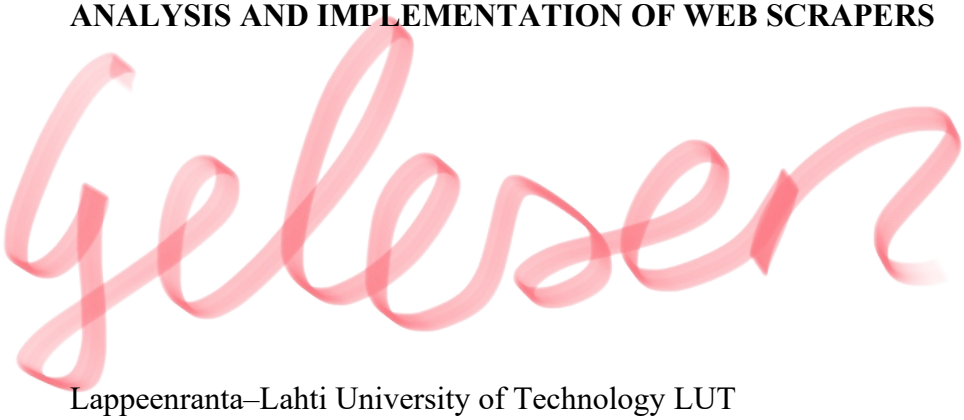
LUT University

ANALYSIS AND IMPLEMENTATION OF WEB SCRAPERS



Lappeenranta–Lahti University of Technology LUT

Bachelor’s Programme in Software and Systems Engineering, Bachelor's thesis

2025

Tom Malinen

Examiners: Associate Professor Antti Knutas

Junior Researcher Majad Qureshi

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Sciences

Software Engineering

Tom Malinen

Analysis and implementation of web scrapers

Bachelor's thesis

2025

35 pages, 7 figures

Examiners: Associate Professor Antti Knutas and Junior Researcher Majad Qureshi

Keywords: web scraping, web scraper, data, data collection tool, API, web crawler, EAN code, database, website, comparison, internet, product, price, review, review average

The creation of the internet was in 1969. Since then, the amount of data available online has reached near unfathomable amounts. Nowadays there are many tools available to gather this information, such as web scrapers.

How could someone implement a web scraper and how does it compare to other data collection tools? This thesis aims to answer the questions through comprehensive literature analysis. In addition, an implemented web scraper will provide hands-on insight into how a web scraper functions in the modern landscape of the internet.

The thesis highlights how web scrapers function and what possible restrictions they face in modern times. With analysis of other works, web scrapers are shown to be more versatile and thorough than other data collection tools but are held back by restrictions placed by website owners. The implemented web scraper successfully shows how to implement a web scraper and gives insight into how one could make their own personal web scraper.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikka

Tom Malinen

Verkonharavointiohjelmien analyysi ja toteutus

Tietotekniikan kandidaatintyö

2025

35 sivua, 7 kuvaa

Tarkastajat: Apulaisprofessori Antti Knutas ja Junior tutkija Majad Qureshi

Avainsanat: verkkoharavointi, verkonharavavointiohjelma, data, datan keräystyökalu, API, hakurobotti, EAN-koodi, tietokanta, verkkosivusto, vertailu, internetti, tuote, hinta, arvostelu, arvostelujen keskiarvo

Internet luotiin vuonna 1969. Siitä lähtien verkossa saatavilla olevan datan määrä on kasvanut lähes käsittämättömään määrään. Nykyään on olemassa monia työkaluja tämän tiedon keräämiseen, kuten verkonharavointiohjelmat.

Kuinka joku voisi aloittaa verkonharavoinnin ja miten se vertautuisi muihin datan keräystyökaluihin? Tämän opinnäytetyön tavoite on vastata kysymyksiin kattavan kirjallisuusanalyysin avulla. Lisäksi toteutettu verkonharavointiohjelma tarjoaa käytännön käsityksen verkonharavointiohjelmien toiminnasta nykyaikaisessa internetin maisemassa.

Tämä opinnäytetyö korostaa, kuinka verkonharavointiohjelmat toimivat ja mitä mahdollisia rajoituksia ne kohtaavat nykyaikana. Muiden akateemisten töiden analyysissä verkonharavointiohjelmat näkyvät monipuolisempina ja perusteellisempina kuin muut tiedonkeräystyökalut, mutta niitä estävät verkkosivujen omistajien asettamat rajoitukset. Toteutettu verkonharavointiohjelma näyttää onnistuneesti, kuinka verkonharavointiohjelma toteutetaan ja antaa käsityksen siitä, kuinka voisi tehdä oman henkilökohtaisen verkonharavointiohjelman.

SYMBOLS AND ABBREVIATIONS

HTTP	Hypertext Protocol
HTTPS	HTTP Secure
URL	Uniform Resource Locator
AI	Artificial intelligence
API	Application Programming Interface
IP	Internet Protocol
WWW	Worldwide web
JS	JavaScript
TOS	Terms of Service

Table of contents

Figures

1 Introduction

1.1 Thesis background

Within the modern landscape of technology, data continues to prove to be one of the most invaluable resources to companies and developers. Almost all modern-day technological inventions and achievements are backed up by mountains of data. One of the most prosperous places to retrieve data is the web. Due to this many different methods have been created to retrieve various data and in return many measures have been put in place to protect data retrieval from being unethical or harmful to others.

According to Statista, there was an estimated 64 zettabytes of data generated in 2020 and 147 zettabytes in 2024 (Taylor, 2024). The growing trend of data allows for more potential applications of data retrieval on the web that help facilitate large billion-dollar companies to individual developers to innovate with their projects using data from the web. The beneficial trade of allowing others to access your data—and accessing data from the web—has created many data retrieval tools and methodologies widely used today, such as web scraping, web crawling, APIs, SQL, and others. With these readily available tools developers have never had an easier time retrieving various and enormous amounts of data for their projects.

It goes without saying that with the growing increase in data in the web by either content creation or companies monitoring user interactions, many laws have been enacted to protect individual users and companies from abusing data retrieval from the web. Many of these laws continue to be updated or revitalized to fit the changing tech landscapes or their respective local laws.

1.2 Thesis objectives

The main purpose of this thesis is to analyse how web scraping has been utilized by various organizations as well as how it can continue to be utilized. How it compares to other data retrieval tools as well is an aspect that will be examined. Web scraping has

proven to be a versatile data retrieval tool that has multiple merits to its usability and efficiency. With the constant growing attention ethical data retrieval has received on the news and by various countries' lawmakers, understanding the tool of web scraping is pivotal to modern day data tech discussions. Lastly, another objective of this thesis is to produce and document a web scraping tool that gathers reviews, prices, availability, etc. of products by their EAN code. The web scraper would showcase the process of implementing a small project web scraper.

