geospatial analysis often requires automated mining and further parsing of place names or addresses (geoparsing, geocoding). A study on apartment rent in Leipzig, Germany illustrated the use of web scraping and its challenges in their study.

In [7] the research is mostly on how a web scraping tool can help in pandemic times like COVID 19. There is a lot of information publicly available on real estate websites. Searching the web for the UK on a daily basis, this article was based on a large database from which we generate timely and highly detailed indicators. One of the original features of this data set is its focus on the supply side of the housing market, which allows for the calculation of innovative indicators that reflect the seller's perspective, such as the number of new listings or price fluctuations of existing listings. during a period of time. Using machine learning to match prices in our data with transaction prices in the notary's database also provides a measure of the buyer's negotiating profit. During the Covid-19 crisis, these indicators indicate a market freeze and a "wait and see" approach for sellers. They also show that stock prices in London continued to fall after the lockdown, while prices rose elsewhere. So web scraping of these data can be used as the training data for the machine learning model for the required prediction.

The purpose of article [8] is to create a web crawler that can collect data from any website and then analyze the data accordingly. For results and analytics, Amazon has implemented web scraping tools that collect product names, prices, number of reviews, ratings and links. The web scraper proved efficient at doing this, going through five pages, analyzing them and visualizing the information in about ten seconds. Therefore, this implementation can be useful for developers who are new to web scraping or researchers who want to reuse code in small data analysis projects. The limitations of this implementation are mainly related to its application to specific product names rather than generic product names, and the retrieval of specific information from the resulting products. Also, BeautifulSoup cannot retrieve all available data, which would slow it down.

3. Problem Definition and Background

In the digital age, the internet serves as a vast repository of information, with millions of websites offering valuable data on a wide range of topics. This information includes product details, market trends, customer reviews, news articles, and more. However, accessing and extracting this data manually from websites is impractical due to the sheer volume and dynamic nature of web content. This is where web scraping comes into play.

Web scraping, also known as web harvesting or web data extraction, is the process of automatically extracting information from websites. It involves accessing web pages, parsing their contents, and extracting relevant data for further analysis. Web scraping enables organizations to gather large amounts of data from multiple sources quickly and efficiently, providing valuable insights and competitive advantages.

The need for web scraping arises because of the inherent complexity of web data. Unlike structured databases, web data is often unstructured or semi-structured, making it difficult to extract and analyze using traditional methods. Extracting relevant data from websites manually can be time-consuming and inefficient. Web scraping allows organizations to solve this problem by automating the data collection process and converting unstructured web data into a structured format that can be easily analyzed and used for decision-making. In marketing, web scraping helps companies collect customer feedback, track social media mentions, and analyze market sentiment.

Web scraping solves this problem by automating the data collection process, allowing users to quickly and efficiently collect, process and analyze large amounts of data from websites. Due to the large volume and

dynamic nature of web content, it is impossible to extract this data manually. Web scraping provides a solution by automating the data mining process, allowing us to collect data efficiently and accurately. Whether monitoring competitors' prices, conducting market research, tracking sentiment analysis, or gathering research data, web crawling provides a powerful tool for extracting valuable insights from the web. Using web scraping technology, companies and researchers can gain a competitive advantage, make informed decisions, and discover hidden trends and patterns in the vast amount of online data.

Web scraping is used in a variety of industries and applications. In e-commerce, businesses use web scraping to monitor competitors' prices, track product availability, and gather customer feedback. In the financial sector, web scraping is used to collect financial data, monitor market trends, and analyze investment opportunities. In marketing, web scraping helps companies collect customer feedback, track social media mentions, and analyze market sentiment.

Although web scraping has its benefits, it also has its challenges. Websites often use methods to prevent scraping, such as CAPTCHA controls, speed throttling, and IP blocking. Additionally, web scraping raises ethical and legal issues such as data privacy and copyright infringement. Web scraping organizations must address these issues responsibly to ensure compliance with laws and regulations. By automating data collection processes, organizations can gain valuable insights, make informed decisions, and stay competitive in today's data-driven world.

4. Requirements analysis and specification

4.1 Hardware Requirements:

CPU: The CPU requirements depend on the number of nodes, the complexity of smart contracts, and the transaction throughput you expect. For a small-scale development and testing environment, a multi-core processor with 4-8 cores is usually sufficient. For larger production networks, more powerful processors with multiple cores may be necessary.

Memory: The amount of RAM you need varies based on your network's size and complexity. For development and testing, a minimum of 8 GB of RAM is good. In production environments, you may need 16 GB or more, especially if your chaincodes and transactions are complex or numerous.

Storage: The storage requirements of the scraped data will be huge. Consider using a cloud provided platform architecture like Amazon S3. If you prefer local storage a simple development environment may require 50 GB of local storage, while production networks may need hundreds of gigabytes or more. Use SSDs for better performance.

Network: A reliable network connection is crucial for Web data procurement, as these networks involve a heavy load to process the scrapping of web data. Low-latency and high-bandwidth connections are desirable for larger data requirements.

Operating System: Linux-based systems are recommended for production use.

Virtualization (if applicable): If you are running the application in a virtualized environment, ensure that the underlying hardware meets the requirements and that the virtualization platform is appropriately configured.

Backup and Redundancy: In production environments, redundancy and backup solutions should be considered to ensure high availability and data recovery.

GPUs (optional): If your use case involves computationally intensive tasks, such as complex cryptographic operations, consider using GPUs to accelerate these processes.

4.2 Software Requirements:

Docker: The orchestration relies on the source code being deployed as a docker image. So, you should install Docker and Docker Compose, which are essential for running the orchestrator.

Python: Python is the core programming language for developing the web scraper. Python provides a rich set of libraries and tools for web scraping, making it the preferred choice for many developers.

Requests: The requests library is essential for making HTTP requests to websites. It simplifies the process of sending requests and handling responses.

ElasticSearch: This package is required for storing the logging metadata. These provide a pythonic way to handle querying in elasticsearch. These are faster and

Boto3(Optional): This package is required for handling services in Amazon Web Services. It may be needed when using S3 for storing the scraped data.

Kubernetes: This package is required for making any kubernetes command pythonically. These are useful for automating the deployment of our solution as a pod in kubernetes which improves the performance of the code being run.

Scraping SDK: These are packages of your choice that supports scraping data from any website, given the tag path where the required data is present. The choice of this SDK is specific to your use case. But choose the scrapper that is legal to be used as they involve handling any private data available on the internet too.

Google Search API: These can be used to make web search on the google web server using google dorking pythonically. These are useful in fetching websites that contain the required product information.

Beautiful Soup: It is a Python library for parsing HTML and XML files. It creates a parse tree from the page source, allowing users to easily extract material. Beautiful Soup provides simple methods and Pythonic idioms to navigate, search, and modify parse trees. It can handle different encodings, parse documents with corruption markers and automatically convert documents to Unicode and UTF-8. In addition, Beautiful Soup supports the parsing of XML files.

Selenium: It is primarily a tool for automating web browsers and is commonly used to test, maintain web content, and automate repetitive tasks. It provides libraries and APIs for interacting with web browsers and supports several programming languages, including Python, Java, and C#. Selenium can simulate user interactions such as clicking buttons, filling out forms, and browsing pages, making it useful for testing web applications by running automated test scripts.

