**CHAPTER 1**

**INTRODUCTION**

**Business is changing based on the emergences of the today technology and every people need to buy something else that would be useful in their lives. It is consisted of clothing, foods, places to stay, accessories and so on. Traditional marketing is to go to the market and buy items that are need for us. But most people do not buy the product of the shop that they see first. People go around the entire market and check every shop that are selling the products that they need to buy. They are checked suitable price, quality of product, discount, specifications, brand and warranty before they buy.**

**Since the internet and technology are tremendously increasing in every situation. So, most people need to buy laptops to use in their use cases such as studying purposes, work, entertainment, simulations, research, and development. They do not buy a laptop that they see first at the shop that they reach first time. Every people go around and check for suitable products for them. Nowadays, E-Commerce websites are available for buying products. They do not need to go physically located shops. They just need to open a Web Browser and buy it. But there are 24 million of websites that sales laptops all over the world. People can’t find or retrieve the suitable product they need to buy.**

As the Internet has grown tremendously and businesses have become increasingly dependent on data, it is now a compulsion to have access to the latest data on every given subject. Data has become the basis and main point of all decision-making processes whether it’s a business or a non-profit organization. Therefore, web scraping has found its applications in every endeavor of note in contemporary times. It is also becoming increasingly clear that those who will make creative and advanced use of web scraping tools will race ahead of others and gain a competitive advantage [10].

Leveraging web scraping and boosting the prospects is the area of endeavor. It is hard to list everything that can leverage web scraping because its applications are now being explored for almost every known human transaction. Web scraping can contribute in a substantial way. There are numerous applications of web scraping under this segment which can be sub-divided into different categories. This includes “News and Reputation Monitoring”, “Data Journalism and Academic purpose” and “Non-profit and Employment areas” [10].

And finally, the most interesting field is “Search engine for classified sites”. There are websites that serve as a search engine for classification type of laptops. There are many websites that people can buy laptop. Visitors can search the site for specific make and model of laptops. However, it is not possible to manually fetch and extract such data. The system is leveraged to crawl and extract the technical specifications of different laptops. The data is used then on the website for visitors to search and access.

* 1. **Objectives**

**The objectives of the system are as follows**

* To understand how a crawler and search engine works
* To study web scraping use cases and applied areas
* To develop simple web scraping tool for choosing suitable product from e-commerce websites.
* To help people in finding the products that they want in a short period of time.
  1. **Web Crawling and Web Scraping**

The need to scrape websites came with the popularity of the Internet, where everybody shares their content and a lot of data. The first widely known scrapers were invented by search engine developers (like Google or AltaVista). These scrapers go through (almost) the whole Internet, scan every web page, and extract information from it, and build an index that human can search. Everyone can create a scraper. Some of them will try to implement such a big application, which could be new competition to Google or Bing. But narrowing the scope to one or two web pages and extract information in a structured manner is a good manner and get the results exported to a database or structured file (JSON, CSV, XML, Excel sheets).

Nowadays, digital transformation is the new buzzword companies use and want to engage. One component of this transformation is providing data access points to everyone or at least to other companies interested in that data through APIs. With those APIs available, developers do not need to invest time and other resources to create a website scraper. Even though providing APIs is something scraper developers won’t benefit from, the process is slow, and many companies don’t bother creating those access points because they have a website and it is enough to maintain [1].

Data is vital for every e-commerce company. And, they are able to see the data on the competitor’s website. The most important thing is the downloaded data that is in a usable format. Most people would be trying to only copy and paste it manually. However, it is not impossible to do it for large websites with hundreds of pages.

This is where web scraping comes into play. [Web scraping](http://www.prowebscraper.com/blog/the-ultimate-guide-to-web-scraping-for-non-programmers/) is a process of automating the extraction of data in an efficient and fast way via Internet. With the help of web scraping, data can be extracted from any website, no matter how large is the data, to customize storage. Moreover, some websites may have data that cannot be copied and pasted. Web scraping can help to extract that kind of data that are need to retrieve.

That is not enough well. Copy and pasting some of data is easy but how to convert or save it in a format of useful way is considerable thing. Web scraping takes care of that too. When extracting web data with the help of a web scraping tools, the data can be able to save in a format such as CSV and can be able to retrieve, analyze and use the data the way in desired format.

Web scraping simplifies the process of extracting data, speeds it up by automating it and creates easy access to the scrapped data by providing it in a CSV format. In simple terms, web scraping saves from the trouble of manually downloading or copying any data and automates the whole process. Sometimes some projects are needed to extract data from one website to load it into another that is called a migration. There are a lot of use cases where the knowledge of website scraping can be leveraged. Some might be common sense, while others are extreme cases.

* 1. **Overview of the System**

**The system is developed to help the people who are trying to find the laptops for their daily use. There are three main choices for user: brand, price and specifications. The user will get the most relevant laptop that match their requirements without the need of time consuming for choosing the laptops.**

**Available links are shown that user can buy laptops via officially sales e-commerce websites. And, system allows user to manually crawl the websites from stretch and they can study the process of how web crawler works via step by step explanation links.**

* 1. **Organization of the Thesis**

In this thesis book, there are four chapters.

Chapter 1 presents introduction, objectives of the thesis, field background, and overview of the system.

Chapter 2discusses about theoretical background in detail. It represents TF-IDF and how to use it. And then it explains Web Scraping and how it works.

Chapter 3provides the design of the system, the flow of the system, and the implementation of the system by using simplex method, and then results the output of the system as the benefit profit of the production plan.

Chapter 4describes the conclusion, benefits, limitations and further extensions of the thesis.

* 1. **Summary**

This chapter represents introduction of Web Crawling and Scraping Agent using TFIDF and explains objectives, field background, overview of the system and organization of the thesis. **In this chapter, difficulty of choosing the right product via numerous websites are mentioned and how Web Crawling and Scraping is come into important part is explained. Detailed theory of Web Crawling and Scraping and how to use it will be explained in the next chapter.**

**CHAPTER 2**

# THEORETICAL BACKGROUND

Web crawling as a market segment has come a long way from being an emerging technology to become an integral part of many businesses. Some companies are formed based on crawling and extracting data. Data trsnsform into money when leveraged effectively, as is evident by the successes of price comparison sites, social media monitoring and reputation management companies.

Web crawlers visit web pages, acquire data, and discover new pages from the ‘seed’ pages. Though most people believe that Google was probably the first crawler to crawl the web in its entirety, web crawling as a technology has a rather long and interesting history behind it. Although the initial crawlers could only collect data, modern day web crawlers are much more robust as apart from data collection, and also capable of monitoring web applications for vulnerability and accessibility. The first crawlers were developed for a much smaller web of about 100,000 web pages, but today some of the popular sites alone have millions of pages. But not all crawlers are built for the entire web.

The second-generation crawlers are either focused crawlers or large-scale crawlers. Focused crawlers were site specific, personally customized and relocatable crawlers such as SPHINX and Mercator. Search engine providers such as Google, Lycos and Excite developed crawlers that were capable of global-scale crawling and indexing of data [19].

**2.1 Web Crawling**

A web crawler also known as a robot or a spider is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving; a service provided by e.g., the Internet archive, where large sets of web pages are periodically collected and archived for posterity.

A third use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them. An example would be attributor, a company that monitors the web for copyright and trademark infringements. Finally, web monitoring services allow them clients to submit standing queries, or triggers, and they continuously crawl the web and notify clients of pages that match those queries.

There are five types of web crawlers. They are Incremental Crawler, Focused Crawler, Parallel Crawler, Distributed Crawler and Deep Web Crawler. The implemented web crawlers are traditional incremental Crawler but they are also parallel as support by Scrapy Framework used in this system [8].

**2.1.1 Incremental Crawling**

A traditional crawler, in order to refresh its collection, periodically replaces the old documents with the newly downloaded documents. On the contrary, an incremental crawler incrementally refreshes the existing collection of pages by visiting them frequently; based upon the estimate as to how often pages change. It also exchanges less important pages by new and more important pages. It resolves the problem of the freshness of the pages. The benefit of incremental crawler is that only the valuable data is provided to the user, thus network bandwidth is saved and data enrichment is achieved [9].

2.1.2 Depth First Tree Transversal

The system uses depth first tree transversal in HTML Scraping. The system reorganizes the entire html pages as a tree structure and apply tree transversal algorithms for web scraping. In [computer science](https://en.wikipedia.org/wiki/Computer_science), tree traversal (also known as tree search) is a form of [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal) and refers to the process of visiting (checking and/or updating) each node in a [tree data structure](https://en.wikipedia.org/wiki/Tree_(data_structure)), exactly once. Such traversals are classified by the order in which the nodes are visited.

These searches are referred to as depth-first search (DFS), as the search tree is deepened as much as possible on each child before going to the next sibling. For a binary tree, they are defined as display operations recursively at each node, starting with the root, whose algorithm is as follows:

The general recursive pattern for traversing a (non-empty) binary tree is this: At node N do the following:

* (L) Recursively traverse its left subtree. This step finishes at the node N again.
* (R) Recursively traverse its right subtree. This step finishes at the node N again.
* (N) Process N itself.

These steps can be done in any order. If (L) is done before (R), the process is called left-to-right traversal, otherwise it is called right-to-left traversal. The following methods show left-to-right traversal [8].

Generic tree is to traverse any tree with depth-first search, perform the following operations recursively at each node:

1. Perform pre-order operation.
2. For each i from 1 to the number of children do:
   * 1. Visit i-th, if present.
     2. Perform in-order operation.
3. Perform post-order operation.

Depending on the problem at hand, the pre-order, in-order or post-order operations may be void, or only want to visit a specific child, so these operations are optional. Also, in practice more than one of pre-order, in-order and post-order operations may be required. For example, when inserting into a ternary tree, a pre-order operation is performed by comparing items. A post-order operation may be needed afterwards to re-balance the tree.

* **Pre-order (NLR)**

**If, Pre-order: F, B, A, D, C, E, G, I, H. Check if the current node is empty or null. Display the data part of the root (or current node). Traverse the left sub tree by recursively calling the pre-order function.**

**Traverse the right sub tree by recursively calling the pre-order function. The pre-order traversal is a topologically sorted one, because a parent node is processed before any of its child nodes is done.**

* **In-order (LNR)**

**If, In-order: A, B, C, D, E, F, G, H, I. Check if the current node is empty or null. Traverse the left subtree by recursively calling the in-order function. Display the data part of the root (or current node). Traverse the right subtree by recursively calling the in-order function. In a binary search tree, in-order traversal retrieves data in sorted**

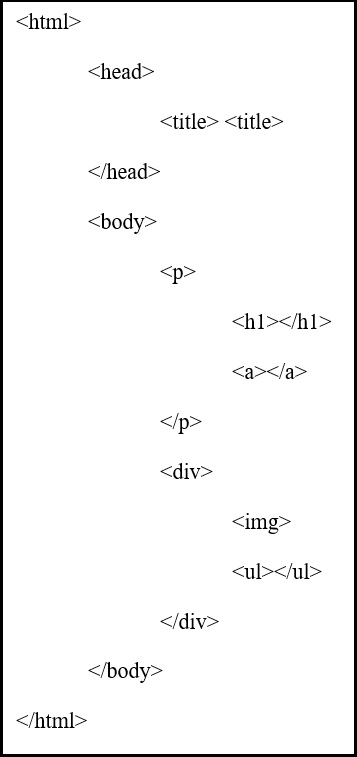
**order.**

* **Out-order (RNL)**

**Check if the current node is empty or null. Traverse the right subtree by recursively calling the out-order function. Display the data part of the root (or current node). Traverse the left subtree by recursively calling the out-order function. In a binary search tree, out-order traversal retrieves data in reverse sorted order.**