This thesis will go over the world of modern-day data retrieval. Its intention is to paint a picture of how such a vast and complex system is handled using modern technologies and techniques. Looking at various sources allows the thesis to explore and clarify modern day data handling. The subject matter is large, so the plan is to condense it into a coherent summarization that focuses on key aspects. The findings will provide insight into how different companies and organizations handle their data, and how data retrieval tools benefit from the modern data landscape.

The implemented web scraper will showcase how a single developer can relatively easily create their own web scraper. The purpose of it is to illustrate the versatility of web scrapers and near boundless opportunities of creating small or large software projects that rely on publicly available data. The complete web scraper tool by the end of the thesis will prove this point, whilst allowing for further analysis on web scrapers. By the end, the thesis will have shown potential worth, and issues presented by web scrapers.

1.3 Thesis organization

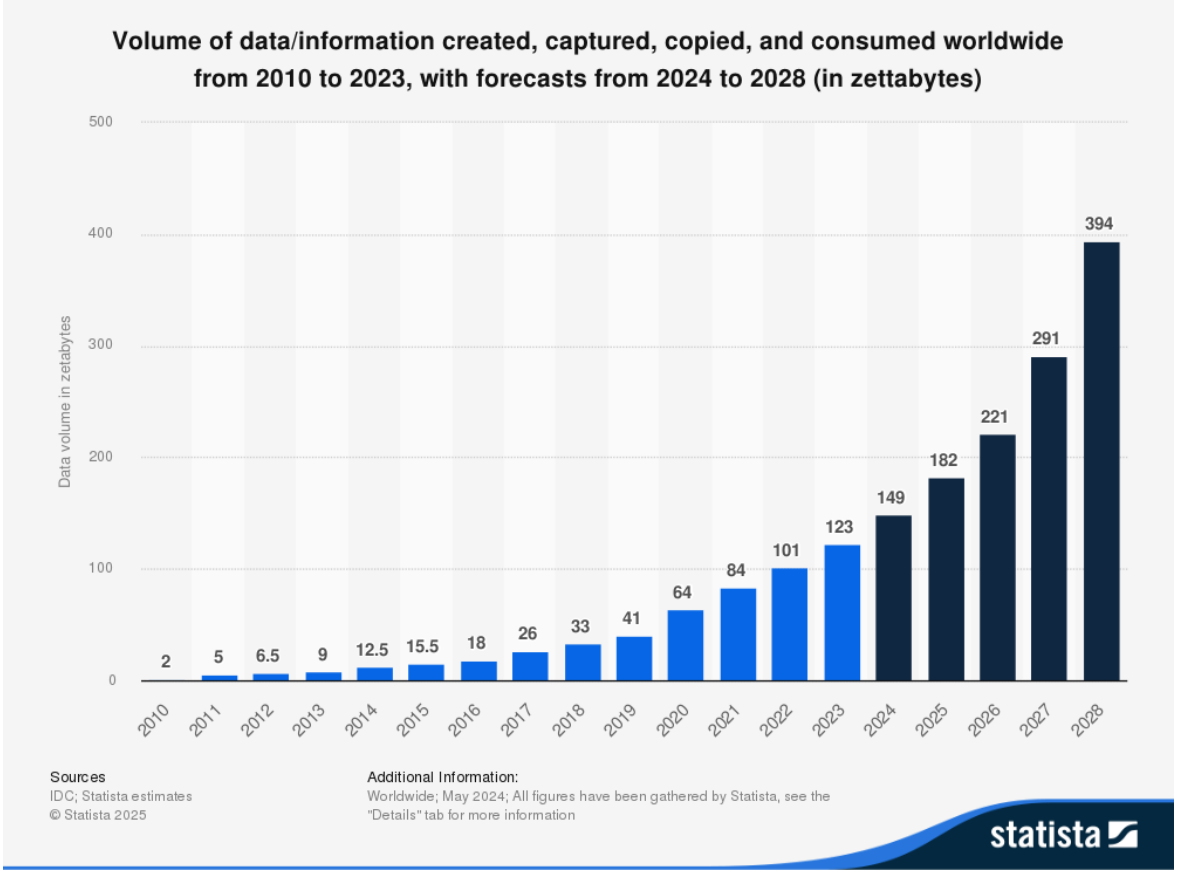
The thesis introduction explains the background, goals, and research questions of the study. The literature review remarks on the previous relevant studies on web scrapers and other data collection tools. The thesis methodology explains the intricacies of developing a web scraper and other questions regarding web scrapers. The Discussion chapter will inspect the completed web scraper, explaining how it functions and anything that could potentially have been done differently. The final chapters of conclusions explain the results of the implemented web scraper and answers the research questions with the information gathered throughout the thesis.

2 Literature review

This chapter looks over previous works done on web scrapers. The reviewed texts will showcase the utility of web scrapers, while looking at its futures and alternatives.

2.1 Reasons to web scrape

The use of web scrapers or alternative web extraction methods are used for a myriad of different data. Many companies are built on their web scraping revenue streams. In Finland websites like vertaa.fi or hintaopas.fi are easy examples of this. On a global scale Google uses Googlebot to scrape for their many operations. Many companies also simply notice the amount of data online and recognize they can use it for their own profits.



(Figure 1: Amount of data created, consumed, and stored 2010–2023, with forecasts to 2028. Statista, 2024.)

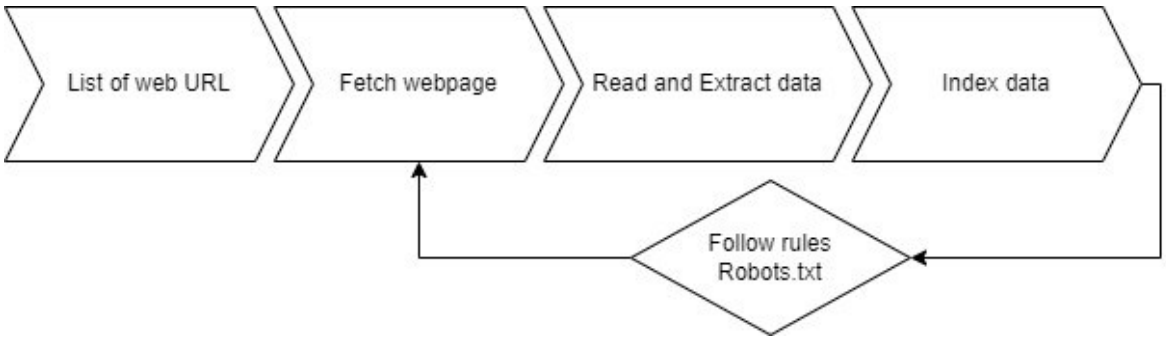
A lot of fields scrape the web for data to improve their own operations. One such field is the medical sector in the US. Web scraping could be used to gather data to analyse the enrichment of a set of genes (Diouf et al., 2019). Tools in the field such as DrugBank, ChemSpider, BioSider, etc. are based on web scraping (Diouf et al., 2019). Google is well known for their use of Googlebot which are utilizes web scraping elements. Online shopping sees the use of many web scrapers to compare prices (Ullah et al., 2018). There are many fields and companies worldwide that make use of web scraping. Search Engines index web pages for search results using web crawlers. SEO Analysis and data mining companies use them to gather public data for research. Web crawling can also be used for archiving as done in the Wayback Machine project. Its consistent use is due to all the benefits that come from proper data collection.