Scrapy: It is a web crawler and web scraping system written in Python. It provides a high-level API for crawling websites and extracting profiles. Scrapy processes requests asynchronously, allowing multiple requests to be executed at the same time, which is very efficient for crawling large sites. It provides built-in support for selecting and extracting data using XPath or CSS expressions. Scrapy also includes a powerful command-line tool to manage scrapy projects, run spiders, and export

scraped data. It supports pipelines to process scraped items, such as storing them in a database or writing them to files.

Pandas: Pandas is a powerful Python open source data manipulation and analysis library. It provides user-friendly data structures such as DataFrame and Series, allowing users to efficiently manipulate and analyze tabular data. With Pandas, users can perform a variety of data operations, including filtering, grouping, merging, and transforming data. Pandas also offers powerful data visualization capabilities that facilitate data exploration and understanding.

Elasticsearch package: The Elasticsearch Python package provides a high-level client for interacting with Elasticsearch clusters. Elasticsearch is a decentralized, RESTful search and analytics engine designed for horizontal scalability, reliability, and instant search capabilities. In Python, the Elasticsearch suite allows users to index and search files, perform aggregations, and perform complex queries against Elasticsearch indexes.

Webdriver-manager: WebDriver Manager is a Python library that helps you manage Selenium web drivers. With WebDriver Manager, users no longer need to manually download and manage web drivers for different browsers. WebDriver Manager automatically downloads the appropriate web driver binaries based on the browser and operating system, ensuring that the correct web driver is available to Selenium. Use a compatible browser. It also helps simplify the process of setting up Selenium automation projects, making it easier for developers to get started with web browser automation.

5. Control Flow Graph (OR) System Design

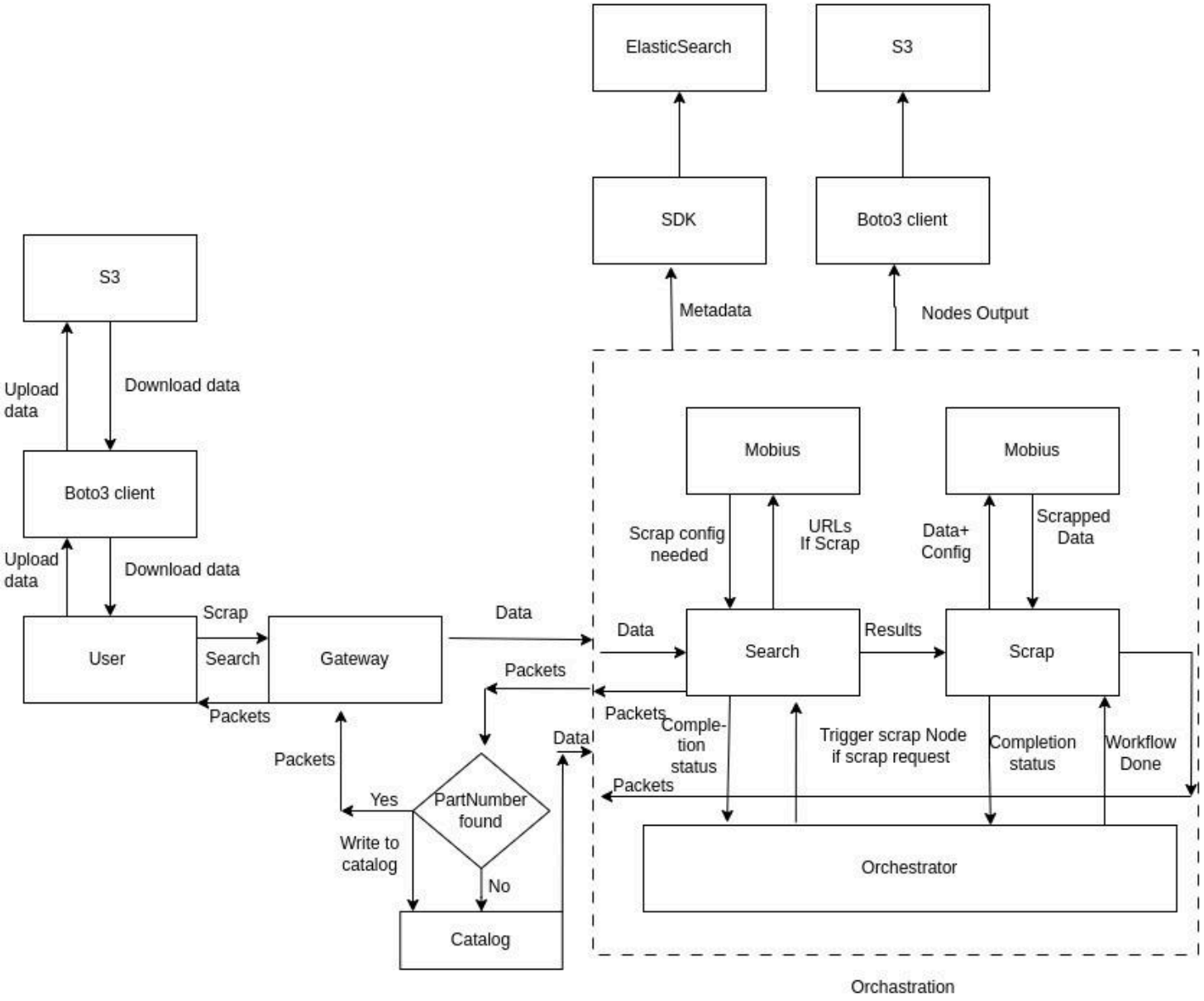


Fig 5.1 Flow diagram

The system architecture consists of two parts – i) The user data supply part, where the user supplies us with product description; ii) The orchestration part where the actual processing of the user supplied data occurs.

The User data supply part consists of providing product details in the form of CSV files. The CSV files consist of mainly two fields which

are Part number, that is the universal model number associated with the product that can be used for uniquely identifying the product. The latter is the description that says a little bit more about the product like the usage that the product benefits on, the specific characteristics of the product etc. This data provided by the client can be in some GigaBytes too as many product details may be required by the client. So there was a need to store and retrieve huge amounts of data when needed and at the same time securely persist it. There are many cloud providers who provide these facilities. If the use case for the application is for an organization that needs to process huge amounts of data and do not have to worry about the infrastructure needed for it and maintaining the data securely without any attack taking place on it, the use of cloud provided storage service can prove to be efficient and costs less too. For this the cloud provider that is chosen by the organization is the Amazon Web Services. They provide a variety of Infrastructure as a Service (IaaS) on top of which we could build our services without the need to think on the infrastructure management and need for it. Amazon Web Services provides a service called Amazon Simple Storage Service (S3). The object storage service Amazon Simple Storage Service (Amazon S3) provides performance, security, scalability, and data availability that are among the best in the business. Any quantity of data can be stored and retrieved at any time, from any location, using Amazon S3. Data is stored by Amazon S3 as objects inside of buckets. A file and any optional metadata that describes it make up an object. You upload the file you wish to store to a bucket in order to store an object on Amazon S3. You can control the object's and associated metadata's permissions when you upload a file.