* **Post-order (LRN)**

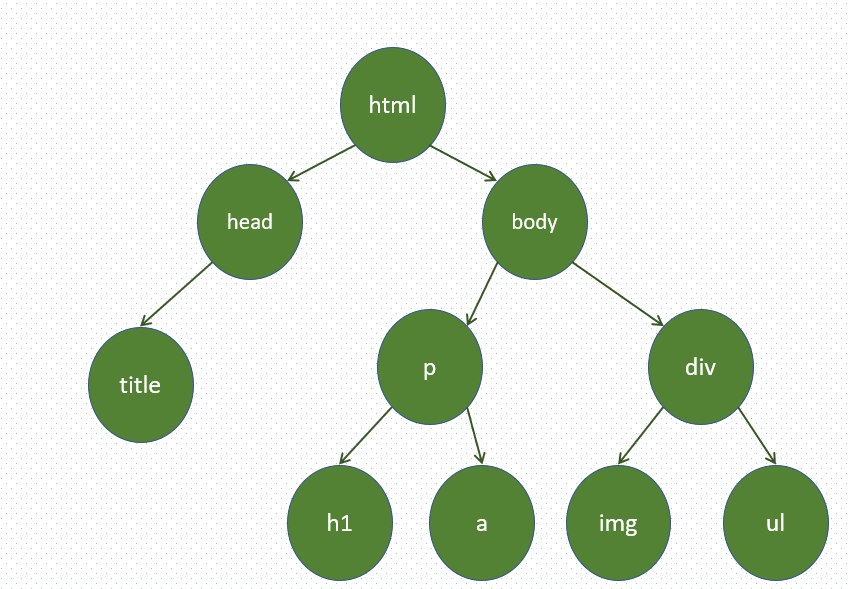
**Post-order: A, C, E, D, B, H, I, G, F. Check if the current node is empty or null. Traverse the left subtree by recursively calling the post-order function. Traverse the right subtree by recursively calling the post-order function. Display the data part of the root (or current node). The trace of a traversal is called a sequentialisation of the tree. The traversal trace is a list of each visited root. No one sequentialisation according to pre-, in- or post-order describes the underlying tree uniquely. Given a tree with distinct elements, either pre-order or post-order paired with in-order is sufficient to describe the tree uniquely. However, pre-order with post-order leaves some ambiguity in the tree structure [9].**



**Figure 2.1 HTML Tags**

**Every html page is transformed into tree diagram so that the system can apply tree Depth-first** Tree Transversal. Starting html tag and closing html tag are combined to one as tree’s node and then transform generic tree to binary tree in which each node has at most two [children](https://en.wikipedia.org/wiki/Child_node), which are referred to as the left child and the right child. A [recursive definition](https://en.wikipedia.org/wiki/Recursive_definition) using just [set theory](https://en.wikipedia.org/wiki/Set_theory) notions is that a (non-empty) binary tree is a [tuple](https://en.wikipedia.org/wiki/Tuple) (L, S, R), where L and R are binary trees or the [empty set](https://en.wikipedia.org/wiki/Empty_set) and S is a [singleton set](https://en.wikipedia.org/wiki/Singleton_set) [9].

Figure 2.1 shows the simple html code with basic tags which means every html tags are transformed into tree node so that tree transversal algorithm can be applied to it. The root node is html tags. Child node of html tag are head and title tags. The child node of head tag are title and child node of body are p and div tags. P is the parent node of a and h1 tags and div tag is parent node of img and ul tags. Figure 2.2 shows the simplified view of transforming html tags info tree structure so that tree transversal can apply to it. The system uses depth first tree transversal method so every transaction goes from top to bottom that is from html tag to head tag and from head tag to title tags.



**Figure 2.2 HTML Tags to Binary Tree**

2.1.3 Visual and Programmatic Crawlers

There are a number of "visual web scraper/crawler" products available on the web which crawl pages and structure data into columns and rows based on the users’ requirements. One of the main differences between a classic and a visual crawler is the level of programming ability required to set up a crawler. The latest generation of "visual scrapers" like [Diffbot](https://www.diffbot.com/), outwithub, and import.io remove the majority of the programming skill needed to be able to program and start a crawl to scrape web data.

The visual scraping/crawling method relies on the user "teaching" a piece of crawler technology, which then follows patterns in semi-structured data sources. The dominant method for teaching a visual crawler is by highlighting data in a browser and training columns and rows. While the technology is not new, for example it was the basis of Needlebase which has been bought by Google (as part of a larger acquisition of ITA Labs), there is continued growth and investment in this area by investors and end-users. This system is implemented as programmatic crawlers which is very useful for nearly every case via Python Scripts.

2.1.4 Crawler Architecture and Identification

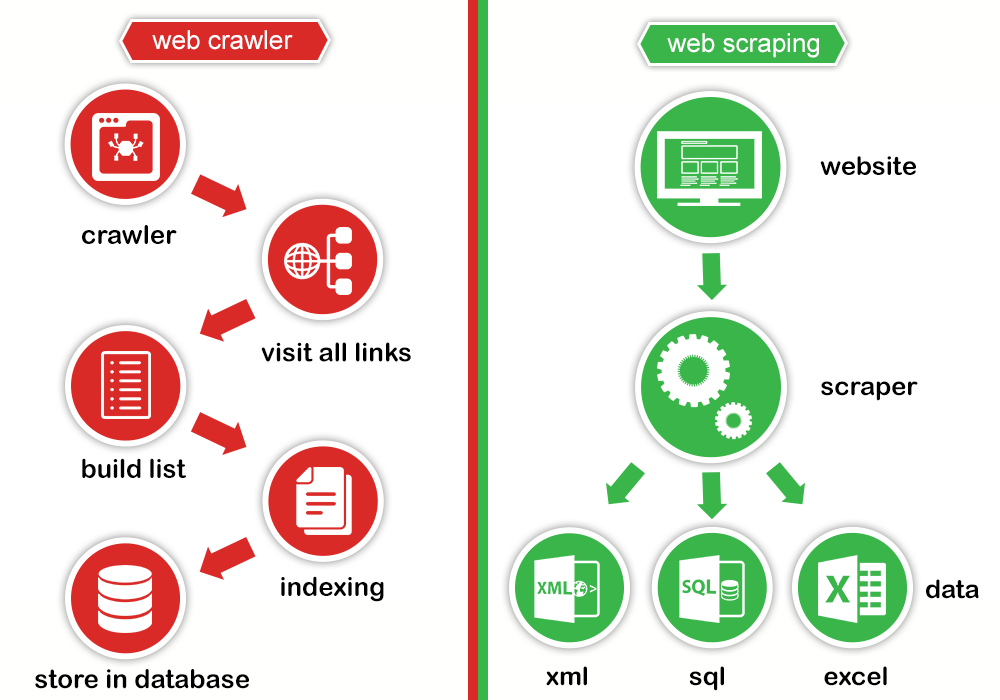
Every crawler in this proposed system is implemented not only to have a good crawling strategy but also to have a highly optimized architecture which can be upgraded into distributed crawling. While it is fairly easy to build a slow crawler that downloads a few pages per second for a short period of time, building a high-performance system that can download hundreds of millions of pages over several weeks presents a number of challenges in system design, I/O and network efficiency, and robustness and manageability.

Web crawlers are a central part of search engines, and details on their algorithms and architecture are kept as business secrets. When crawler designs are published, there is often an important lack of detail that prevents others from reproducing the work. There are also emerging concerns about "[search engine spamming](https://en.wikipedia.org/wiki/Spamdexing)", which prevent major search engines from publishing their ranking algorithms.

Web crawlers typically identify themselves to a Web server by using the [User-agent](https://en.wikipedia.org/wiki/User_agent) field of a [HTTP](https://en.wikipedia.org/wiki/HTTP) request. Web site administrators typically examine their [Web servers](https://en.wikipedia.org/wiki/Web_server)' log and use the user agent field to determine which crawlers have visited the web server and how often. The user agent field may include a [URL](https://en.wikipedia.org/wiki/Uniform_Resource_Locator) where the Web site administrator may find out more information about the crawler. Examining Web server log is tedious task, and therefore some administrators use tools to identify, track and verify Web crawlers. [Spambots](https://en.wikipedia.org/wiki/Spambots" \o "Spambots) and other malicious Web crawlers are unlikely to place identifying information in the user agent field, or they may mask

their identity as a browser or other well-known crawler.

It is important for Web crawlers to identify themselves so that Web site administrators can contact the owner if needed. In some cases, crawlers may accidentally be trapped in a [crawler trap](https://en.wikipedia.org/wiki/Spider_trap) or they may be overloading a Web server with requests, and the owner needs to stop the crawler. Identification is also useful for administrators that are interested in knowing when they may expect their Web pages to be indexed by a particular [search engine](https://en.wikipedia.org/wiki/Web_search_engine). This proposed system is completed all about Web Crawling so that the entire system can be upgraded to Search Engine.



**Figure 2.3 The Difference between Web Crawler and Web Scraper**

Figure 2.3 shows the difference between web crawler and web scraper. Web crawler is used for collecting links, build list, indexing and storing in the database while web scraping is for scrapping off the desired data from each of the links and converted into xml, sql, excel, json or other format that can be used for later analyzing process.

**2.2 Web Scraping**

**The automated data collection from the internet is nearly as old as the internet itself. Although web scraping is not a new term, in years past the practice has been more commonly known as screen scraping, data mining, web harvesting, or similar variations. General consensus today seems to favor web scraping, so that throughout this thesis book, agent refers to programs that specifically traverse multiple pages as web crawlers or refers to the web scraping programs themselves as web bots or web spiders.**

**In theory, web scraping is the practice of gathering data through any means other than a program interacting with an API (or, obviously, through a human using a web browser). This is most commonly accomplished by writing an automated program that queries a web server, requests data (usually in the form of HTML and other files that compose web pages), and then parses that data to extract needed information. In practice, web scraping encompasses a wide variety of programming techniques and technologies, such as data analysis, natural language parsing, and information security. Because the scope of the field is so broad, this thesis book covers the fundamental basics of web scraping and crawling in Chapter 2 and delves into implementation topics in Chapter 3.**

**If the only way accessing the internet is through a browser, probably missing out on a huge range of possibilities. Although browsers are handy for executing JavaScript, displaying images, and arranging objects in a more human-readable format (among other things), web scrapers are excellent at gathering and processing large amounts of data quickly. Rather than viewing one page at a time through the narrow window of a monitor, and can view databases spanning thousands or even millions of pages at once. In addition, web scrapers can go places that traditional search engines cannot. A Google search for “cheapest flights to Boston” will result in a slew of advertisements and popular flight search sites. Google knows only what these websites say on their content pages, not the exact results of various queries entered into a flight search application. However, a well-developed web scraper can chart the cost of a flight to Boston over time, across a variety of websites, and tell the best time to buy the ticket [5].**

**There is a one thing to consider about the data gathering. This is about Application Programming Interfaces (API). APIs can be fantasied, if that suits with the purposes. They are designed to provide a convenient stream of well-formatted data from one computer program to another. There is a lot of API for many types of data to for miscellaneous use, such as Twitter posts or Wikipedia pages. In general, it is preferable to use an API (if one exists), rather than build a bot to get the same data. However, an API might not exist or be useful for the purposes and for several reasons: The system is relatively gathering small, finite sets of data across a large collection of websites without a cohesive API [1].**

**The data is fairly small or uncommon, and the creator did not think it warranted an API. The source does not have the infrastructure or technical ability to create an API. The data is valuable and/or protected and not intended to be widely spread. Even when an API does exist, the request volume and rate limits, the types of data, or the format of data that it provides might be insufficient for the purposes [5].**

**This is where web scraping steps in. With few exceptions, if data can be viewed from browser, that data can be accessed via Python Script. If data can be accessed via script, that data can be stored in a database. Further processing can be done on the data stored in a database.**

**There are obviously many extremely practical applications of having access to nearly unlimited data: market forecasting, machine-language translation, and even medical diagnostics have benefited tremendously from the ability to retrieve and analyze data from news sites, translated texts, and health forums, respectively.**

Other industries or domains where web scraping has found its applications area is “News and Reputation Monitoring”. In order to keep track of the information regarding individuals, products or company, news scraping is quite useful. Web scraping is integral to the process because it allows quick and efficient extraction of data in the form of news from different sources. Such data can then be processed in order to glean insights as required. As a result, it also makes it possible to keep track of the brand and reputation of a company.