Recently AI companies have been scrutinized for their use of web scraping for improving their algorithms. A news article TechSpot reported that Wikipedia's servers are struggling due to AI scraping the site for data (Maruccia, 2025). This is a common trend nowadays due to the rapid growth of new AI models. Companies with AI models are reliant on big data pools which they acquire from the web. The rise of AI will likely continue to heavily affect bot traffic across the web. Since the rise of AI is recent, there are relatively few studies done on its effect on the worldwide web traffic.

Overall, any sector or business that would benefit from some form of data, likely has web scraping elements or other data extraction methods in place.

2.2 Modern web scraping

One of the most common ways to implement a web scraper is to make it be a part of a web crawler. Web crawlers are automated programs that browse the web URLs to index data. They are primarily used by search engines like Google and store web content for search results. A web crawler script starts with a list of URLs, the crawler then begins with visiting the web stores to fetch web pages by downloading the HTML contents of web page from the stores. The next step then is for the script to read and look for the data (most critical and difficult part). The web crawlers then can index the data and repeat the process. Web crawlers also look for robots.txt file to make sure that the rules are followed. The figure below shows the workflow of a typical web crawler.



(Figure 2: Workflow of typical web crawler)

Web scraping itself is a very popular modern tool for online data extraction. It is one form of data scraping. Data scraping itself is the process of extracting data from online forms with the use of programs (Ullah et al., 2018). Web scrapers persist as tools and have adapted to the modern internet landscape. It has seen improvements in its functionalities and availability to individuals as to businesses.

As bots have become more apparent and harmful for website owners, the restrictions placed on websites has tightened. In 2024 Akamai Technologies reported that 42% of all web traffic is bots. Captchas have continued to improve to negate bots and continue to adapt to more advanced bots. Most website place some form protection from bots/crawlers and utilize robots.txt to reduce bot activity (Lawson, 2015).

With web scraping there are primarily two methods to extract data; one is to mimic human interaction and go the website's appropriate pages by clicking on the correct links like a human would; another is to have the bot get the data directly from the database (Glez-Peña et al., 2013). With the first method, the bot interacts with the site's HTML language. The programmer must recognize the path the bot must take and have it follow the correct path. This is done by inspecting the HTML and locating the variable names leading to the desired data. When the bot loads a page, it searches the HTML for it and does the corresponding programmed action given to the found variable. The second method is more straight forward, as the program must simply do a request to the database through the API (Glez-Peña et al., 2013). Most databases are structured easily to gather any data inside of it with minimal steps. Of the two the second method is quicker and easier though it can be a rare option depending on the sites wished to be scraped and/or what data the web scraper must acquire.

2.2.1 Web scraping techniques

Modern web scraping techniques have three main categories: (i) libraries for general-purpose programming languages, (ii) frameworks and (iii) desktop-based environments (Glez-Peña et al., 2013). Libraries are extensions integrated into a programming language to provide functions for making HTTP requests, parsing HTML, and extracting data. Frameworks are comprehensive tools designed for scalable and automated web scraping, often supporting dynamic content handling and browser automation. Desktop-based environments are GUI-based tools that allow users to scrape data without coding (Glez-Peña et al., 2013).

Those three are the most common data scraping techniques and many other techniques fall beneath one, two, or all of them. According to A Comparative Study on Web Scraping, 9 web scraping techniques were identified (Sirisuriya, 2015). Also 16 different web scraping specific software, each with their own specialities. With web scraping there are numerous ways to collect website data. In 2025 with web data being so useful there are countless communities and businesses that cater to individuals or companies, allowing easy access to web scraping.

How a web scraper also collects from the website has a couple different approaches. Other than the common data collection of reading html and storing the data, a web scraper could in turn use a website's provided sitemap instead. In *Web Scraping with Python,* Lawson (2015, p. 4) summarizes sitemaps concisely as:

> "Sitemap files are provided by websites to help crawlers locate their updated content without needing to crawl every web page."

Sitemaps are useful for most web scrapers and allows website owners navigate how bot traffic travels through their website to help alleviate server load. Sitemaps vary in quality depending on website. When developing a web scraper, it is very useful to first consider if the sitemap should be utilized or not.

Another web scraping method would be to use web crawling. Most modern data collection bots use web crawling. With web crawling developers use their bots to the follow URL links from a list of root links to index the websites. After reaching the necessary pages it then uses a scraper to store any data on the site (Parikh et al., 2018). In general, web scraping is a technique that is often used with other methods to fit the needs of its developers.

There are also many websites that offer their own web scrapers to users. These web scrapers are accessible by signing up or purchase and offer a web scraping tool with varying use-cases (Sirisuriya, 2015). Most of these web scrapers aren't as flexible and universal as one might expect (Sirisuriya, 2015). When deciding to use one of these it is important to recognize its strengths and weaknesses, as to know how well it perform its required tasks.

2.2.2 Web scraping in python

There are a multitude of programming languages available for web scraping building (Lawson, 2015). Python is one of the more accessible ones compared to other languages. With python developers can easily modify their data collection tools (Lawson, 2015).

There are of course also alternatives like Java. Java is more object-oriented in its code structure, allowing for clear coding blocks. The general performance is also higher than python, though for it to be noticeable the task size would have to be quite large.

2.3 Legality of web scrapers

When building any application, especially one that relies on external information, it is important that it follows all relevant laws. Concerning web scrapers relevant laws would be those regarding data protection rights, copyright laws, terms of service, and others that depend on the web scraper's functionality itself. In *Web scraping in Python* by Lawson (2015) describes web scraping being in its "Wild West" stage. 10 years later there still is a lot of grey area regarding the matter. It is important to note that many countries handle laws concerning web scraping very differently to each other. Though, there are some international agreements some countries have agreed upon. Most online data protection laws are handled by a country's own national laws, but cooperation between countries is common and often necessary to govern their respective laws.

Within Europe countries apart of the European Union follow their own agreed laws. With web scraping the EU's decision making is directed by two notable regulations: The General Data Protection Regulation (GDPR) and the Database Directive. GDPR's approach to web

scraping primarily focuses on the individual's privacy and user rights. Some key protections outlined in the GDPR (Regulation (EU) 2016/679) include:

- "Consent is presumed not to be freely given if it does not allow separate consent to be given to different personal data processing operations."
- "The legitimate interests of a controller, including those of a controller to which the personal data may be disclosed, or of a third party, may provide a legal basis for processing, provided that the interests or the fundamental rights and freedoms of the data subject are not overriding, taking into consideration the reasonable expectations of data subjects based on their relationship with the controller."