This CSV can be uploaded directly to the system or can be persisted globally in a S3 bucket so that it can be used across systems. As the use case of this application is for an organization's client the data is stored in the S3 bucket of the organization for us to maintain the data more securely while at the same time retrieve it quickly. This data that is persisted here is provided as part of request where each packet of request contains one record data in its payload and several other data is attached as metadata in the headers so that any packet passed through the system that do not have the required metadata and credentials must not be accepted and processed by the system. These packets cross through a gateway where these requests are forwarded and the metadata checks and credentials verification is made in this part itself so that any invalid or suspicious packet do not have the endpoint to the orchestration part itself and if it is a valid packet the packet is processed by the orchestration part.

The orchestration part contains the actual python scripts that need to be executed on an incoming packet. This serves as the logical abstraction that we provide to the user. They first contain the search part code that makes a web query search on chrome browser internally and the results of the web search are limited by applying google dorking techniques which ensures the results arrived are websites that define the specification of the product given. Google Dorking is a method for finding information on the internet that may not be easily found with ordinary search queries by using sophisticated search operators. This tactic locates particular text strings within search results by utilizing the capabilities of Google's search algorithms. Security experts frequently utilize Google Dorking, which is completely legal, to find weaknesses in their systems. So URL fetching for a given product is handled here.

The second node in the orchestrator setup contains the python script to scrape data from websites of a particular URL when given the config of where the data to be scrapped is available on that website. This part makes use of selenium, which opens up a browser interface based on chromium and gathers the required data based on the input configuration which mentions what details are to be collected. The input configuration contains the metadata like the URL to be scrapped on and also the CSS path to the various parts of interest the client is interested in and also various credentials that the request must possess in order to be considered as a valid and legitimate packet to be considered. If the config isn't present for any particular request, the scraper part cannot process any request that has been made by the user. The processed data that a packet requests from the scraper node has been stored in the S3 bucket so that the processed data can be fetched from the S3 bucket whenever needed by the customer. At the same time the metadata of the request and the processed response packet is stored and indexed in a ElasticSearch deployment handled internally by the organization. This storage of metadata helps in querying the required metadata of the packet that has been processed by the scrapper via the orchestration setup in which it is part of. This helps in debugging the processing done by the scrapper node, so that any error in the request could be identified either as the mistake made by the client side or the attempt was made by any hacker on breaching through the system to use its capabilities for any malicious activities. ElasticSearch supports indexing of any data stored to it which allows it to be queried faster. The ElasticSearch also allows us to make complex querying on the data that it has indexed, so that makes it even more flexible for limiting the data according to the need and can be faster too, making it a suitable choice for us.

The orchestrator present takes care of the communication between both the services. This takes care of the status of each packet given out by these nodes. Orchestrator is the part that has full knowledge of the setup that has been made in the orchestrating environment. So, it is the part that acts as a mediator on any request that happens inside the system. It takes action according to the packet information responded by the nodes. It is the one that invokes the appropriate scripts needed for the use case supplied by the user. Orchestrator also has some predefined checks that we make on the output of a node such that the node is checked whether the node is active and it still has the capability to process the request or is it down due to the heavy workload that has been processed by it. Checks are also made such that any node that is within the orchestrating system might have been compromised by an external attacker by any attack like DDOS. If any error occurs while packet handling, it can take action like retrying on the same packet, if the error persists the orchestrator reports the received status to the user. The error being thrown to the user mostly means to be a part where the request being sent by the user is either conflicting with the setup or the metadata that they have provided aren't good enough for it to be processed.

The results of either search or scrap are stored globally in AWS S3 along with the results sent as packets to the user so that the results could be looked upon later. The metadata of each packet along with the process that handled that packet is stored in the ElasticSearch which can act as the logs on the events happening inside the orchestration scripts. This paves way to debug the issues when an error occurs to a packet and make the system even more efficient and durable.

6. Proposed Approach

The current system has been made up of a system such that there are two solutions that we have in our setup which are a search node that gets the URLs of the product id provided along with its description of how it is useful and a scraping node that gets the data needed in a URL for a user with the attached configs of where the data is it to be extracted in the form of CSS tag paths and these two acts independently . The proposed approach consists of having some prerequisites for scraping ready beforehand by using the search node script that we have so that this can be reused by the scraping process scripts that can increase the time spent by the scraping script on fetching data from websites. Search node could be a prerequisite for scrap as the scrape by default has to make request to the browser using the same Google dorking that the search uses so as to fetch the URLs that are associated with the given product id and description given to it and has to fetch the metadata configs required for it to retrieve data from the websites that they are supposed to take data from. But here we already have a search node solution that can get the scraper node the required URLs that it needs to scrape on and the configs can also be made ready through Mobius for it to search its data on.

For this to work as expected, the web search script must be optimized so that it takes less time and is efficient enough to have completed fetching of required URLs and the config fetching through Mobius has to be made efficient for scraping data from the website available on that URLs. For making the existing search to be efficient, we limit the web search queries that it makes on Google API i.e, previously the search makes four google search calls that were using the site function that the google dorking functionalities provides each one with the various possible combinations of using the product id along with the description

so that all the possible resulted URLs that can be obtained for the given product was shown as the payload of the resulted packet. Now the Google Dorking web search call is made on the given product id and description with only two calls made to the web browser so it significantly reduced the load on the scrape node to make API calls and to wait for it to provide with the requested URLs and at the same time it makes almost the same hits on URLs as before and if possible can also query on historical logs if available on the URLs to be used. This way the results that can be obtained from the scrape script can be faster. There is also a catalog that stores the historical data of product ids that has been already used so that this data stored in the catalog can be reused to fetch the metadata of the website that we search on so that any request that has been already made recently can be fetched directly from the catalog rather than the Google API calls to fetch the URLs request for the given product. Basically the catalog acts as a cache data that the scrape node can rely on so as to fetch its results. Catalog associated with the search node for re-fetching also has a fixed retention period for which the details are considered as valid for it to be relied on and after the retention period the data in the catalog will be removed from the orchestration environment. This retention period for the catalog data storage is needed so that we adhere to the legal constraints of handling the data and at the same time we also do not provide any data that has been recently deprecated, so having this retention period is essential for us to have an accurate at the same time faster information retrieval through search nodes. By making it faster, we can have the results of the current search in a queue on given product details and can be prefetched and enqueued in the queue in the order of receiving the requests and that can be utilized by the latter script whenever it finishes its current run on scraping data.