Second is “Data Journalism and Academic purpose”. As the name indicates, it is a kind of [journalism](https://datajournalismhandbook.org/1.0/en/getting_data_3.html) that uses data to bolster the news stories. The use of infographics or graphs is a typical example of how data is woven into these stories. The reason why data matters to them a lot is because data provides credibility to the arguments and claims made in the stories. It is also useful as it enables readers to understand complex topics in a visual way. Web scraping comes in handy here because it makes the data available in the first place and enables the journalist to create the impact through the creative use of the data [5] [10].

Academic world depends a lot on data. Academic work revolves largely around one or the other kind of information. Whether it’s a teaching assignment or a research project, academics have to get hold of data and then process it in order to arrive at the necessary insights. Web scraping has now made it extremely easier for them to extract and process the data they need [10].

The field before mentioning the last one is “Non-Profit and Employment

areas”. Even in the case of non-profit organizations, they need data in order to define their mission and further their work. Web scraping tools easily extract the data they need in order to work out their goals and outcomes so that they can forge ahead in their noble projects[10].

Job posters use crawlers to crawl different websites and scrape the information regarding new job postings. It collects information such as job postings, job descriptions, company profiles, and employee profiles. This enables them to provide information regarding job postings and connect job seekers with employers.

### 2.2.1 Web Scraping Techniques

There are many web scraping techniques. They are Text pattern matching, Socket Programming, Vertical Aggregation, Semantic Annotation Recognizing, Hyper Text Markup Language Parsing, Document Object Model Parsing and Computer vision web-page analysis techniques.

Sometimes even the best web-scraping technology cannot replace a human’s manual examination and copy-and-paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up barriers to prevent machine automation.

In text pattern matching, a simple yet powerful approach to extract information from web pages can be based on the UNIX [grep](https://en.wikipedia.org/wiki/Grep" \o "Grep) command or [regular expression](https://en.wikipedia.org/wiki/Regular_expression)-matching facilities of programming languages (for instance [Perl](https://en.wikipedia.org/wiki/Perl) or [Python](https://en.wikipedia.org/wiki/Python_(programming_language))).

For Vertical aggregation techniques, there are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the [Long Tail](https://en.wikipedia.org/wiki/Long_Tail) of sites that common aggregators find complicated or too labor-intensive to harvest content from.

The pages being scraped may embrace [metadata](https://en.wikipedia.org/wiki/Metadata) or semantic markups and annotations, which can be used to locate specific data snippets in Semantic annotation recognizing. If the annotations are embedded in the pages, as [Microformat](https://en.wikipedia.org/wiki/Microformat" \o "Microformat) does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer, are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructions from this layer before scraping the pages [4].

In HTML Parsing Technique, many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template. In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a [wrapper](https://en.wikipedia.org/wiki/Wrapper_(data_mining)). Wrapper generation algorithms assume that input pages of a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme. Moreover, some [semi-structured data](https://en.wikipedia.org/wiki/Semi-structured_data) query languages, such as [XQuery](https://en.wikipedia.org/wiki/XQuery) and the HTQL, can be used to parse HTML pages and to retrieve and transform page content [3].

For Computer vision web-page analysis, there are efforts using [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision) that attempt to identify and extract information from web pages by interpreting pages visually as a human being might [5].

In DOM Parsing, by embedding a full-fledged web browser, such as the [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer) or the [Mozilla](https://en.wikipedia.org/wiki/Mozilla) browser control, programs can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on which programs can retrieve parts of the pages. Among all these Web Scraping Techniques, this system is used HTML Parsing.

2.2.2 HTML Parsing

Parsing; syntactic analysis is the process of analyzing a [string](https://en.wikipedia.org/wiki/String_(computer_science)) of [symbols](https://en.wikipedia.org/wiki/Symbol_(formal)), either in [natural language](https://en.wikipedia.org/wiki/Natural_language), [computer languages](https://en.wikipedia.org/wiki/Computer_languages) or [data structures](https://en.wikipedia.org/wiki/Data_structure), conforming to the rules of a [formal grammar](https://en.wikipedia.org/wiki/Formal_grammar). The term parsing comes from Latin parts (orations), meaning [part (of speech)](https://en.wikipedia.org/wiki/Part_of_speech).

The term has slightly different meanings in different branches of [linguistics](https://en.wikipedia.org/wiki/Linguistics) and [computer science](https://en.wikipedia.org/wiki/Computer_science). Traditional [sentence](https://en.wikipedia.org/wiki/Sentence_(linguistics)) parsing is often performed as a method of understanding the exact meaning of a sentence or word, sometimes with the aid of devices such as [sentence diagrams](https://en.wikipedia.org/wiki/Sentence_diagram). It usually emphasizes the importance of grammatical divisions such as [subject](https://en.wikipedia.org/wiki/Subject_(grammar)) and [predicate](https://en.wikipedia.org/wiki/Predicate_(grammar)).

Within [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) the term is used to refer to the formal analysis by a computer of a sentence or other string of words into its constituents, resulting in a [parse tree](https://en.wikipedia.org/wiki/Parse_tree) showing their syntactic relation to each other, which may also contain [semantic](https://en.wikipedia.org/wiki/Semantics) and other information. Some parsing algorithms may generate a parse forest or list of parse trees for a [syntactically ambiguous](https://en.wikipedia.org/wiki/Syntactically_ambiguous) input.

The term is also used in [psycholinguistics](https://en.wikipedia.org/wiki/Psycholinguistics) when describing language comprehension. In this context, parsing refers to the way that human beings analyze a sentence or phrase in spoken language or text in terms of grammatical constituents, identifying the parts of speech, syntactic relations, etc. This term is especially common when discussing what linguistic cues help speakers to interpret [garden-path sentences](https://en.wikipedia.org/wiki/Garden_path_sentence).

Within computer science, the term is used in the analysis of [computer languages](https://en.wikipedia.org/wiki/Computer_languages), referring to the syntactic analysis of the input code into its component parts in order to facilitate the writing of [compilers](https://en.wikipedia.org/wiki/Compilers) and [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)). The term may also be used to describe a split or separation.

A parser is a software component that takes input data (frequently text) and builds a [data structure](https://en.wikipedia.org/wiki/Data_structure) – often some kind of [parse tree](https://en.wikipedia.org/wiki/Parse_tree), [abstract syntax tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree) or other hierarchical structure, giving a structural representation of the input while checking for correct syntax. The parsing may be preceded or followed by other steps, or these may be combined into a single step. The parser is often preceded by a separate [lexical analyzer](https://en.wikipedia.org/wiki/Lexical_analysis), which creates tokens from the sequence of input characters; alternatively, these can be combined in [scanner less parsing](https://en.wikipedia.org/wiki/Scannerless_parsing). Parsers may be programmed by hand or may be automatically or semi-automatically generated by a [parser generator](https://en.wikipedia.org/wiki/Parser_generator). Parsing is complementary to [templating](https://en.wikipedia.org/wiki/Templating_language" \o "Templating language), which produces formatted output. These may be applied to different domains, but often appear together, such as the [scanf](https://en.wikipedia.org/wiki/Scanf" \o "Scanf)/[printf](https://en.wikipedia.org/wiki/Printf" \o "Printf) pair, or the input (front end parsing) and output (back end code generation) stages of a compiler [2].

The input to a parser is often text in some [computer language](https://en.wikipedia.org/wiki/Computer_language), but may also be text in a natural language or less structured textual data, in which case generally only certain parts of the text are extracted, rather than a parse tree being constructed. Parsers range from very simple functions such as [scanf](https://en.wikipedia.org/wiki/Scanf" \o "Scanf), to complex programs such as the frontend of a [C++ compiler](https://en.wikipedia.org/wiki/C%2B%2B_compiler) or the [HTML](https://en.wikipedia.org/wiki/HTML) parser of a [web browser](https://en.wikipedia.org/wiki/Web_browser). An important class of simple parsing is done using [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), in which a group of regular expressions defines a [regular language](https://en.wikipedia.org/wiki/Regular_language) and a regular expression engine automatically generating a parser for that language, allowing pattern matching and extraction of text. In other contexts, regular expressions are instead used prior to parsing, as the lexing step whose output is then used by the parser [2].

The use of parsers varies by input. In the case of data languages, a parser is often found as the file reading facility of a program, such as reading in HTML or [XML](https://en.wikipedia.org/wiki/XML) text; these examples are [markup languages](https://en.wikipedia.org/wiki/Markup_language). In the case of [programming languages](https://en.wikipedia.org/wiki/Programming_language), a parser is a component of a [compiler](https://en.wikipedia.org/wiki/Compiler) or [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)), which parses the [source code](https://en.wikipedia.org/wiki/Source_code) of a [computer programming language](https://en.wikipedia.org/wiki/Computer_programming_language) to create some form of internal representation; the parser is a key step in the [compiler frontend](https://en.wikipedia.org/wiki/Compiler_frontend). Programming languages tend to be specified in terms of a [deterministic context-free grammar](https://en.wikipedia.org/wiki/Deterministic_context-free_grammar) because fast and efficient parsers can be written for them. For compilers, the parsing itself can be done in one pass or multiple passes – see [one-pass compiler](https://en.wikipedia.org/wiki/One-pass_compiler) and [multi-pass compiler](https://en.wikipedia.org/wiki/Multi-pass_compiler).

The implied disadvantages of a one-pass compiler can largely be overcome by adding [fix-ups](https://en.wikipedia.org/wiki/Relocation_(computing)), where provision is made for code relocation during the forward pass, and the fix-ups are applied backwards when the current program segment has been recognized as having been completed. An example where such a fix-up mechanism would be useful would be a forward GOTO statement, where the target of the GOTO is unknown until the program segment is completed. In this case, the application of the fix-up would be delayed until the target of the GOTO was recognized. Conversely, a backward GOTO does not require a fix-up, as the location will already be known.

Context-free grammars are limited in the extent to which they can express all of the requirements of a language. Informally, the reason is that the memory of such a language is limited. The grammar cannot remember the presence of a construct over an arbitrarily long input; this is necessary for a language in which, for example, a name must be declared before it may be referenced. More powerful grammars that can express this constraint, however, cannot efficiently be parsed. Thus, it is a common strategy to create a relaxed parser for a context-free grammar which accepts a superset of the desired language constructs (that is, it accepts some invalid constructs); later, the unwanted constructs can be filtered out at the [semantic analysis](https://en.wikipedia.org/wiki/Semantic_analysis_(compilers)) (contextual analysis) step.