Concerning data collection, the GDPR recognises the legislative difference between processing for statistical, scientific, and historic research purposes (Altobelli, Forgó, Johnson & Napieralski, 2021). While it sets clear boundaries in certain areas, uncertainty remains in some other areas of data collection (Altobelli, Forgó, Johnson & Napieralski, 2021).

The Database directive focuses more on the rights of creators of databases. It limits how outsiders of the database can access the database if at all. It has many recitals and articles defending database owners from other taking substantial data harming their investments. One such recital is Recital 42 in the Database Directive (Directive 96/9/EC), which has the excerpt: "Whereas the special right to prevent unauthorized extraction and/or re-utilization relates to acts by the user which go beyond his legitimate rights and thereby harm the investment …". The directive clearly defines which databases are protected under law.

Notably from the Database Directive Article 8 explains what databases are allowed to be used. The Database Directive (Directive 96/9/EC) in Article 8, paragraph 1, specifies that "The maker of a database which is made available to the public in whatever manner may not prevent a lawful user of the database from extracting and/or re-utilizing insubstantial parts of its contents, evaluated qualitatively and/or quantitatively, for any purposes whatsoever." The second and third paragraphs further clarify that the user can't use their extracted data to harm the original owner of the database or respective copyright owners of works within the database. This article is one of the vital data rights laws in EU that enable the rightful use of web scrapers on publicly available data.

GDPR and the Database Directive are primarily only applicable to EU countries and companies that provide a service or collect data EU citizens. Again, many countries themselves define their own laws regarding database protection and vary how clearly their laws apply in specific use cases. Notably the US has its own federal laws and state laws that its citizens and companies must abide. That said, an important aspect of these countries' various laws is to remember that the internet is globally accessible. For example, if a user from the US were to use a website based on an EU country. In this case the website would be subject to its local laws and its users' local laws. It is up to the companies and organizations to recognize their user's rights according to their country and follow them; failure to do so may lead to significant legal issues. Most issues regarding privacy and data protection are typically listed in the website's terms of service.

Given the freedom the law allows data collection, web scrapers themselves follow most countries' laws. Profiteering and marketing from other's data can still lead to legal issues. It is important when developing a web scraper or any data retrieval tool to follow all relevant laws surrounding its functionality.

Many companies to protect their sites from scrapers include terms of service (TOS) and robots.txt. If a website has a TOS it often presented on a user's first visit to the site. As any agreement it allows the website owner's outline the conditions for using the website and is legally binding between the provider and its users. The robots.txt is less recognizable for the average user for it's a file specifically outlining the use of web scraping on the website. The read the robots.txt if can be found on this URL format: www.[website].com/robots.txt; an example of this would be www.gigantti.fi/robots.txt. Web scrapers are instructed to follow these for ethical web scraping.

Currently web scraping is considered to be in a grey zone where not enough regulations are in place and companies often don't seek legal action for unethical web scraping. A lot of literature on the legality and ethics web scrapers, draw the conclusion of web data being paradoxical (Krotov, Johnson & Silva, 2020). The Worldwide Web (WWW) was meant to have data be open and accessible, which benefits many companies. However, web data is also very critical for many companies and needs to be protected (Krotov, Johnson & Silva, 2020). With large projects many consider the legal guidelines do not align with the ethical guidelines. Small projects could see their IPs banned to prevent web scraping, only big

projects which cause serious issue to the companies see legal recourse for their unsanctioned web scraping.

Ultimately web scraping has seen more prevalent usage over the years. More and more companies are delving into web scraping, along with AI's popularity which also heavily utilizes web scraping. More recognition on web scraping has caused companies to protect their domains. Undetected web scraping bots tend to severely impact a websites ecommerce due to inflated users amounts and server costs.

2.4 Recent research and future trends

As any technology web scraping has seen new innovations. It's possible applications and methods have been taken in new directions. Recently many have begun using artificial intelligence (AI) to improve web scraping (Weerasinghe, 2024). AI continues to offer a potential improvement most technologies, web scraping being no different. By utilizing AI web scrapers can improve in accuracy, efficiency, and scalability (Weerasinghe, 2024). Web scraping has already seen an improvement in ease of use due to AI and will continue to improve as AI gets more advanced (Parikh et al., 2018).

The main alternative to web scraping is APIs. Throughout the years more companies and organizations have implemented a free API to access their data (Mullins, 2021). Often when considering building a web crawler or scraper, it is good to consider if there is an API that can be used instead. APIs are quicker and more reliant to when gathering data. The main drawbacks are only that there isn't suitable API, or a created API has restricted or not yet implemented a way to gather some data.

Due to the prevalence of malicious web crawlers, more websites have begun incorporating preventative measures against bots. A big trait of modern web apps is the desired uptime of 100% (Gheorghe, Mihai & Dârdală, 2018). They are designed to usable on any device, be secure, and handle any typical traffic spikes. The desire to ensure good uptime, flexibility, security, and scalability have influenced the methods scraping can be implemented (Gheorghe, Mihai & Dârdală, 2018). This trend will indubitably continue, and more preventative measures will be produced by companies.

Since web scraping does not have the clearest legal guidelines of any practice, there are many publications released on the ethics of web scraping. Many in the field of data collection have prompted the desire for clear ethical guidelines in web scraping (Alim, 2014). Currently many consider the ethical guidelines to be to follow the wishes of the website owners and not overload any servers with too many requests. It happens not too uncommonly when web scraping to consider all ethical aspects (Brewer et al., 2021). It is important to properly consider how an implemented web scraper could negatively affect a website. It is also good to consider how "public" data is before scraping it. Content posted on Twitter, Youtube, and Reddit would be considered "public" data, but use of it without the original creator's consent would still require ethical consideration (Brewer et al., 2021).

2.5 Other data collection methods

Web scraping is only one of many ways to extract information from the web. Until APIs became more apparent, web scraping was the only way to extract information from the web (Khder, 2021). APIs have become one of the more thought of solutions when considering on how to collect data from the web. Generally, a good API will beat a web scraper on speed and ease of use. But web scrapers are more flexible than APIs due to being able to extract from multiple web pages (Dongo, et al., 2020). Web scrapers prove to be a necessity to programmers due to not all websites providing an API. Even the websites that provide an API are restricted what data is available and how often it can be accessed (Dongo, et al., 2020). The advantage of web scrapers is on its ability to be automated and/or programmed (Diouf et al., 2019). Flexibility is a large part of web data collection, which web scrapers cater to better.

AI web scraping is a new continually improving method to scrape the web for data (Gheorghe, Mihai & Dârdală, 2018). Recent advancements in AI have caused shifts in multiple technology fields, web data extraction being one of them. AI can be opted in various ways into web scraping. The easiest way is to use an existing generative AI, like chat GPT, and ask it to gather information from a website. OpenAI's new tools allow the chat GPT to go to any many websites and gather data. Asking it to gather product data from popular Finnish store fronts, it doesn't appear to have any issue with. Testing its efficiency of random

web scraping, Chat GPT didn't appear to have any major issues. Websites that require logins or have strong anti-bot measures, can cause AI to be unable to gather information from.