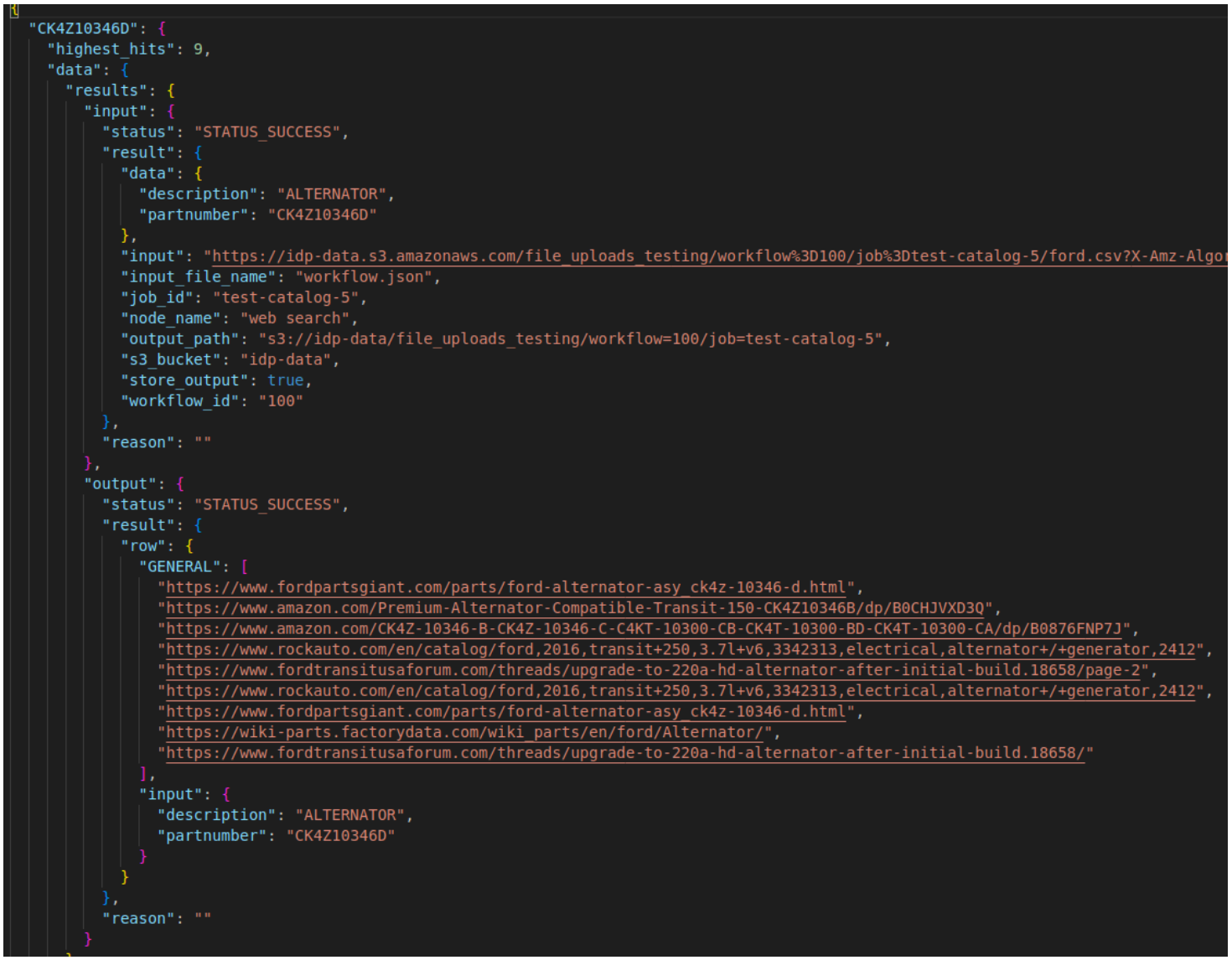
Then the scrap script should be changed in such a way the scrap request packet must be populated with the URLs fetched from the web search script and is placed in the queue that is available in the orchestrator so that the search results are available prior to the scraping node requires those data for scraping the data which is handled by the orchestrator. Now the scrap packet must be hindered from searching any web search call that it has to make for knowing about the website for scraping the data from it. As now the config is fetched from Mobius and the URL is now available for it to work on in the queue , it can process any subsequent requests faster and efficiently than before.

There is also a need to handle failure that the setup can handle so that the setup does not abruptly fail when given a product identification number and description that it could not find details on. This is much like the related results that a google search engine provides to its users. There is a need to provide related suggestions to the client as the result so that they could find something useful for them to work on rather than relying on the client to provide us with the accurate information. So for handling that kind of failover, there is a catalog that updates on every request sent to the orchestration environment so that there are some ground truth values available for us to provide suggestions. If the request given by the client fails due to some product id not found as an issue. We take the most recent product id that has been matching with the description that has been provided by the client so that we provide them the details that is most suitable for their use case while at the same time informing them as a message in the response packets as they are the results of a related product id that has been utilized as the product id provided by the customer was not found. This ensures that the client is also aware of the fact that there is an issue in the product id that they have provided as the request.

7. Implementation:

- **get_chromrdriver_path():** This is the utility that initializes an object that can be used for making web queries in the Google Chrome.
- **get_mcpd_creds():** This is the utility that fetches the credential details that can be utilized in the headers of the request packet of a search call.
- **get_alternate_codes():** This is the utility that searches for the additional product ids that could be tried upon by hitting the catalog stored as part of the search node's instance in a dynamoDB so that these product ids can be used for related searches for the given purpose.
- **get_scouting_output():** This is the utility that uses the Google Chrome API object that has been created and searches with the given searching query input from the main function and returns the URLs that match the given query.
- **get_search_results():** This is the main function that integrates the above utilities. It starts by fetching the required product details provided by the client from S3 using the boto3 client instance package. Then it sets up the chrome driver for web searching through chrome. It then uses the get_alternate_codes utility to get the product identities that are related to the product description given which is used in case of a failure. Then it parses through the arrived packet for its payload and finds the product id and the description that is provided and makes the payload that can be utilized by the get_scouting_output function. It then calls the get_scouting_output function with the prepared payloads in a multi threaded call and the time taken for the completion of this task is

noted to be stored in the ElasticSearch for metadata storage. If there is an error when fetching the results due to the product id could not be made, then the alternate codes that are fetched as related products are called upon and another multithreaded call on get_scouting_output function is made with the updated payload so as to fetch the related results. The results fetched are then written to the S3 bucket where the client could look upon the results that he wants whenever needed. The metadata of the packet received by this node is stored in elastic search too along with the time taken parameter so that the results could be debugged.



```
{
  "CK4Z10346D": {
    "highest_hits": 9,
    "data": {
      "results": {
        "input": {
          "status": "STATUS_SUCCESS",
          "result": {
            "data": {
              "description": "ALTERNATOR",
              "partnumber": "CK4Z10346D"
            },
            "input": "https://idp-data.s3.amazonaws.com/file_uploads_testing/workflow%3D100/job%3Dtest-catalog-5/ford.csv?X-Amz-Algo
            "input_file_name": "workflow.json",
            "job_id": "test-catalog-5",
            "node_name": "web search",
            "output_path": "s3://idp-data/file_uploads_testing/workflow=100/job=test-catalog-5",
            "s3_bucket": "idp-data",
            "store_output": true,
            "workflow_id": "100"
          },
          "reason": ""
        },
        "output": {
          "status": "STATUS_SUCCESS",
          "result": {
            "row": {
              "GENERAL": [
                "https://www.fordpartsgiant.com/parts/ford-alternator-asy_ck4z-10346-d.html",
                "https://www.amazon.com/Premium-Alternator-Compatible-Transit-150-CK4Z10346B/dp/B0CHJVXD3Q",
                "https://www.amazon.com/CK4Z-10346-B-CK4Z-10346-C-C4KT-10300-CB-CK4T-10300-BD-CK4T-10300-CA/dp/B0876FNP7J",
                "https://www.rockauto.com/en/catalog/ford,2016,transit+250,3.7l+v6,3342313,electrical,alternator+/+generator,2412",
                "https://www.fordtransitusaforum.com/threads/upgrade-to-220a-hd-alternator-after-initial-build.18658/page-2",
                "https://www.rockauto.com/en/catalog/ford,2016,transit+250,3.7l+v6,3342313,electrical,alternator+/+generator,2412",
                "https://www.fordpartsgiant.com/parts/ford-alternator-asy_ck4z-10346-d.html",
                "https://wiki-parts.factorydata.com/wiki_parts/en/ford/Alternator/",
                "https://www.fordtransitusaforum.com/threads/upgrade-to-220a-hd-alternator-after-initial-build.18658/"
              ],
              "input": {
                "description": "ALTERNATOR",
                "partnumber": "CK4Z10346D"
              }
            }
          },
          "reason": ""
        }
```