In [computer science](https://en.wikipedia.org/wiki/Computer_science), top-down parsing is a [parsing](https://en.wikipedia.org/wiki/Parsing) strategy where one first looks at the highest level of the [parse tree](https://en.wikipedia.org/wiki/Parse_tree) and works down the parse tree by using the rewriting rules of a [formal grammar](https://en.wikipedia.org/wiki/Formal_grammar). Left-to-right Leftmost-derivation [parsers](https://en.wikipedia.org/wiki/LL_parser) (LL parsers) are a type of parser that uses a top-down parsing strategy.

Top-down parsing is a strategy of analyzing unknown data relationships by hypothesizing general [parse tree](https://en.wikipedia.org/wiki/Parse_tree) structures and then considering whether the known fundamental structures are compatible with the hypothesis. It occurs in the analysis of both natural [languages](https://en.wikipedia.org/wiki/Language) and [computer languages](https://en.wikipedia.org/wiki/Computer_language).

Top-down parsing can be viewed as an attempt to find [left-most derivations](https://en.wikipedia.org/wiki/Context-free_grammar#Derivations_and_syntax_trees) of an input-stream by searching for [parse-trees](https://en.wikipedia.org/wiki/Parse_tree) using a top-down expansion of the given [formal grammar](https://en.wikipedia.org/wiki/Formal_grammar) rules. Inclusive choice is used to accommodate [ambiguity](https://en.wikipedia.org/wiki/Syntactic_ambiguity) by expanding all alternative right-hand-sides of grammar rules.

Simple implementations of top-down parsing do not terminate for left recursive grammars, and top-down parsing with backtracking may have exponential time complexity with respect to the length of the input for ambiguous CFGs. However, more sophisticated top-down parsers have been created by Frost, Hafiz, and Callaghan which do accommodate ambiguity and left recursion in polynomial time and which generate polynomial-sized representations of the potentially exponential number of parse trees.

A [compiler](https://en.wikipedia.org/wiki/Compiler) parses input from a programming language to an internal representation by matching the incoming symbols to [production rules](https://en.wikipedia.org/wiki/Formal_grammar#The_syntax_of_grammars). Production rules are commonly defined using [Backus-Naur form](https://en.wikipedia.org/wiki/Backus-Naur_form). An [LL parser](https://en.wikipedia.org/wiki/LL_parser) is a type of parser that does top-down parsing by applying each production rule to the incoming symbols, working from the left-most symbol yielded on a production rule and then proceeding to the next production rule for each non-terminal symbol encountered. In this way the parsing starts on the Left of the result side (right side) of the production rule and evaluates non-terminals from the Left first and, thus, proceeds down the parse tree for each new non-terminal before continuing to the next symbol for a production rule.

It would match and attempt to match next. And it would be tried too. As one may expect, some languages are more ambiguous than others. For a non-ambiguous language in which all productions for a non-terminal produce distinct strings: the string produced by one production will not start with the same symbol as the string produced by another production. A non-ambiguous language may be parsed by an LL(1) grammar where the (1) signifies the parser reads ahead one token at a time. For an ambiguous language to be parsed by an LL parser, the parser must lookahead more

than 1 symbol, e.g. LL(3).

The common solution to this problem is to use an [LR parser](https://en.wikipedia.org/wiki/LR_parser), which is a type of [shift-reduce parser](https://en.wikipedia.org/wiki/Shift-reduce_parser), and does [bottom-up parsing](https://en.wikipedia.org/wiki/Bottom-up_parsing).

**2.3 Preprocessing**

**Preprocessing is one of the major steps when dealing with any kind of text models. During this stage, the distribution of the data is checked and, what techniques are needed and how deep to clean are determined.**

**This step never has a one hot rule, and it totally depends on the problem statement. Few mandatory preprocessing is converting to lowercase, removing punctuation, removing stop words and lemmatization/stemming. In problem of system it seems like the basic preprocessing steps may be sufficient.**

**2.3.1 Lowercase** Conversion

**During the text processing each sentence is split to words and each word is considered as a token after preprocessing. Programming languages consider textual data as sensitive, which means that “The” is different from “the”. Humans know that those both belong to same token but due to the character encoding those are considered as different tokens. Converting to lowercase is a very mandatory preprocessing step. In this system, NUMPY method is used which can convert the list of lists to lowercase at once.**

**2.3.2 Stopwords and Tags Removal**

Stop words are the most commonly occurring words which don’t give any additional value to the document vector. Removing stopwords will increase computation and space efficiency. Stop words are the most common words in a language like “a”, “an”, “the”, “to”, “not”. These words do not carry important meaning and are usually removed from texts. It is possible to remove stop words using [Natural Language Toolkit (NLTK)](http://www.nltk.org/), a suite of libraries and programs for symbolic and statistical natural language processing.

NLTK library has a method to download the stopwords, so instead of explicitly mentioning all the stopwords. NLTK library can be used to iterate over all the words and remove the stop words [14].

In tags removal, the proposed system gets all the data from each web pages as

mixture of tags, information and unwanted data. Removing tags and unwanted data are vital for proposed system because the actual processing step are only intended for desire information.

**2.3.3 Ngrams**

In the fields of [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) and [probability](https://en.wikipedia.org/wiki/Probability), an n-gram is a contiguous sequence of n items from a given [sample](https://en.wikipedia.org/wiki/Sample_(statistics)) of text or speech. The items may be [phonemes](https://en.wikipedia.org/wiki/Phoneme), [syllables](https://en.wikipedia.org/wiki/Syllable), [letters](https://en.wikipedia.org/wiki/Letter_(alphabet)), [words](https://en.wikipedia.org/wiki/Word) or [base pairs](https://en.wikipedia.org/wiki/Base_pairs) according to the application. The n-grams typically are collected from a [text](https://en.wikipedia.org/wiki/Text_corpus) or [speech corpus](https://en.wikipedia.org/wiki/Speech_corpus). When the items are words, n-grams may also be called shingles.

Using [Latin numerical prefixes](https://en.wikipedia.org/wiki/Latin_numerical_prefixes), an n-gram of size 1 is referred to as a "unigram"; size 2 is a "[bigram](https://en.wikipedia.org/wiki/Bigram)" (or, less commonly, a "digram"); size 3 is a "[trigram](https://en.wikipedia.org/wiki/Trigram)". [English cardinal numbers](https://en.wikipedia.org/wiki/Cardinal_number_(linguistics)) are sometimes used, e.g., "four-gram", "five-gram", and so on.

In computational biology, a [polymer](https://en.wikipedia.org/wiki/Polymer) or [oligomer](https://en.wikipedia.org/wiki/Oligomer) of a known size is called a [k-mer](https://en.wikipedia.org/wiki/K-mer) instead of an n-gram, with specific names using [Greek numerical prefixes](https://en.wikipedia.org/wiki/Greek_numerical_prefixes) such as "monomer", "dimer", "trimer", "tetramer", "pentamer", etc., or English cardinal numbers, "one-mer", "two-mer", "three-mer", etc. In this system, every word in websites are calculated as “bigram” to improve flexibility and accuracy of the entire system.

**2.4 Information Retrieval**

Information retrieval (IR) is the activity of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-based indexing. Information retrieval is the science of searching for information in a document, searching for documents themselves, and also searching for the metadata that describes data, and for databases of texts, images or sounds.

Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. Web search engines are the most visible IR applications.

An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may be matching the query, perhaps with different degrees of relevancy.

An object is an entity that is represented by information in a content collection or database. User queries are matched against the database information. However, as opposed to classical SQL queries of a database, in information retrieval the results returned may or may not match the query, so results are typically ranked. This ranking of results is a key difference of information retrieval searching compared to database searching.

Depending on the application the data objects may be, for example, text documents, images, audio, mind maps or videos. Often the documents themselves are not directly kept or stored in the IR system, but are instead represented in the system by document surrogates or metadata.

Most IR systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top-ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

The system can be assumed as Information Retrieval system because it extracts information about the laptop from three websites and make processing to it in helping the user to get the suitable laptop.

2.4.1 TFIDF

In information retrieval, Term Frequency–Inverse Document Frequency (TFIDF) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The TFIDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. TFIDF is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use TFIDF.

Variations of the TFIDF weighting scheme are often used by [search engines](https://en.wikipedia.org/wiki/Search_engine) as a central tool in scoring and ranking a document's [relevance](https://en.wikipedia.org/wiki/Relevance_(information_retrieval)) given a user [query](https://en.wikipedia.org/wiki/Information_retrieval). TFIDF can successfully be used for [stop-words](https://en.wikipedia.org/wiki/Stop-words) filtering in various subject fields, including [text summarization](https://en.wikipedia.org/wiki/Automatic_summarization) and classification. One of the simplest [ranking functions](https://en.wikipedia.org/wiki/Ranking_function) is computed by summing the TFIDF for each query term; many more sophisticated ranking functions are variants of this simple model [3].

The TFIDF weight is used in text mining and IR. The weight is a measure that is used to evaluate how important a word is to a document in a collection of documents. When using a simple technique like a frequency table of the terms in the document, stop words are removed, apply punctuation and stem the word to its root. And then, the importance of the word is measured in terms of its frequency; higher the frequency, the more important the word.

In case of TFIDF, the only text pre-processing is removing punctuation and lower casing the words. TFIDF is the product of the TF and IDF scores of the term. Higher the TFIDF score, the rarer the term is and vice-versa. TFIDF is successfully used by search engines like Google, as a ranking factor for content. The whole idea is to weigh down the frequent terms while scaling up the rare ones.

Term Frequency (tf): gives the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. It increases as the number of occurrences of that word within the document increases. Each document has its own tf.

(2.1)

**Inverse of Data Frequency (idf) is**used to calculate the weight of rare words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score. It is given by the equation below.

(2.2)

Combining these two come up with the TF-IDF score (w) for a word in a document in the corpus. It is the product of tf and idf:

(2.3)

Where, tfi,j = number of occurrences of i in j

dfi= number of documents containing i

N = total number of documents

For example, sentence 1 is “The car is driven on the road” and sentence 2 is “The truck is driven on the highway” In this example, each sentence is a separate document. And then calculate the TF-IDF for the above two documents, which represent our corpus.

## Table 2.1 IF-IDF Demonstration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Word | TF | | IDF | TFIDF | |
| A | B | A | B |
| The | 1/7 | 1/7 | log(2/2)=0 | 0 | 0 |
| Car | 1/7 | 0 | log(2/1)=0.3 | 0.043 | 0 |
| Truck | 0 | 1/7 | log(2/1)=0.3 | 0 | 0.043 |
| Is | 1/7 | 1/7 | log(2/2)=0 | 0 | 0 |
| Driven | 1/7 | 1/7 | log(2/2)=0 | 0 | 0 |
| On | 1/7 | 1/7 | log(2/2)=0 | 0 | 0 |
| The | 1/7 | 1/7 | log(2/2)=0 | 0 | 0 |
| Road | 1/7 | 0 | log(2/1)=0.3 | 0.043 | 0 |
| Highway | 0 | 1/7 | log(2/1)=0.3 | 0 | 0.043 |

From the table 2.1, TF-IDF of common words was zero, which shows they are not significant. On the other hand, the TF-IDF of “car”, “truck”, “road”, and “highway” are non-zero. These words have more significance [12][18].

**2.5 Cosine Similarity**

The system uses Cosine Similarity to calculate the similarity between different documents which means the features extracted from TFIDF is compared with the sample feature to determine the web page is about Laptops [13]. For cosine similarity beginning with the definition of the dot product for two vectors:  = (a1, a2, a3, …) and  = (b1, b2, b3, …) where an and bn are the components of the vector (features of the document, or TF-IDF values for each word of the document in our example) and the \mathit{n} is the dimension of the vectors ai and bi.