Another use of AI is to implement it in the web scraper. One important aspect of the future of web scraping is the increased usage of AI (Khder, 2021). Like any web scraper you program it to gather the data but can streamline the tasks more easily by making an AI click the links, gather the data, or analyse the data. Since the tasks are relatively simple most web scrapers with AI can be done with local AIs as opposed to online options. The AI can be prompted to read the HTML, or it can utilize visually read the website, then use the website as a human would. Users with AI can make them learn any new things the user desires, making the AI more efficient over time (Khder, 2021). Over the years, AI has continued to improve rapidly, likely causing its use in web scrapers to become more common and efficient while at it.

Hybrid options are also abundantly available. These are common since they allow more specialization for a company's specific needs. Depending on the size of the project, a hybrid approach may be the best one.

3 Thesis methodology

To properly highlight the merits and disadvantages of web scrapers, a program to gather product data from different Finnish vendors will be made. The program will primarily automate the process a user would go through to compare product information across different popular Finnish stores. This chapter will detail the thought processes into making a web scraper, using the knowledge gathered from the literature review.

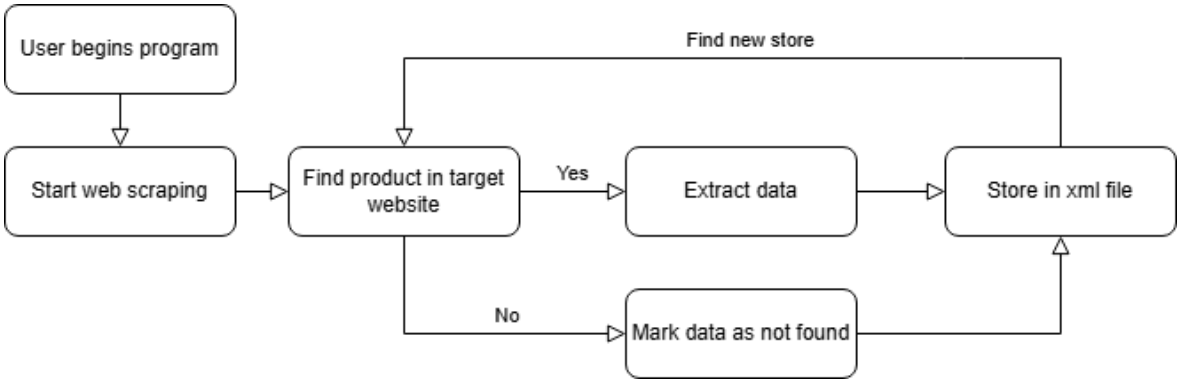
The developed web scraper will utilize a product's EAN code to go into different websites to retrieve the product's price, review score, and number of reviews. EAN code is used to ensure the code can retrieve the correct product from different vendors, despite having names for it. For example: iPhone 16 – 5G älypuhelin 128 GB ultramariini (gigantti) or Apple iPhone 16 128 Gt -puhelin, ultramariini (MYEC3) (verkkokauppa). Without an EAN, across multiple different product names across multiple different stores, the web scraper may inaccurately find or not find a product from the stores.

3.1 How the web scraper will be modelled

The web scraper will establish connections to the vendor's websites through the HTTP/HTTPS protocols. The web scraper will mimic human browsing by searching properly the website with the provided EAN code. It will wait for the site to load and search for the appropriate variables in the website's html to continue until it reaches the correct store page. From there it will gather the necessary data as written in the code, again by searching the correct variables that the coder has already written to look for.

To interact with and gather data from the websites the web scraper will rely on the playwright library. This library allows the program to interact with the websites dynamically, simulating clicking and waiting for things to load. When the program enters the website, it will read the html. It will navigate through the html by either clicking or reading any variable inside a div class. The programmer must first correctly find the correct elements to get to the product by themselves and them add them to the code.

When programming a web scraper, it is vital to know how to navigate the website or web pages. A web scraper is much easier to build if the target websites have been thoroughly examined beforehand. Viewing the structure of the website and understanding which links to follow to product pages are key components of building a web scraper. With the required information gathered, the web scraper code can have a clear path to follow. Once the necessary path is laid out in the code, the program must simply follow the directed path. At the end of the path the data should be available, and the program will read its products details. It will gather the data based on which part of the html code the programmer has marked. The code will be modelled follow the workflow as shown in this figure.



(Figure 3: Workflow of proposed web scraper)

The layout of the web scraper will be of an index html with a JavaScript (JS) file to connect to the backend python. The index and JS file are to provide a nice layout to input the EAN code and notify of any errors. The python backend file can be connected to start the web scraping process then.

3.1.1 Where to get the data from

When considering where and how to extract product information, it is important to consider good ethical practice in web scraping. As a small-scale academic program, the program won't hinder the traffic of the websites so that won't be an issue. It would be good practice to also follow all website's TOS and robots.txt. With that in mind, the sites where product data is collected must first be checked to allow web scraping and only take appropriate data. Also, if there are clear blockades to bots, such as Captcha, then to avoid those sites as well.

The web scrapers primality function is to simply gather a product price, review score, and review amount from a site. To do this limiting the scope of stores to popular electronic vendors allows more cohesion in products comparisons. If the store sells too vastly a selection of products, the program will often struggle to find the EAN code product in both stores. With this in mind, some of the best stores would be Gigantti, Verkkokauppa, Power, and Prisma. All stores sell an up-to-date selection of electronic goods. Gigantti, Verkkokauppa, and Power have robots.txt that discourages web searching with a web scraper. Prisma's robots.txt does allow searching. Since the project is small-scale and has delays in place to allow things to load, the program still can function without causing undue strain on the websites' servers. The program functions as an automation tool and runs on each website on per user input. With that in mind, the program mostly functions as a proof of concept and won't affect their services noticeably.

The web scraper will be set to gather product data from the afore mentioned popular Finnish stores. These stores were chosen due to their popularity and good product availability as electronics retailers. All mentioned stores have up-to-date products when it comes to phones, laptops, home appliances, televisions, cameras, and any other common electronic products.

By using these stores, the web scraper can be modelled to navigate multiple different store's html. Each store has their own unique html code and displays for the products information. Utilizing these stores offers the web scraper utility in showcasing product information from Finland's most popular stores. It also allows the program to showcase how to build a web scraper to handle multiple different sites for data collection.