Fig 7.1 Sample response from Search

- **get_scraped_output():** This is the utility that initializes the Mobius scraper object and utilizes the created object to scrape the website data that has been given to it by the main function
- **convert_to_output_format():** This is a utility function that converts the data fetched from the Mobius scrapper into the format that can be sent as a JSON along with the payload of the response.
- **get_scrap_results():** It is the main function of the scraping node. It first parses through the request payload it receives from the user. It looks metadata like product id and the description of the product form the request payload. It then looks for the queue to get the search node's output that it has generated like the URLs to be looked at for this request. The configs that has been passed in the payload i.e, the CSS path on which the website data needs to be looked on is fetched and is changed into the payload format that can be understanded by the Mobius scrapper. Then there is a multithreaded call to get_scraped_output utility that processes the given request to Mobius in parallel through threads. Then the output dictionary from Mobius has been fetched from the get_scraped_output function, then it has been formatted into a JSON payload format that can be sent along with the response packet. The response that has been received is also written to the S3 bucket where it is persisted so that the client can view it at any time. The metadata of the packet received is also saved as an index file into the ElasticSearch so that the metadata of request processed can be queried at ease and at a faster rate. These help in the debugging of the results if required.

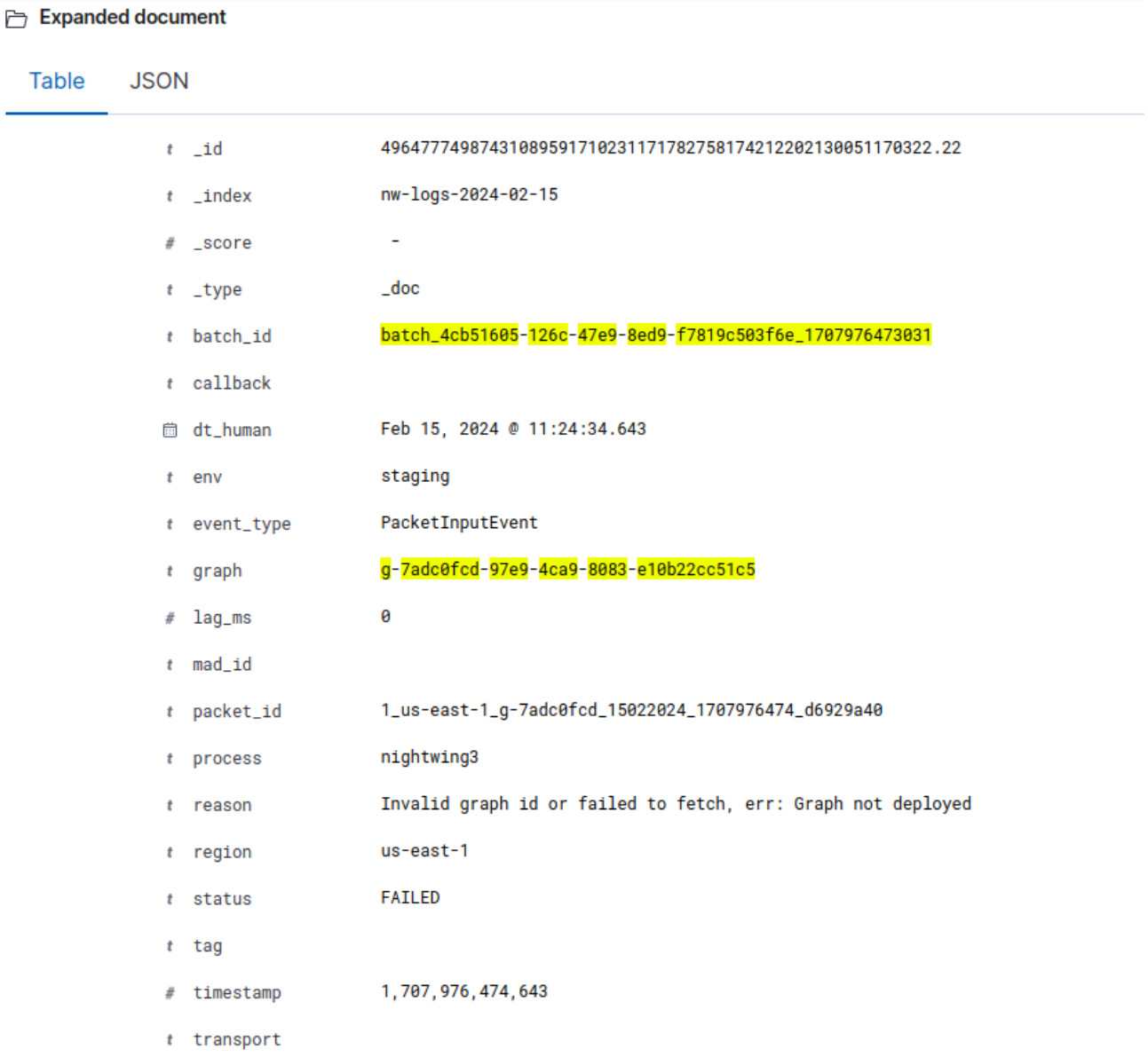


Fig 7.2 Sample ElasticSearch metadata

8. Experiments and Results:

The proposed system is evaluated for real-time performance analysis via programmed modules for various performance metrics. The experimental setup has a controller system with an Intel i5-5200U processor clocked at 2.20 GHz and 8 GB of RAM. The setup consisted of orchestration being made on kubernetes cluster and being connected to S3 bucket via boto3 so as to store the experiment results. There is also an

Elastic search setup that is being accessed through python packages to store the metadata. The orchestration engine was set up in the Amazon Web Services as a EC2 cluster so that the orchestrated environment could be set up on which the experiment could take place.

The experiment was done to make sure that the search query being used provides consistent results. The setup was made such that on a sample data of product identification numbers given as input to the searching code, while reducing the web searching query being used for searching on web browsers for finding the relevant websites for the product details given. It was found that the web search queries can be reduced to a certain level such that the results obtained were almost the same as the one that was found when using all the queries for finding the data on the web. The whole combination of product identification number and description is not needed for fetching results that were almost the same that all combinations provided. It generally required two google dorking search queries that it could search on to find the relevant URLs for the provided products. So this significantly increased the time search code took as there is need for making fewer calls while at the same time having almost the same accuracy in results.