(2.4)

The definition of the dot product is a simple multiplication of each component from the both vectors added together. See an example of a dot product for two vectors

with 2 dimensions each (2D):

(2.5)

The first thing probably noticed is that the result of a dot product between two vectors isn’t another vector but a single value, a scalar.

This is all very simple and easy to understand, but what is a dot product? What is the intuitive idea behind it? What does it mean to have a dot product of zero? To understand it, the need to understand what is the geometric definition of the dot product.

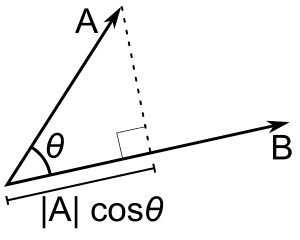
The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because only the magnitude of each word count (TF-IDF) of each document is not considered, but the angle between the documents. The cosine similarity equation is to solve the equation of the dot product for the cos .

(2.6)

Rearranging the equation to understand it better using the commutative property:

(2.7)

The term  is the projection of the vector  into the vector  as shown in Figure 2.4.



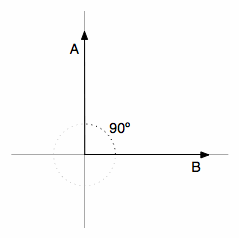
θ

A

B

**Figure 2.4 Cosine Similarity**

The vector  is orthogonal (with an angle of 90 degrees) to the vector  like on the image below as shown in Figure 2.5.

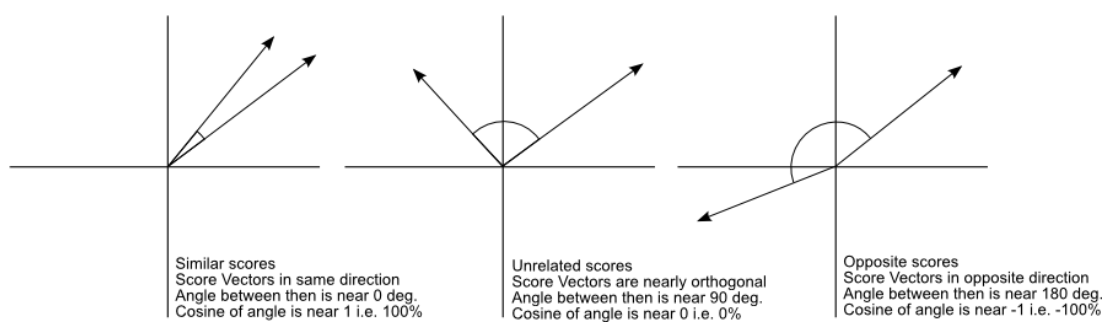


B

A

90˚

**Figure 2.5 Equivalent Degrees**

There will be no adjacent side on the triangle, it will be equivalent to zero, the term  will be zero and the resulting multiplication with the magnitude of the vector will also be zero. When the dot product between two different vectors is zero, they are orthogonal to each other (they have an angle of 90 degrees), this is a very neat way to check the orthogonality of different vectors. It is also important to note that are using 2D examples, but the most interesting fact about it is that can also calculate angles and similarity between vectors in higher dimensional spaces, and maths show far than the obvious which can’t visualize or imagine what is the angle between two vectors with twelve dimensions for instance.

**Figure 2.6 Similarity Scores**

In Figure 2.6, if a vector pointing to a point far from another vector, they still could have a small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents. If a document with the word “sky” appearing 200 times and another document with the word “sky” appearing 50, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, this is important when comparing documents.

A vector pointing to a point far from another vector, still could have a small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents. A document with the word “sky” appearing 100 times and another document with the word “sky” appearing 25, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, which is what matters when comparing documents.

**2.6 Edit Distance Algorithm**

The system is act like a Search Engine. So, the last part of a Search Engine is Searching. In this system, searching is implemented by using Edit Distance Algorithm. In and computer science and computational linguistics, edit distance is a method of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

Different definitions of an edit distance use different sets of string operations. The Levenshtein distance operations are the insertion, removal or substitution of a character in the string. Being the most common metric, the Levenshtein distance is usually what is meant by "edit distance".

2.6.1 Levenshtein Distance

Levenshtein distance is a [string metric](https://en.wikipedia.org/wiki/String_metric) for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after the Soviet mathematician [Vladimir Levenshtein](https://en.wikipedia.org/wiki/Vladimir_Levenshtein), who considered this distance in 1965.

Levenshtein distance may also be referred to as edit distance, although that term may also denote a larger [family of distance metrics](https://en.wikipedia.org/wiki/Edit_distance). It is closely related to [pairwise string alignments](https://en.wikipedia.org/wiki/Sequence_alignment#Pairwise_alignment). The Levenshtein distance is a number that tells how different two strings are. The higher the number, the more different the two strings are.

For example, the Levenshtein distance between “kitten” and “sitting” is 3 since, at a minimum, 3 edits are required to change one into the other.

1. **k**itten → **s**itten (substitution of “s” for “k”)
2. sitt**e**n → sitt**i**n (substitution of “i” for “e”)
3. sittin → sittin**g** (insertion of “g” at the end).

An “edit” is defined by either an insertion of a character, a deletion of a character, or a replacement of a character.

**2.7 Summary**

In this chapter, Web crawling and Web Scraping is explained in detail. The usage of TFIDF algorithm in the system explained. Each step of preprocessing before passing TFIDF algorithm is explained. These steps are lowercase conversion, stopwords and tags removal. Cosine similarity is used to calculate the similarity between different web pages. Levenshtein distance algorithm is used to search suitable laptops. In the next chapter, detail about the system design and implementation will be explained.

**CHAPTER 3**

**SYSTEM DESIGN AND IMPLEMENTATION**

This chapter describes in detail of the system flow, design and implementation of Web Crawling and Scraping Agent System with TFIDF and usage explanation to find a suitable laptop.

**3.1 System Design**

There are two major parts in the system: Crawling part and Searching part. The first part is Crawling part to recursively visit every link and download to process it. And, the second one is to apply a searching process for the user so that the relevant item can be shown.

**3.2 System Flow Diagram**

As there are two main parts, the system has two flow diagrams for each part of the system.

3.2.1 System Flow for Crawling

When system starts crawling, three websites are crawled to get the required information from that websites as shown in Figure 3.1 and each page is passed into IF-IDF checker code, and that code returns Boolean value of TRUE or FALSE. TRUE means the page is related to laptops or that page is about laptops. If FALSE is returned, the system knows that the page is not related to laptops or that page is not about laptops. If the page is related to laptops, and then extract features of each laptops.

These extracted features are processor, ram, storage capacity, price, display, links and graphic card information. But extraction is not easy as some of feature are written complexly in pages. That is, for example, for the [Intel® Core™ i9-9980HK Processor](https://ark.intel.com/content/www/us/en/ark/products/192990/intel-core-i9-9980hk-processor-16m-cache-up-to-5-00-ghz.html), the system tries to normalize the detail of each feature as 9 generation and core i9 by taking generation number and brand modifier. And, for the RAM, the system forgets about speed and version of RAM and take only capacity. For example, 16GB DDR4 as 16GB are used in the system. For the storage, only total capacity of 2048+256 = 2304 is used instead of taking full specifications such as 256 SSD and 2T HDD @ 5400 rpm.

For display and graphic information, everything that are shown in website such as 144Hz, 3ms, IPS-Level, 72%NTSC Thin Bezel, 100%sRGB and Nvidia GeForce RTX 2080 Ti are taken because meanwhile display and graphic information are not heavily used on this system. After normalization, the list of each laptop with detail is stored in database and the crawling process is finished. Three websites are crawled by the proposed system.

Pages/

Links

Start

Crawl entire 3 websites

Filter the Laptop pages using IT-IDF

Extract laptop's features

Normalization for laptop entry

Normalized laptop list

End

**Figure 3.1 System Flow for Crawling**

3.2.2 System Flow for Searching

When searching is come to process, levenshtein distance is used to get the required information. The user is asked to enter two options for choosing the laptop. After that, the system shows the relevant laptops according to user input by analyzing the database by using levenshtein distance as shown in figure 3.2. Normalized laptop list is obtained from the process of Web Crawling as shown in figure 3.1. Laptop feature is obtained from user input in which user can mainly choose only two in three of brand, specifications and prices. These inputs are regarded as single string in Levenshtein Distance Algorithm which is processed to get a relevant laptops list from the database which is useful for user to buy from the actual e-commerce websites.

Start

Laptop's feature

Retrieve relevant laptop by using Levenshtein distance

Normalized laptop list

Relevant laptops

End

**Figure 3.2 System flow for Searching**

**3.3 System Implementation**

When the system starts, the home page will be available to user that everyone can understand what website is for and what it can do for user. It has two buttons in the middle of the home page in Figure 3.3. First one is “Start Crawling”, which means when user selects on that button, the entire system wraps out the database table that already stored laptop’s data and starting extracting the laptop data from stretch via three websites. They are “[Linn](https://www.linonlinestore.com) Online Store”, “Shwe LaMin Nagar” and “Royal Smart”.



**Figure 3.3 Home Page**

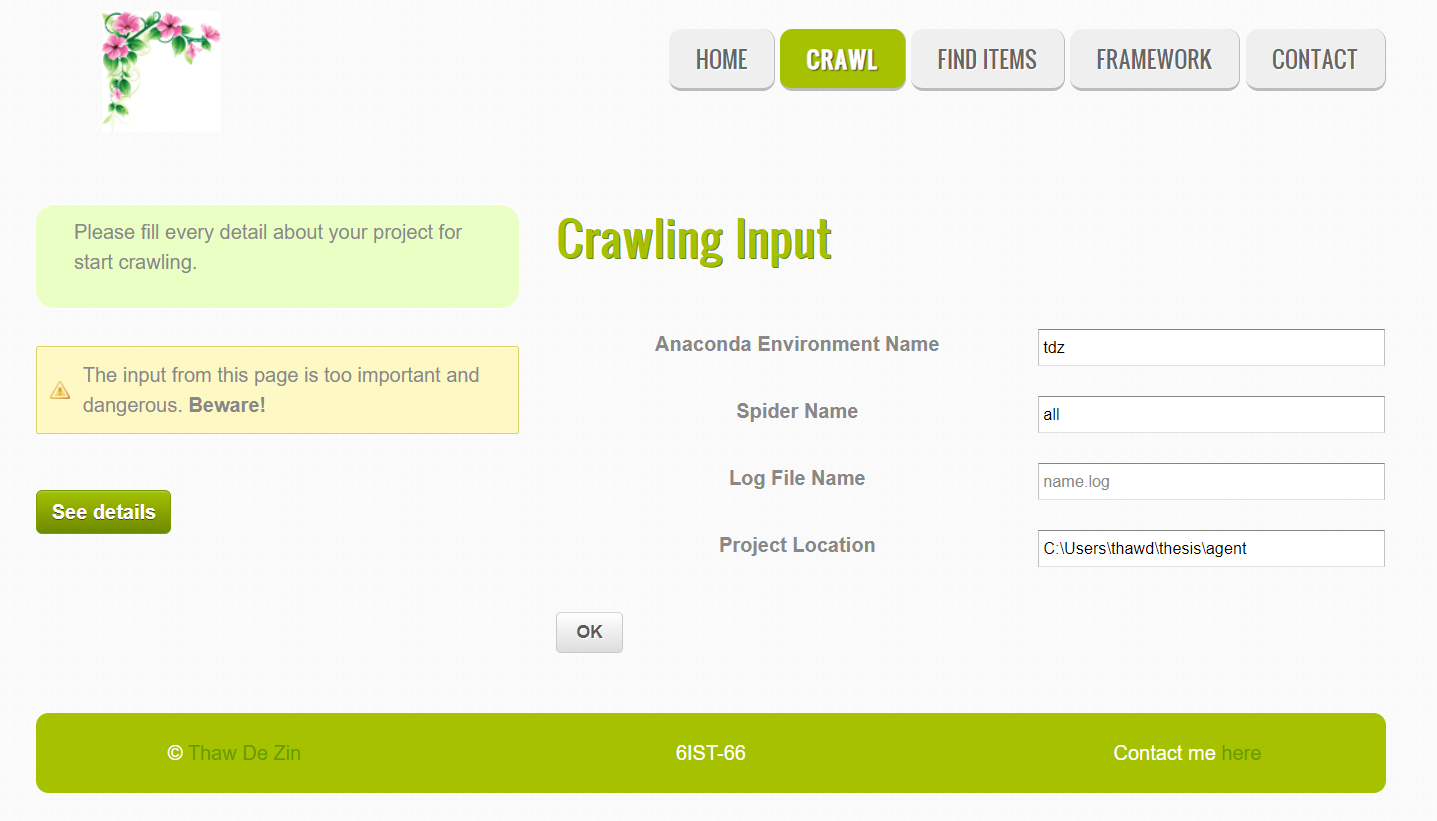
When user selects on “Find Items” button, the crawled and extracted data are available for user to search which laptops are match with requirements. User can choose two options from three options that are showed by page. These three options are “Brand”, “Specification” and “Price”. User can choose two of them.