3.1.2 How to extract information

To begin extracting data on a product, the user must give the program the product's EAN code. This code ensures that when the program goes through different website's product offers, the found product is guaranteed to be the same. With only a name or text description the program may find similar items in the stores but are ultimately not the exact same. An example would be searching the first product result for Iphone 16 Pro Max;

- Gigantti: iPhone 16 Pro Max 5G älypuhelin 256 GB mustatitaani
- Verkkokauppa: Apple iPhone 16 Pro Max 256 Gt -puhelin, arotitaani (MYWX3)

- Power: Apple iPhone 16 Pro Max 256 Gt, arotitaani

Despite having similar products names in their stores, the actual product is slightly different. By using EAN codes the program is ensured to always compare the same products from the different stores.

Once the program has the EAN code it will reach the stores by their website's URL with a search function, such as; https://www.gigantti.fi/search?query={ean_code}. This method will either directly open to the store's product's page or show which products match the EAN code. If it is the latter, then the program will click on the first and only matched product on the list. Though on some of the websites the searches may have to be done on the website's own search bar since their website structure doesn't include URL searches with EAN code. The program will accept to allow cookies, because the program never retains cookies. Every time it is run the pop up will appear and can hinder data gathering if not clicked.

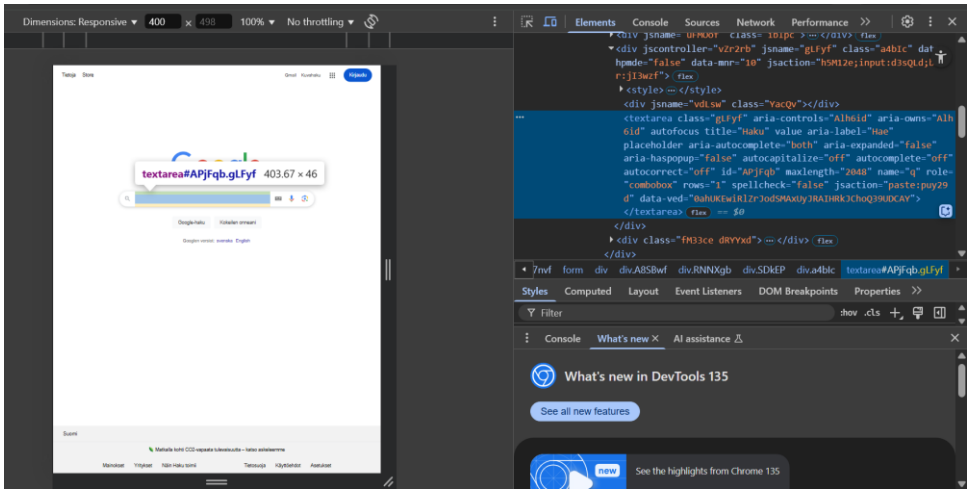
Once it has reached the product page it will read the product's price, reviews, and number of reviews. After collecting the data, it will store it into a csv file. Each website has these items kept in different classes and divs in the html. This will cause the code to have specific classes and areas to look in the store pages depending on the store.

On the Prisma store page a product's price will be in a span class within a div in the html. Here is one example of where the Iphone 16 512GB Musta MYEK3QN/A price is stored on the product page:

 1 099,00 €.

With this information the web scraper can recognize how to extract the data. This process of locating the correct div and classes, making the web scraper either press them or extract their data, is required for each step of the full web scraper. The easiest way to locate the specific data is to right-click on the desired data and choose to inspect. This will open the html code on the button or right of the screen, depending on device, and highlight on the

web page what the code refers to. This can be seen this screen capture of the google.com web page (Figure 4):



(Figure 4: Showcase of inspecting html on browser on google.com)

When reading the data, the program will store in an csv file. The file will contain the EAN code, store name, price, review score, and number of reviews. After completing the retrieval and storage of the data, the program will continue to another store to do the entire process again. This will continue until all stores have been checked, in which case the browser will close, and the program will be waiting on any new commands.

3.1.3 Which tools to rely on

Simply coding everything all the related functionalities required for a web scraper is not needed. To simplify the code, the web scraper will utilize libraries specifically to help collect data.

When viewing which web scraper libraries to utilize two names often show up as reliable libraries, selenium and playwright. Deciding between the two is important when considering what tasks the web scraper must excel at.

Selenium is credited for its quick response, while Playwright is better at mimicking an actual user's movements. For this project Playwright will be used since it has a faster run speed, and the setup is simpler. This web scraping project is only a showcase and handles simple

data collection. The flask library will be used to make the python backend file be reachable through a local port for the JS file.

4 Discussion

This chapter discusses the implemented web scraper. The aim of this chapter is to show how the web scraper succeeded in it tasks, what hurdles presented themselves, and lastly consider different approaches.

4.1 Collecting the data

The web scraper successfully extracted product information from four stores. First it extracted product information from Prisma mimicking how a human would navigate through the site. Then it did the same for Gigantti, Verkkokauppa, and Power.

The user can input the EAN code in the pop-up browser and must wait for the web scraper to collect all data. If there are errors in retrieving the data, an error will be shown in the index page. If there are no issues, the product's EAN code, store, price, review average, and review amount are then stored in an csv file where are product data is captured.

4.2 Restrictions and limitations

Due to using a more static web scraper with no AI, the program was limited in scope. Primarily the web scraper was only able to gather data from the pre-registered stores. To make the program work with any store would require a web crawler and be eased with the use of ML or AI. The intended web scraper would be able to gather much more a data across more websites.

Many websites rely on JavaScript frameworks to load content dynamically. This prevents more traditional scraping techniques to fail, like the BeautifulSoup library. This forced the web scraper to use Playwright or something similar to circumnavigate the issue.

One of the main drawbacks of web scrapers was not avoidable, which is making it future proof. The website's pages will go through structural changes in the future. If these changes affect marked html code chunks, then the program will fail to navigate on the changed

websites. To only way to resolve this issue, with how the web scraper is programmed, is to have as many maintenances as necessary.

The same product has different names, descriptions, or categorization across retailers, making comparison difficult without an EAN code. Currently, the program only functions with the EAN code due to the potential issues of relying on a name or description. This means the user must first find the EAN code.

Since the program is meant to cause to limit its effect on the websites, throttles were places each time the web scraper reached a website. In practice; time.sleep(5) was added after every page.goto() command. This slows the program's efficiency to collect the data, but ensures to minimal load to the servers.

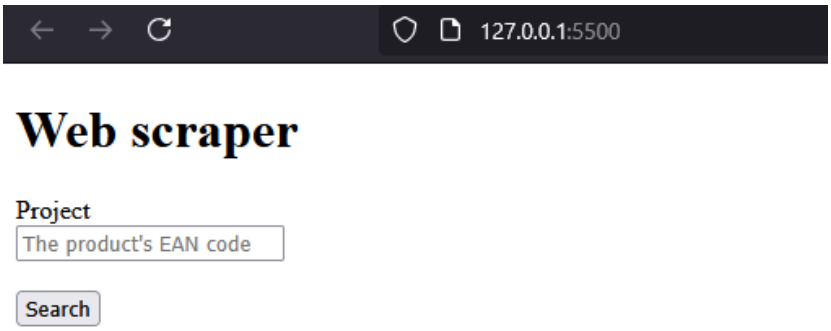
Retailers often update a product's price, and the changes would be difficult to follow. The web scraper is only be able to get the current price given on when the program was run. Any previous sales or upcoming discounts won't be made visible to the user. This functionality would be good to add to the program if it were wanted to be developed further.