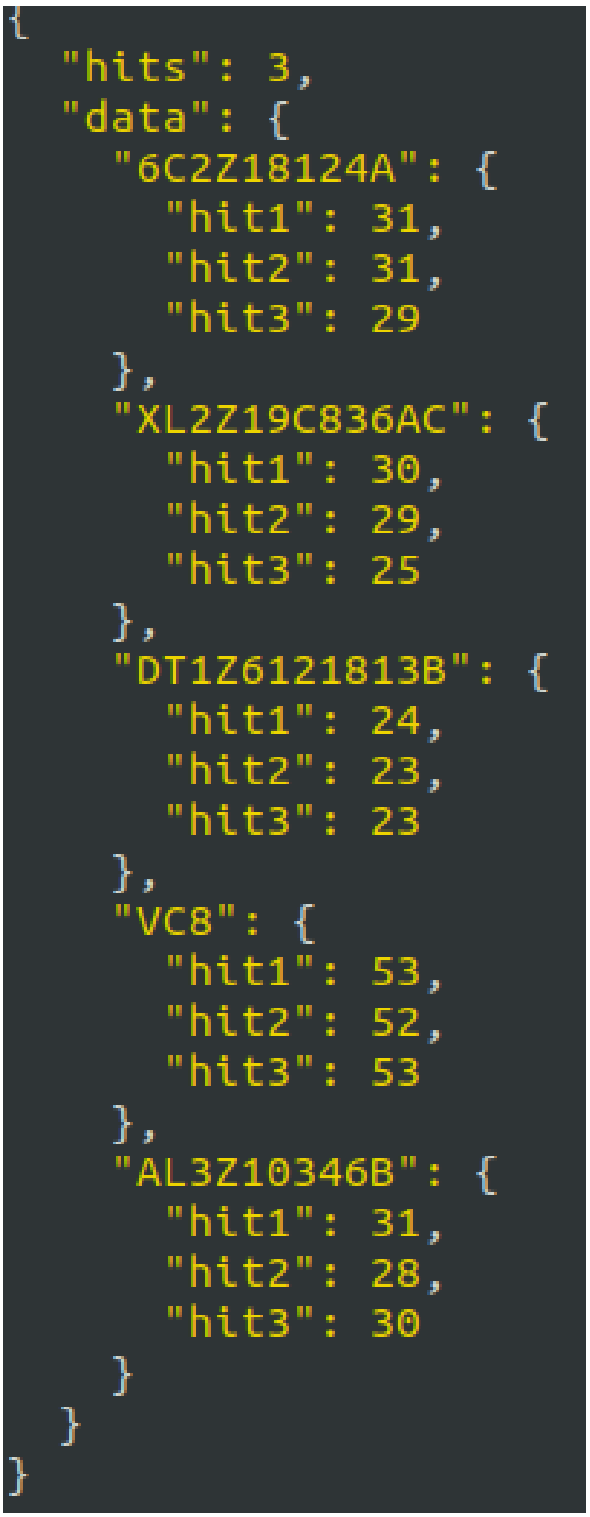


Fig 8.1 Search Hits

Then Scrap code were tested on having been able to fetch the data from the provided websites so that the websites provided by search code is successfully usable by the scrap node and could potentially increase the

efficiency on information fetching by having the URLs to be used for scraping along with the configs for what needs to be fetched be ready prior when a current fetch on a website is done by the scrapper. It is checked by testing whether all the required fields are non empty and are filled with the required data from the applied websites.

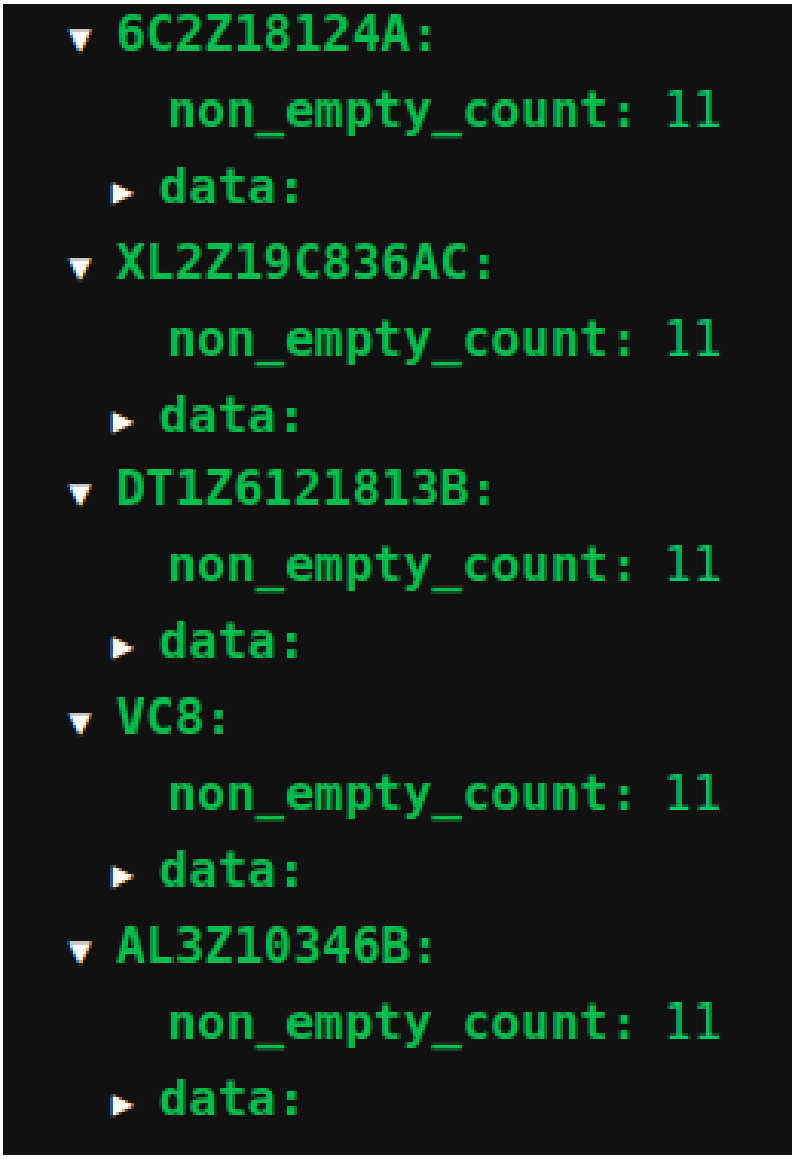


Fig 8.2 Scrap Hits

For logging and debugging purposes, ElasticSearch was used during development. This contains the metadata that a response from the

orchestrator on the given request to it. ElasticSearch allows for easy querying of required information and is also faster.

After making the unit tests on the working of the proposed system, there was an attempt to compare the metrics across the method followed earlier without the prior search and method that has been proposed now. The results seem promising. The time taken by the old approach on searching websites is given in Fig 7.3.

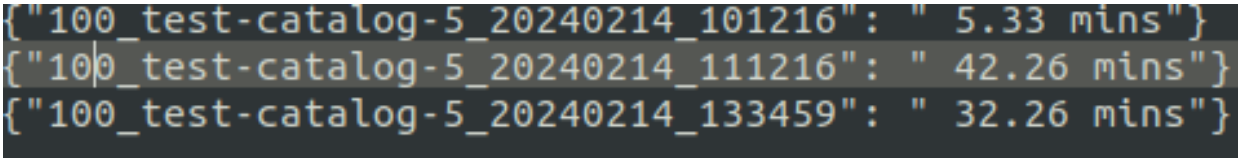


Fig 8.3 Earlier Search metrics

The metrics shown above seemed to be too long for searching URLs even for a small amount of data. The cutting down of search queries made these metrics faster. The metrics obtained when using the proposed way of searching for a product's website is given in Fig 7.4.