3.3.1 System Implementation for Crawling

In Figure 3.3, when user selects “START CRAWLING” button, the page as shown in Figure 3.4 is available for user to input the most important data to start crawling. It is also administrator page. Usually, Crawling is only done by Website Administrator. But, in this system, user demands crawling options is available.

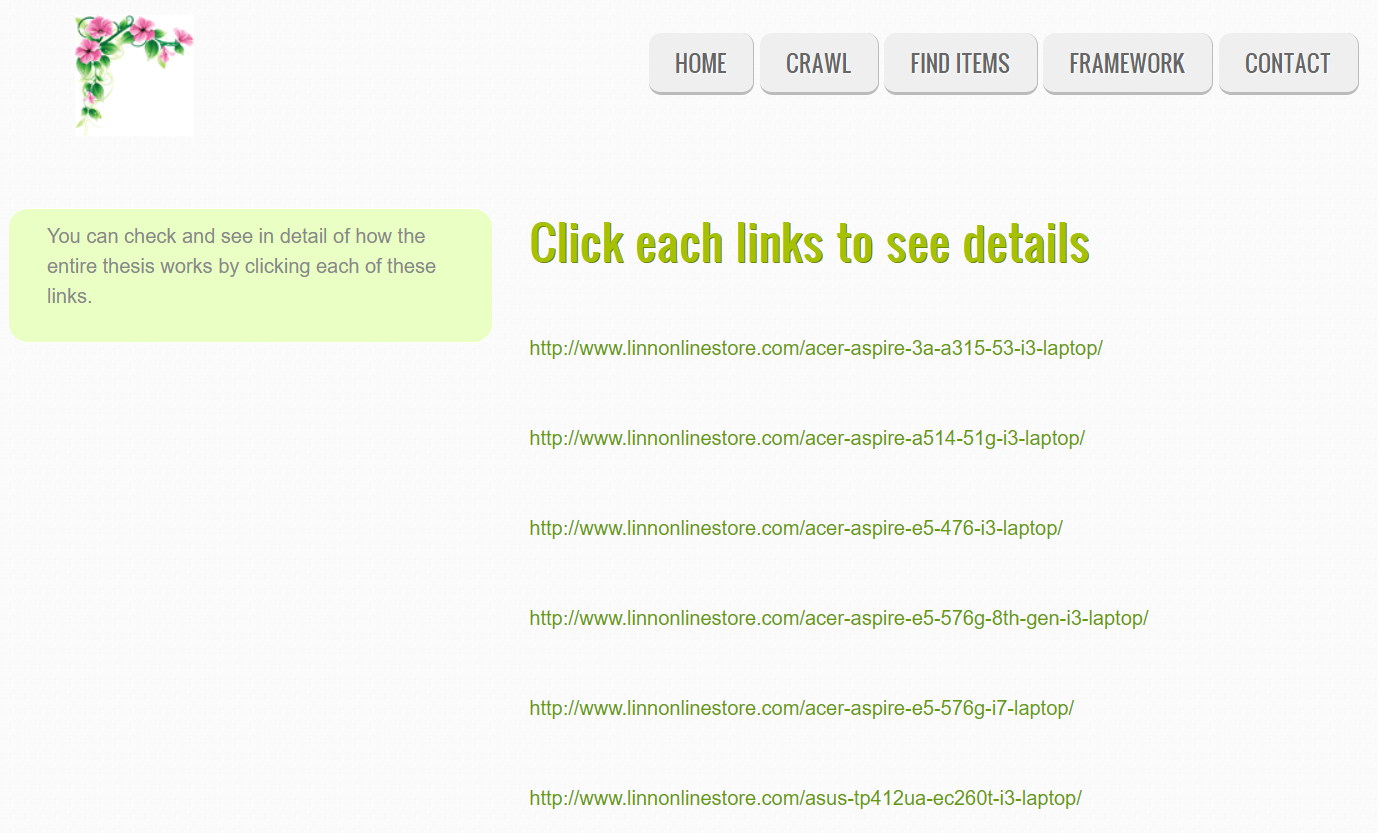
Proper Anaconda Environment Name is given, specific web spider name is also chosen, and user can rename the output of Log File and Project Location which is set default by web administrators.

When the user selects “FRAMEWORK” button on each page, the original website of the framework to learn and understand in detail is displayed about “Scrapy Framework” which is the main framework of the entire system.



##### Figure 3.4 Crawling Page

In Figure 3.4, there are four crawlers also known as spiders to fill in “Spider Name” Box. The first one is “all” spider which crawl all three websites to get the information about laptops. And other three crawlers are lin, sln and royal crawlers which crawls individual websites. Robots.txt in each website can be performed in crawling process.

**Figure 3.5 Detail Link Page**

When user selects on “See details” button, the page is available for user to select on each links and checks how web crawling and scraping with TF-IDF works as shown in Figure 3.5.



##### Figure 3.6 Response.body

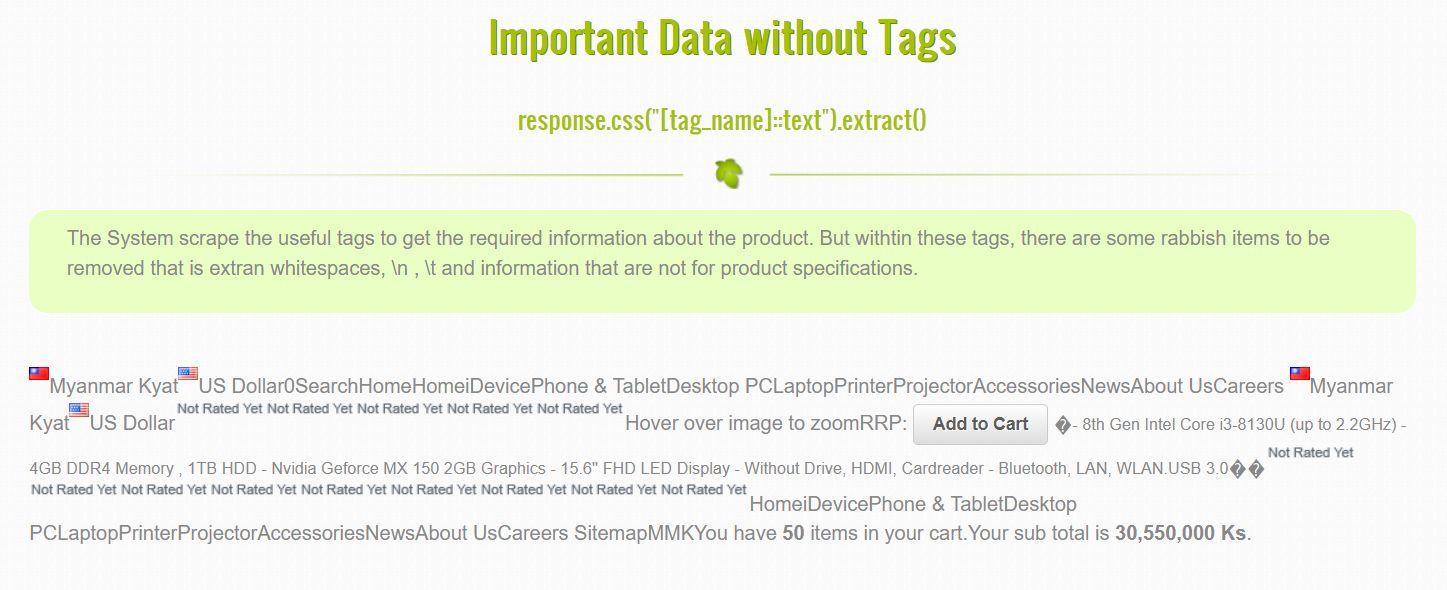
In Figure 3.6, when user selects on the links, get Step by Step Explanation of how system works is described to user. The first explanation is “response.body”. As the system uses Python’s Scrapy as a Crawling Framework. “response.body” is same as selecting view source in web browsers when viewing a webpage and Ctrl+U button is equivalent.

After that, removing html tags that are started not useful for the system’s process. So, the source of webpage are obtained without html tags as shown in Figure 3.7.

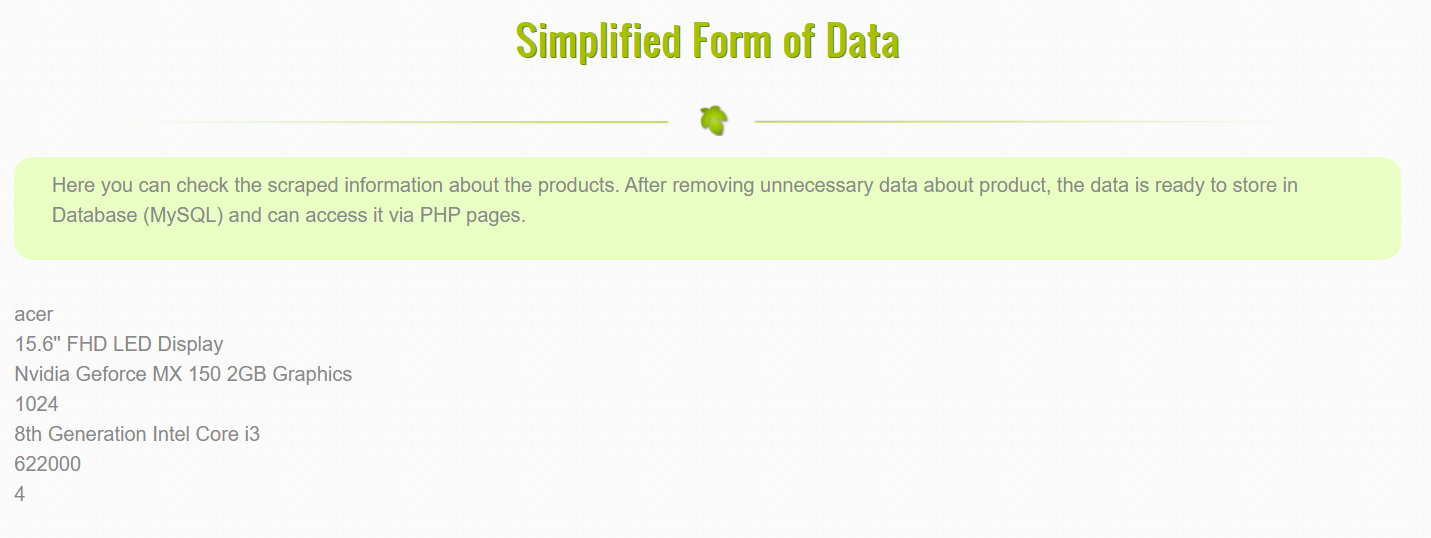
**Figure 3.7 Response.body without HTML Tags**

In Figure 3.7, the “response.body without html tags” strings are passed to TF-IDF method, which checks the input pages is related with Laptops are not. Before passing through to TF-IDF method, the required preprocessing such as stop words removal is also performed by the system. If the page is related to Laptops, continues to perform further processing. If the page is not related to Laptops, the system drops that page and start processing on new page.