Many websites don't allow free form web crawling or scraping. Many site's robots.txt disallow practically all functions or some primary ones. This vastly limits the options for websites for product comparison, if doing a lar-scale projects. Some offer alternatives to their sitemaps, which are perfectly usable in most cases. The web scraper program was attempted to be used on the Gigantti sitemap. It proved to be much more difficult to use since it did organized products by their categories and subcategories. On the main page products could be searched by their EAN code whereas on the sitemap products could not be searched by the EAN code or the name. A program could only search a category and go through every product in the category until they found one that match the EAN code. This method proved to take a too long to find the correct product, so it was not included in the final code.

4.3 Structure of code explained

The complete code is structured into three files. The html file, the JavaScript file, and the python file.

The html files (index.html) is the frontend of the program. It displays where to input the EAN code and shows the products collected information. It is connected to the JavaScript file.



(Figure 5: Frontend of implemented web scraper)

The JavaScript file (script.js) handles getting user inputs and connecting to the python file. It requests the python file to collect the data, and upon receiving the data it uses the html file to display it.

The python file (backend.py) handles all the web scraping logic and data extraction. It receives the requests from the JS file and connects to the websites. Each function handles one necessity such as, getting price/reviews/review number, loading csv file, and saving data. The file reuses the data extraction functions for each website with different html variables as the inputs. If a function can't be used it is clearly stated and replaced in the code with a more unique code for the specific website.

```
EAN Code,Store,Price,Review Average,Number of Reviews
0195949822445,Gigantti,849,4.7,30
0195949822445,Verkkokauppa,849,"4,6",24
0195949822445,Power,849,4.5,103
0617885059919,Gigantti,45,No rating found,0
6438409084880,Gigantti,139,3.5,34
6438409088178,Gigantti,849,No rating found,No reviews found
6438409088178,Verkkokauppa,169,"4,5",26
```

(Figure 6: Final collected data in csv file)

Here is some sample code of the web scraper entering the Prisma site. Similar code was utilized across all websites (Figure 7).

```
store = "Prisma"
            homepage_url = 'https://www.prisma.fi/'
            page.goto(homepage_url, wait_until="domcontentloaded", timeout=50000)

            accept_cookies(page, 'button[data-testid="uc-accept-all-button"]')

            search_input_selector = 'input[data-test-id="search-input"]'
            page.fill(search_input_selector, ean_code)
            page.press(search_input_selector, 'Enter')


            try:
                page.wait_for_selector('li[data-test-id^="products-list-item-"]', timeout=100000)

                product_link_selector = 'li[data-test-id^="products-list-item-"] a[data-test-id="product-
card-link"]'
                page.wait_for_selector(product_link_selector, timeout=100000)

                product_link = page.query_selector(product_link_selector)

                if product_link:
                    product_link.click()
                    time.sleep(5)

                    try:
                        price = page.locator('section[data-test-id="product-price"] span.text-heading-
medium-medium').text_content().strip()
                        price = price.replace('\xa0', '')
                    except:
                        price = "Price not found"

                    update_csv(ean_code, store, price, "No reviews", "No reviews")
```

(Figure 7: Code snippet of Prisma extraction code)

The most difficult part in writing the code was understanding what variables held the price. Once the correct variables were known the rest of the code's programming faced little issues.

5 Conclusions

This thesis was used to see the viability of web scrapers despite them losing popularity.

The implementation of a web scraper proved that producing one tailored for specific user needs is still very plausible today. It can be made to retrieve specifically what the programmer wants. Other alternative data gathering tools like API showcase a more mainstream and easier tool for resource gathering. While web scrapers may need to jump some hurdles to gain their data, API's only hurdle is whether the API provider allows the data to be gathered.

Depending on the circumstance a web scraper may be more appropriate than an API. For small personally projects web scrapers present a versatile tool while for larger projects API's present more usable experience that will not be outdated given some years.

5.1 Other potential methods

There are multiple other ways to implement this type of web scraper.

Typically, one could rely on a website that already handles such product comparison through different websites. In Finland the most well know ones are vertaa.fi or hintaopas.fi. Both offer an ease of use when comparing prices for products between different Finnish retailers. Vertaa.fi gives the price, review, and review average for multiple stores. Hintaopas.fi only gives price comparison. With these the user has a lack of flexibility in their comparisons, compared to a custom web scraper. Both are cemented in price comparison market in Finland with their easy-to-understand systems. Not wanting any specific data or niche store comparison for product information, these websites are adequately sufficient for the average consumer.

To specifically gather data from these stores would require a web scraper or crawler. These stores offer no API support for gathering product price, review average, number of reviews or any other product related info. Verkkokauppa has an API, though it is relegated to streamlining purchases to facilitate ordering within organizational systems.

The code could also be implemented to search with product name's instead of EAN code. This method would allow the code to be more flexible for more vendor websites, though the implementation is tricky. Many websites use different variations to the same product name in their store. Searches would have to be done with keywords, but this poses its own issue. If the keywords are kept strict that searches are quicker, it is also more likely to incorrectly not be able to find the product in the store. While if the keywords are kept too lax searches are more likely to correctly find if the store has the product but the time to check would increase quite a lot. This method can be implemented with careful coding or with using AI to recognize with product listing is the same as another store's.

To implement the same web scraper, a programmer could also use an AI chatbot to view the store's listing of the same products. Doing tests with the EAN code of a product and the prompt to find the products' price, review average and review amount, Chat GPT managed to do good comparisons in a couple of seconds. This an easy way to automate the process of comparing product listing and can accommodate any other needs of the user, due to the AI's current advancements in web searches.

5.2 Summary

The goal of this thesis was to see how web scraping compares to other data retrieval tools. How can a web scraper be implemented. The thesis's objective was to answers these by showcasing the how web scrapers compare in previous data retrieval comparisons and make a web scraper that goes through some of Finland's most popular retailers.

Searching the utilities of all manner of modern data retrieval tools revealed web scrapers to continue to be an integral part of data retrieval. For most large-scale projects a web scraper is a good addition to data retrieval when paired with others. For more individual or small-scale projects a web scraper can easily get you de desired data.

The implemented web scraper succeeded in collecting the required data from the predetermined websites. The program allows the user to go through some of Finland's most popular retailer of technology products and compare the prices, reviews, and review numbers. It shows what is possible with automating a web scraper with the limitations that come with only relying on a web scraper.