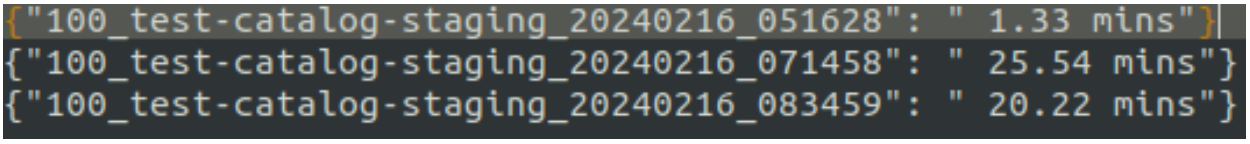


Fig 7.4 Proposed Search metrics

The metrics analysis was continued for the scrapping part too. The scraping part took too long even for a single website scraping in the earlier approach as the website search is done all over again in the old approach. This took significant time for scrapping to even start its task. The metrics that the old scrap code took is given in Fig 7.5.

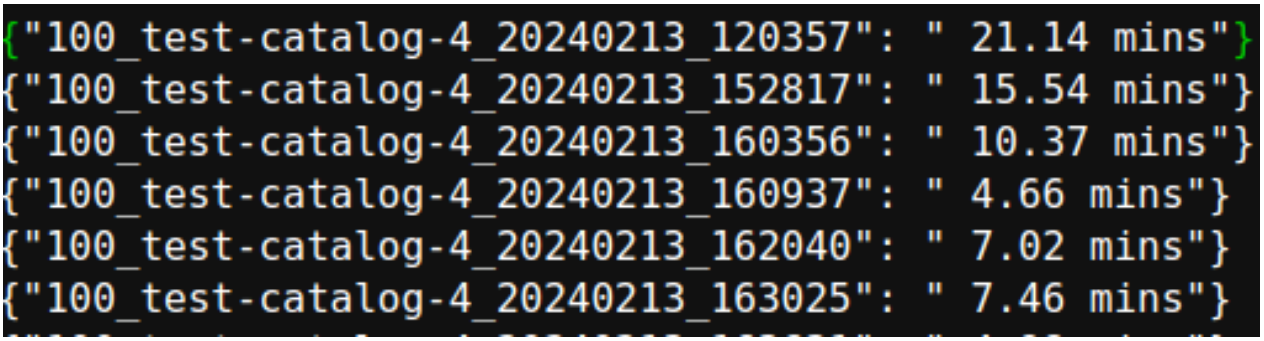


Fig 7.5 Earlier Scrap metrics

The metrics analysis after orchestrating the search code as the prerequisite for the scrap code has made significant improvement in the performance of the scrap code. The improvement was achieved as the URLs to be scrapped on along with the configs needed for scraping that websites such as the tags to look on to for fetching data is available prior in a queue so that the resources have been effectively utilized and the scrap node can act on fetching data from the websites without any delay. With a stronger bandwidth and resources for processing the request, the proposed method can significantly provide quick results. The metrics taken by the newer approach is given in Fig 7.6.

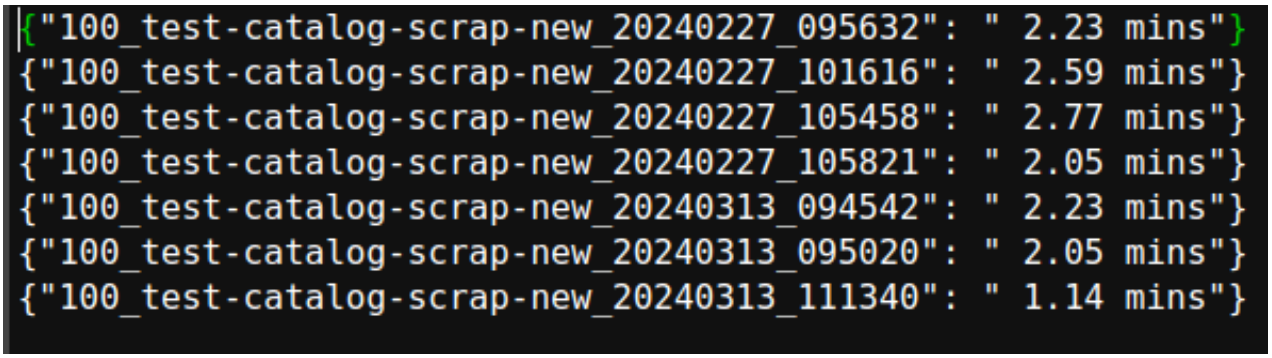


Fig 7.6 Proposed Scrap Metrics

9. Conclusion and Future work

In conclusion, the web scraping project has been successfully implemented using Python, leveraging libraries such as Requests, Boto3, ElasticSearch, Mobius, selenium and scrapy. The project aimed to extract data from websites efficiently and effectively, demonstrating the efficiency of Python and orchestration on web scraping tasks. In addition, it demonstrates the potential of Python for data collection and analysis, opening up opportunities for further research and development in the field of web scraping. By the completion of this project, people were able to use this product as a search engine as well as a scrapper for the needed websites which can handle any unfound data with related suggestions for the user. The project can act as a suitable platform for any other researchers who are interested in processing any data for their research that are available for them in the world wide web.

Moving forward, we can still optimize this product by effectively handling the suggestions part a bit more efficiently and logs can be added as a sidecar task instead of the main nodes handling it which would be handled in the future. We can also implement more advanced data processing techniques to clean and structure the resulting data, ensuring its quality and usability for downstream analysis and also integrate the service with the available data processing open source software which could use this for their processing requests. Indeed, implement safeguards to ensure that web scrapers comply with legal and ethical guidelines, including website terms of service and data protection regulations.

10. References

[1] Galvez-Hernandez P, Gonzalez-Viana A, Gonzalez-de Paz L, Shankardass K, Muntaner C“Generating Contextual Variables From Web-Based Data for Health Research: Tutorial on Web Scraping, Text Mining, and Spatial Overlay Analysis” JMIR Public Health Surveill 2024 https://doi.org/10.2196/50379

[2] Nariman, D. (2024). Enhancing Effectiveness and Efficiency of Customers Reviews Data Collection Through Multithreaded Web Scraping Approach. In: Barolli, L. (eds) Advanced Information Networking and Applications. AINA 2024. Lecture Notes on Data Engineering and Communications Technologies, vol 200. Springer, Cham. https://doi.org/10.1007/978-3-031-57853-3_24

[3] G. P. Bhatraju, C. Vikas and G. Saranya, "Cookie Analysis Using Web Crawling and Web Scraping," 2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2024, pp. 148-153, doi: 10.1109/Confluence60223.2024.10463260.

[4] Jonne Y. Guyt, Hannes Datta, Johannes Boegershausen, Unlocking the Potential of Web Data for Retailing Research, Journal of Retailing, Volume 100, Issue 1, 2024, Pages 130-147, ISSN 0022-4359, https://doi.org/10.1016/j.jretai.2024.02.002 .