Among that all available data, most of them are not useful for system’s processing. So, removing that unnecessary data is essential for system for further processing to extract laptop’s feature for every page as shown in Figure 3.8.

**Figure 3.8 Important Data without Tags**

As shown in Figure 3.9, after filtering the required data from tags, the simplified data are ready to store in database as brand, display, graphic, hdd, processor, price and ram.

**Figure 3.9 Simplified Form of Data**

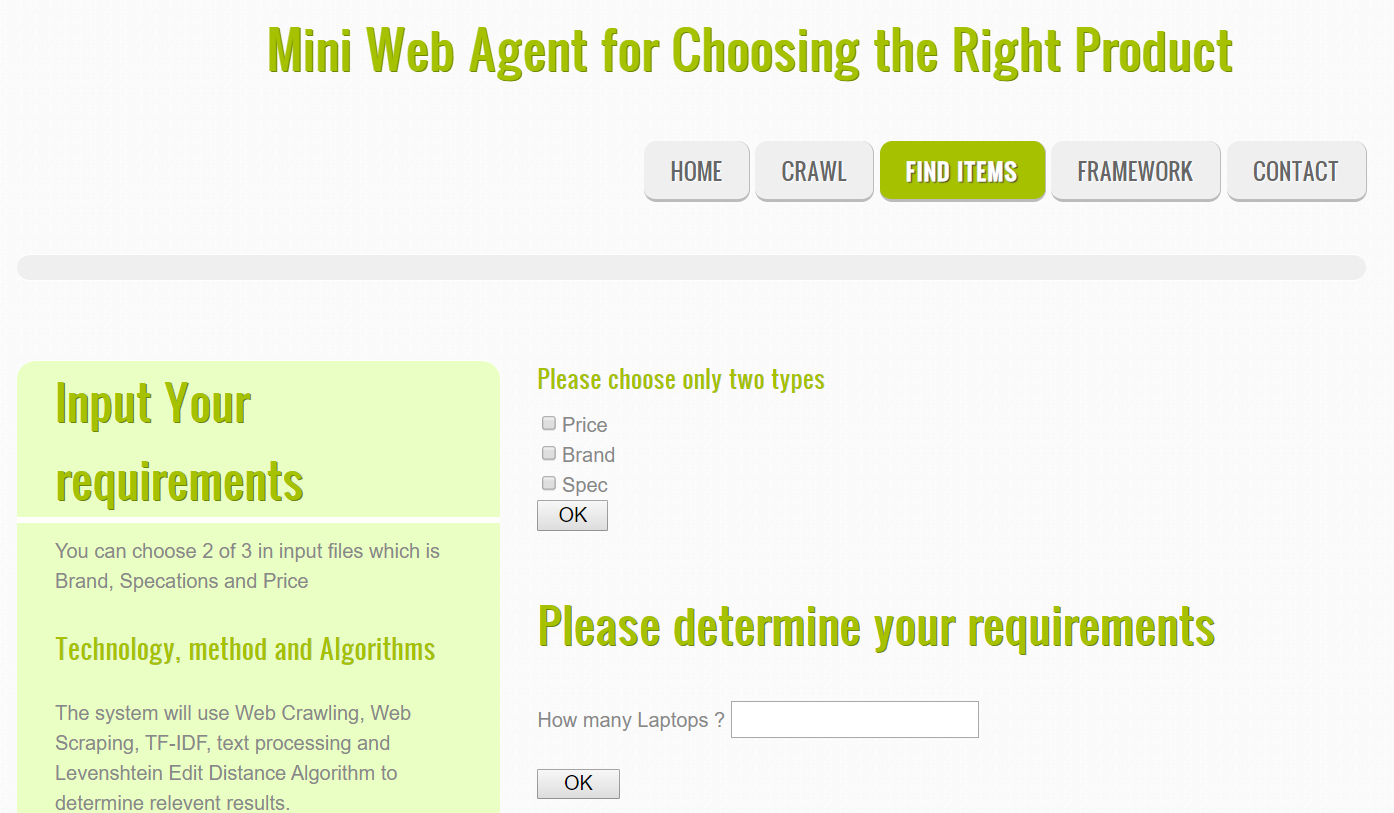
After processing each links of websites, the required data are automatically queried to database and can view the result of it as shown in Figure 3.10.



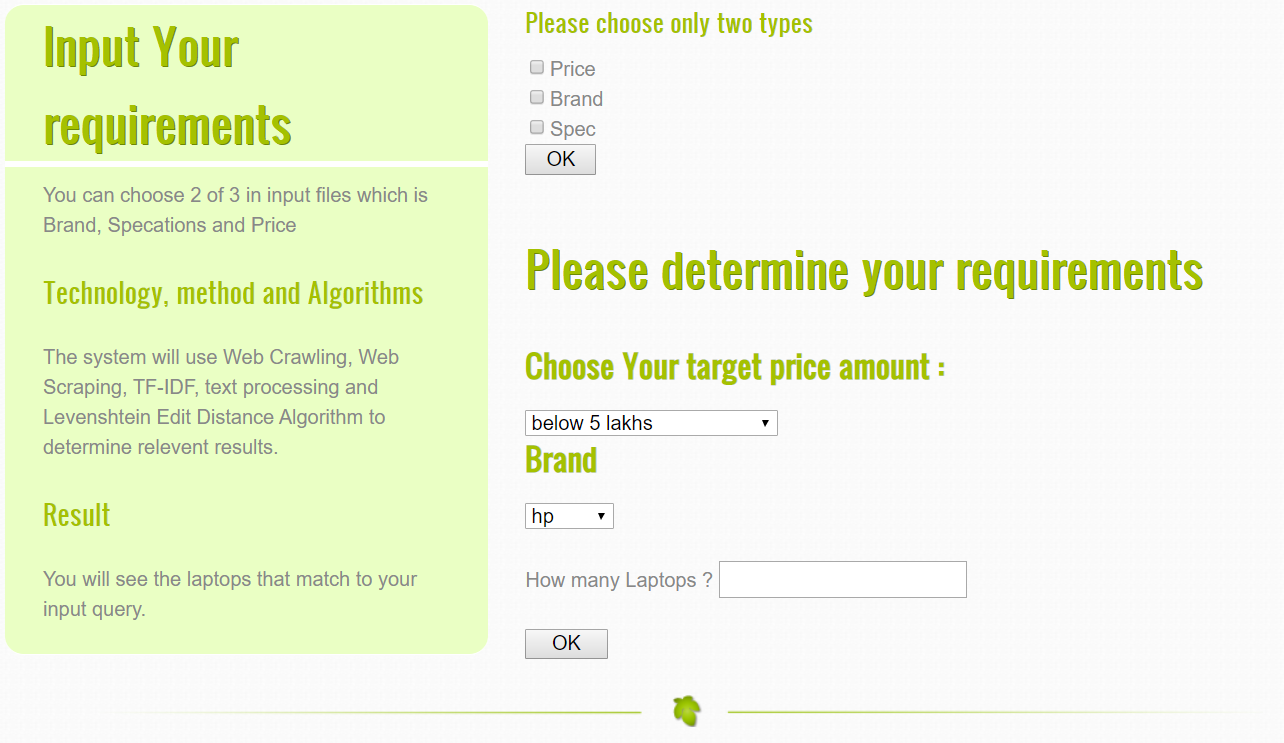
**3.10 Database View**

3.3.2 System Implementation for Searching

When the user selects “find items” button as shown in Figure 3.3 also known as Home Page, the searching page is shown to user. User can choose two in three options of price, brand and specifications as shown in Figure 3.11.