Web scrapers can be implemented relatively easily now due to a myriad of web scraping libraries and services that allow a programmer much freedom in designing their web scraper. With web scrapers

Comparing web scrapers to other data retrieval tools proved to highlight the significance of web scrapers positives and negatives. With laws and site's own data retrieval instructions, less and less websites allow free data retrieval with web scrapers. Many provide alternative with sitemap but those can be more restrictive than the normal website. But if a site doesn't have API and allows data collection web scrapers are very easy to implement.

API tools prove their utility compared to web scrapers. APIs often are easy to implement and avoid the hurdles must face. Such as waiting for entire website to load instead of the one chunk they require. When APIs are an option, they match a web scrapers data collection capability or often even surpass it.

Web scrapers are proven to continue to be very useful to this day. They can be implemented for many data retrieval projects when utilizing the web. Compared to other data retrieval tools web scrapers have their own merits and won't be fully replaced anytime soon.

References

Alim, S. 2014, “An initial exploration of ethical research practices regarding automated data extraction from online social media user profiles”. *First Monday*, 19(7). [Online]. Accessed: 16/4/2025. Available at: https://firstmonday.org/ojs/index.php/fm/article/view/5382

Altobelli, C., Johnson, E., Forgo, N. & Napieralski, A. 2021, “To scrape or not to scrape? The lawfulness of social media crawling under the GDPR”. In: *Proceedings of the 8th International Conference on Internet Law & Politics (IDP 2021)*. Accessed: 16/4/2025. Available at: https://www.researchgate.net/publication/351227024_To_Scrape_or_Not_to_Scrape_The_Lawfulness_of_Social_Media_Crawling_under_the_GDPR

Akamai Technologies. 2024, “Bots compose 42% of overall web traffic; nearly two-thirds are malicious”. *Akamai Newsroom*. [Online]. Accessed: 16/4/2025. Available at: https://www.akamai.com/newsroom/press-release/bots-compose-42-percent-of-web-traffic-nearly-two-thirds-are-malicious

Brewer, R., Westlake, B., Hart, T. & Arauza, O. 2021, “The ethics of web crawling and web scraping in cybercrime research: Navigating issues of consent, privacy, and other potential harms associated with automated data collection”. In: Lavorgna, A. & Holt, T. J. (eds.) *Researching Cybercrimes: Methodologies, Ethics, and Critical Approaches*. Cham: Palgrave Macmillan, pp. 447–456. [Online]. Accessed: 16/4/2025. Available at: https://doi.org/10.1007/978-3-030-74837-1_22

European Parliament & Council of the European Union. 2016. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. Official Journal of the European Union, L119, 4 May, pp. 1–88. [Online]. Accessed: 16/4/2025. Available at: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng

Diouf, R., Sarr, E.N., Sall, O., Birregah, B., Bousso, M. & Mbaye, S.N. 2019, “Web scraping: State-of-the-art and areas of application”. *2019 IEEE International Conference on Big Data*. Accessed: 16/4/2025. Available at: https://ieeexplore.ieee.org/document/9005594

Dongo, I., Cardinale, Y., Aguilera, A., Martínez, F., Quintero, Y. & Barrios, S. 2020, "Web scraping versus Twitter API: A comparison for a credibility analysis". *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services.* pp. 263–273. Accessed: 16/4/2025. Available at: https://dl.acm.org/doi/10.1145/3428757.3429104

European Parliament & Council of the European Union. 1996. *Directive 96/9/EC of 11 March 1996 on the legal protection of databases*. Consolidated version 06.06.2019. Accessed: 16/4/2025. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A01996L0009-20190606

Gheorghe, M., Mihai, F.-C. & Dârdală, M. 2018, "Modern techniques of web scraping for data scientists". *Revista Română de Interacţiune Om-Calculator*, 11(1), pp.63–75. Accessed: 16/4/2025. Available at: https://rochi.utcluj.ro/rrioc/articole/RRIOC-11-1-Gheorghe.pdf

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M. & Fdez-Riverola, F. 2013, "Web scraping technologies in an API world". *Oxford academic*. Accessed: 16/4/2025. Available at: https://academic.oup.com/bib/article/15/5/788/2422275

Khder, M.A. 2021, "Web scraping or web crawling: State of art, techniques, approaches and application". *International Journal of Advanced Soft Computing Applications*, 13(3). [Online]. Accessed: 16/4/2025. Available at: http://www.i-csrs.org/Volumes/ijasca/2021.3.11.pdf

Krotov, V., Johnson, L. & Silva, L. 2020, "Tutorial: Legality and ethics of web scraping". *Communications of the Association for Information Systems*, 47. Accessed: 16/4/2025. Available at: https://digitalcommons.murraystate.edu/faculty/86/

Lawson, R. 2015, "Web Scraping with Python". Packt Publishing. Accessed: 16/4/2025 Via Google Books. Available at: https://books.google.fi/books?id=V_1_CwAAQBAJ

Maruccia, A. 2025, "Wikipedia servers are struggling under pressure from AI scraping bots". *TechSpot*. Accessed: 16/4/2025. Available at: https://www.techspot.com/news/107407-wikipedia-servers-struggling-under-pressure-ai-scraping-bots.html

Parikh, K., Singh, D., Yadav, D. & Rathod, M. 2018, "Detection of web scraping using machine learning". *Open Access International Journal of Science & Engineering,* Volume 3(Special Issue 1). Accessed: 16/4/2025. Available at: https://www.oaijse.com/VolumeArticles/FullTextPDF/208_26.DETECTION_OF_WEB_SCRAPING_USING_MACHINE_LEARNING.pdf

Sirisuriya, S.C.M. de S. 2015, "A comparative study on web scraping". *Proceedings of the 8th International Research Conference*, General Sir John Kotelawala Defence University, Sri Lanka, pp. 135–140. Accessed: 16/4/2025. Available at: http://192.248.104.6/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y

Taylor, P. 2024, "Amount of data created, consumed, and stored 2010–2023, with forecasts to 2028". *Statista*. Accessed: 16/4/2025. Available at: https://www.statista.com/statistics/871513/worldwide-data-created/

Ullah, H., Ullah, Z., Maqsood, S. & Hafeez, A. 2018, "Web scraper revealing trends of target products and new insights in online shopping websites". *International Journal of Advanced Computer Science and Applications*, 9(6), pp. 427–432. Accessed: 16/4/2025. Available at: https://thesai.org/Downloads/Volume9No6/Paper_58-Web_Scraper_Revealing_Trends_of_Target_Products.pdf

Weerasinghe, K.G.R.S.M., Maduranga, M.W.P. & Kawya, M.V.T. 2024, "Enhancing web scraping with artificial intelligence: A review". *4th Research Symposium of Faculty of Computing 2024, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka*. Accessed: 16/4/2025. Available at: https://www.researchgate.net/publication/379024314_Enhancing_Web_Scraping_with_Artificial_Intelligence_A_Review

Figures

Code

GitHub link:

https://github.com/Trev-ops/Product-comparison-web-scraper