[5] BIO Web Conf.Volume 102, 2024,70th Scientific Conference with International Participation “FOOD SCIENCE, ENGINEERING AND TECHNOLOGY – 2023” https://doi.org/10.1051/bioconf/202410203008

[6] Alexander Brenning, Sebastian Henn, “Web scraping: a promising tool for geographic data acquisition” arXiv:2305.19893v1 [cs.IR] 31 May 2023, doi: https://doi.org/10.48550/arXiv.2305.19893

[7] Jean-Charles Bricongne, Baptiste Meunier, Sylvain Pouget, “Web-scraping housing prices in real-time: The Covid-19 crisis in the UK”, Journal of Housing Economics, Volume 59, Part B, March 2023, doi: https://doi.org/10.1016/j.jhe.2022.101906

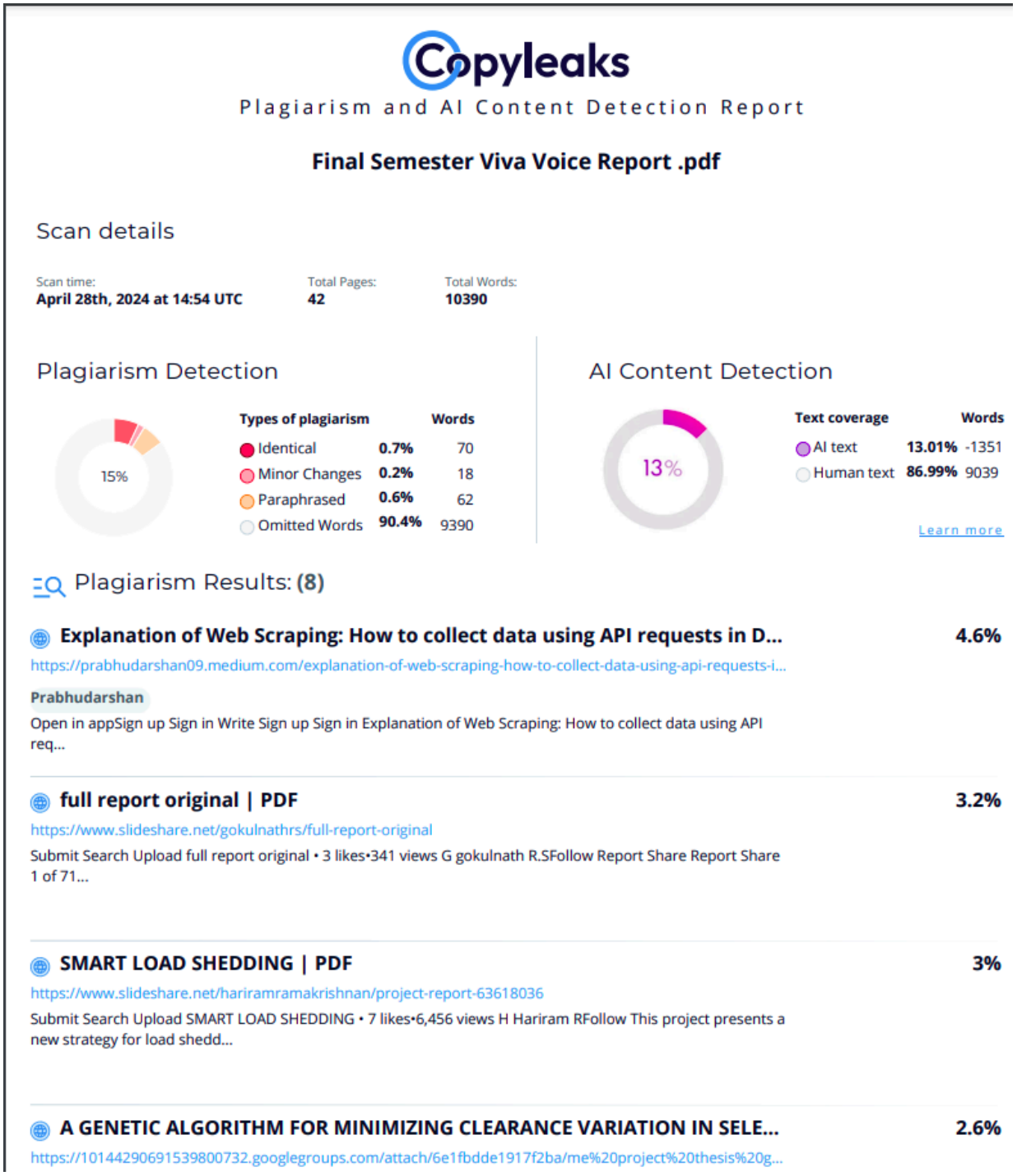
[8] A. Abodayeh, R. Hejazi, W. Najjar, L. Shihadeh and R. Latif, "Web Scraping for Data Analytics: A BeautifulSoup Implementation," 2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia, 2023, pp. 65-69, doi: https://doi.org/10.1109/WiDS-PSU57071.2023.00025

[9]https://www.mobiusservices.com/data-management-services/data-extraction

[10] https://aws.amazon.com/elasticache/redis/

[11] https://docs.celeryq.dev/en/stable/getting-started/introduction.html

11. Plagiarism Report



Copyleaks

Plagiarism and AI Content Detection Report

Final Semester Viva Voice Report .pdf

Scan details

Scan time:	Total Pages:	Total Words:
April 28th, 2024 at 14:54 UTC	**42**	**10390**

Plagiarism Detection

15%

Types of plagiarism		Words
Identical	0.7%	70
Minor Changes	0.2%	18
Paraphrased	0.6%	62
Omitted Words	90.4%	9390

AI Content Detection

13%

Text coverage		Words
AI text	13.01%	-1351
Human text	86.99%	9039

Learn more

Plagiarism Results: (8)

Explanation of Web Scraping: How to collect data using API requests in D... 4.6%

https://prabhudarshan09.medium.com/explanation-of-web-scraping-how-to-collect-data-using-api-requests-i...

Prabhudarshan

Open in appSign up Sign in Write Sign up Sign in Explanation of Web Scraping: How to collect data using API req...

full report original | PDF 3.2%

https://www.slideshare.net/gokulnathrs/full-report-original

Submit Search Upload full report original • 3 likes•341 views G gokulnath R.SFollow Report Share Report Share 1 of 71...

SMART LOAD SHEDDING | PDF 3%

https://www.slideshare.net/hariramramakrishnan/project-report-63618036

Submit Search Upload SMART LOAD SHEDDING • 7 likes•6,456 views H Hariram RFollow This project presents a new strategy for load shedd...

A GENETIC ALGORITHM FOR MINIMIZING CLEARANCE VARIATION IN SELE... 2.6%

https://10144290691539800732.googlegroups.com/attach/6e1fbdde1917f2ba/me%20project%20thesis%20g...

Matched Text 15%

Submitted text (10390 words)

SPEEDING UP INFORMATION COLLECTION BY
COMBINING SEARCH AND SCRAP
A PROJECT REPORT
Submitted by
GUHAN M P S (20C024)
in partial fulfillment for the award of the degree
of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING
THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI - 15
(A Government Aided Autonomous Institution Affiliated to Anna University)
ANNA UNIVERSITY: CHENNAI 600 025
MAY 2024
THIAGARAJAR COLLEGE OF ENGINEERING
MADURAI-15
(A Government Aided Autonomous Institution Affiliated to Anna University)

BONAFIDE CERTIFICATE
Certified that this project report "Speeding up Information collection by combining search and scrap" is the bonafide work of "GUHAN M P S (20C024)" who carried out the project work under my supervision during the

Powered by Copyleaks

Page 1 of 14