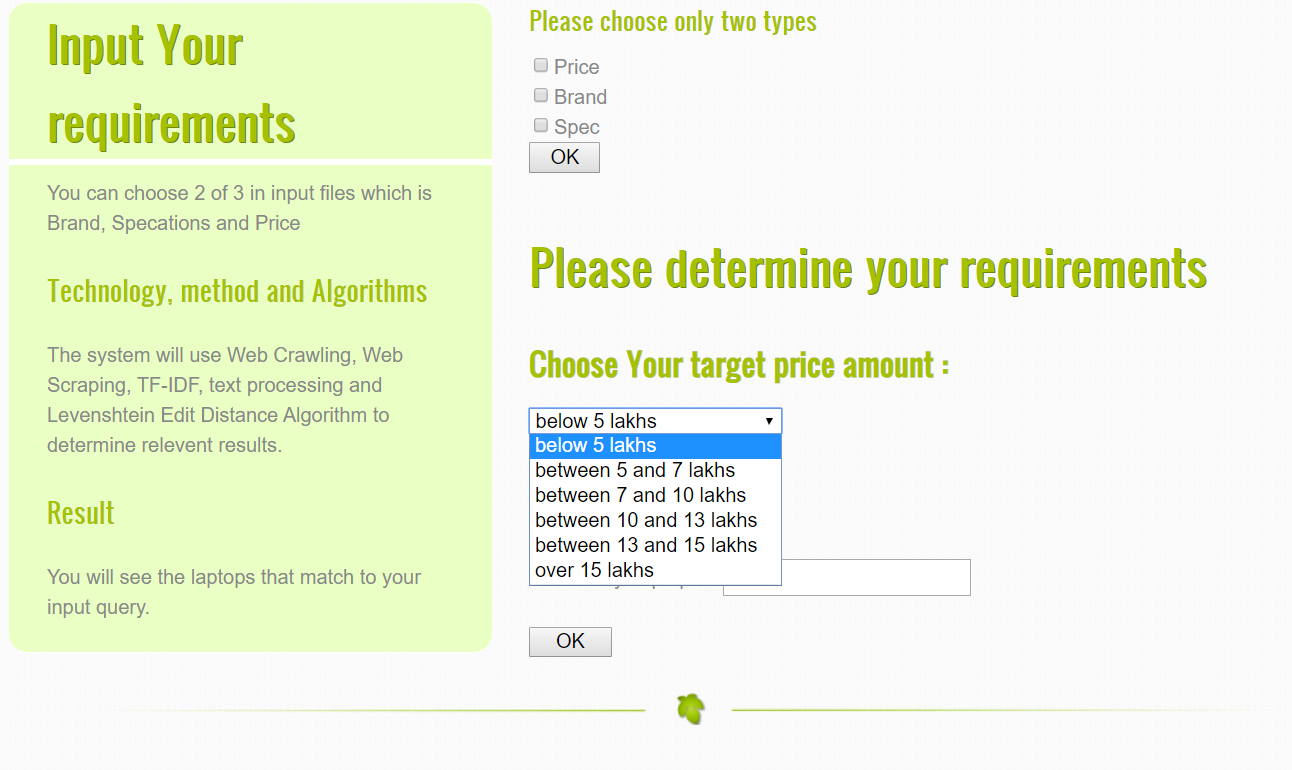


##### Figure 3.11 Searching Page with Two Options

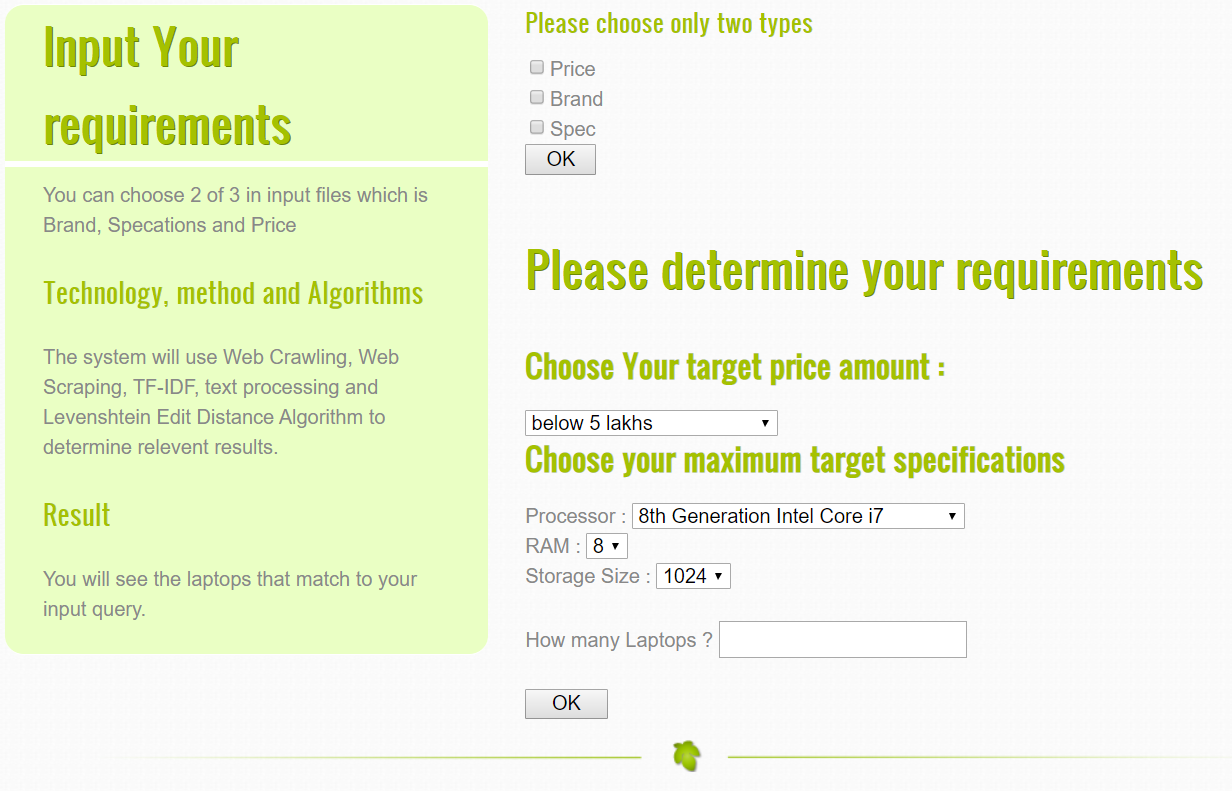


##### Figure 3.12 Price and Brand Options

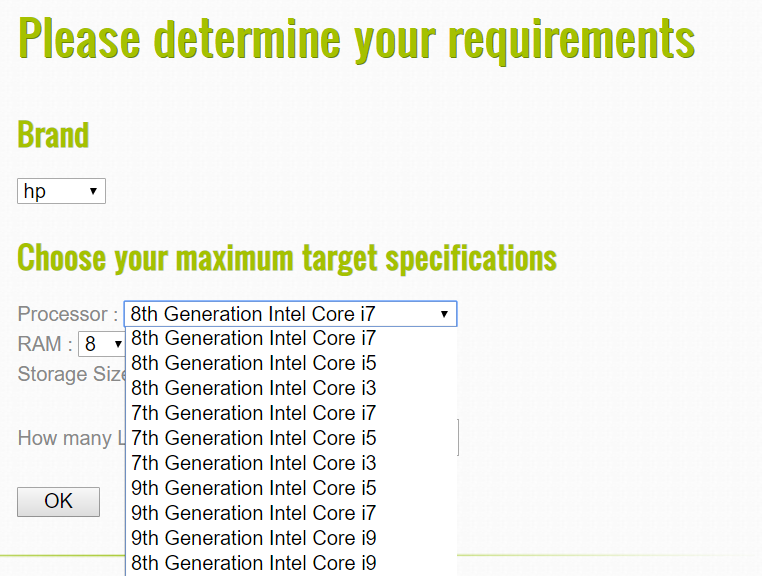
In Figure 3.12, when user chooses price and brand options, the target amount and brand are shown to input by user.

**Figure 3.13 Price Range**

In Figure 3.13, the price range is determined by the proposed system. For the user convenience, the proposed system is divided price range into six from minimum to maximum amount of laptop price.

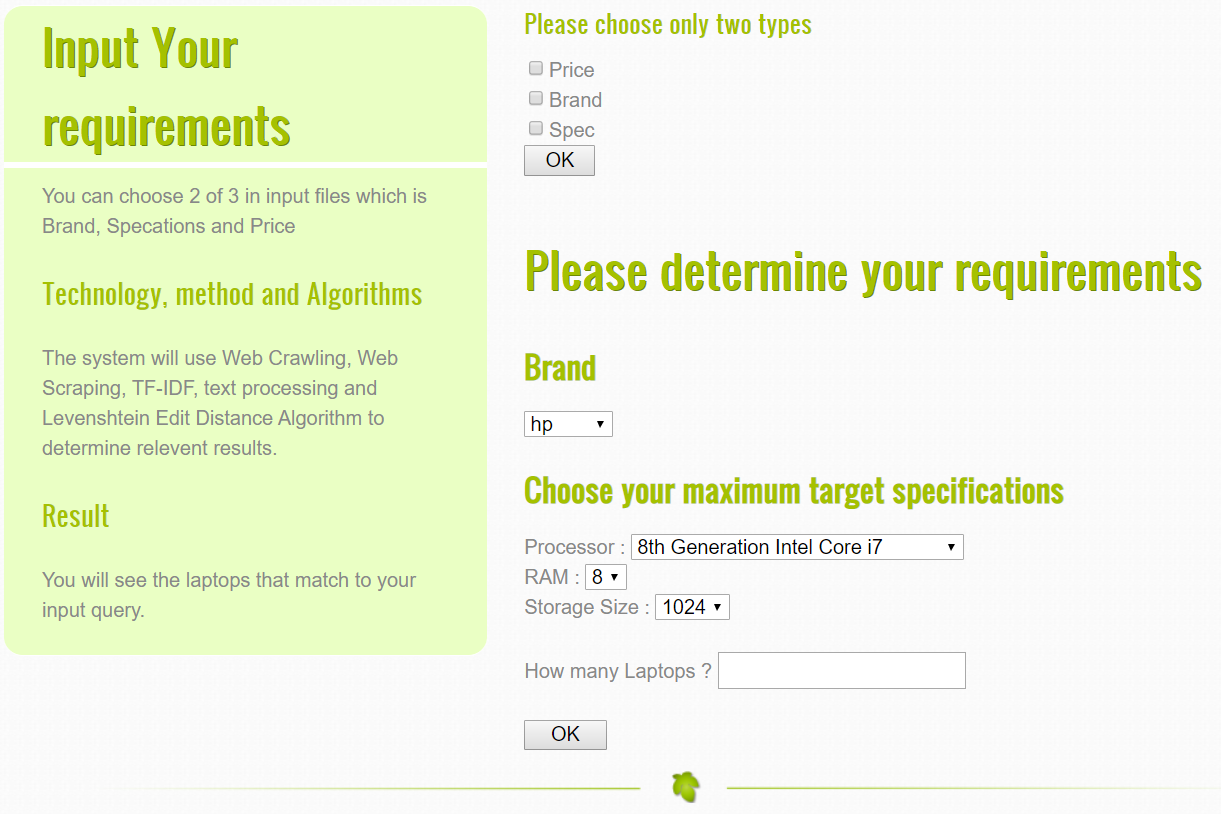
**Figure 3.14 Price and Specifications Options**

In Figure 3.14, if user chooses price and specification options, the processor, ram and storage size is added as input and also target price amount.



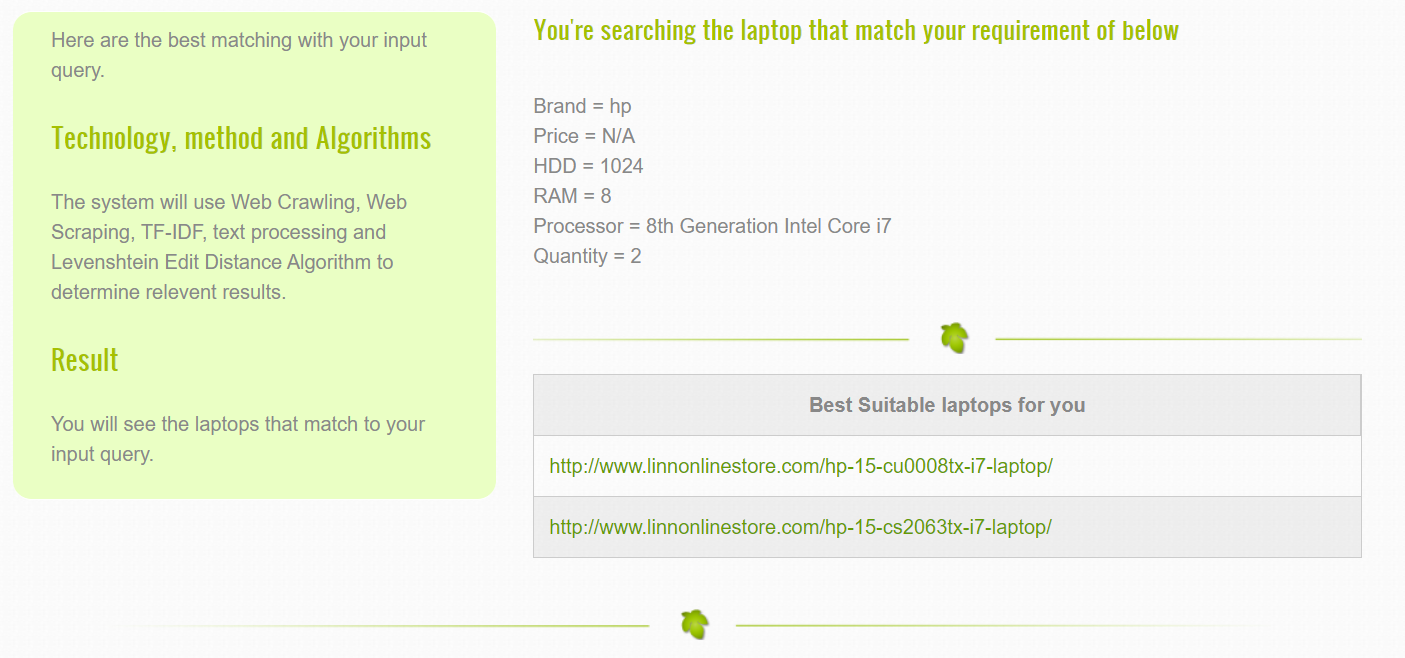
##### Figure 3.15 Processor Generation and Brand Identifiers

For the user convenience, the complex processor name is simplified into user readable format by taking two options of “Generation” and “Brand Identifiers” as shown in Figure 3.15.



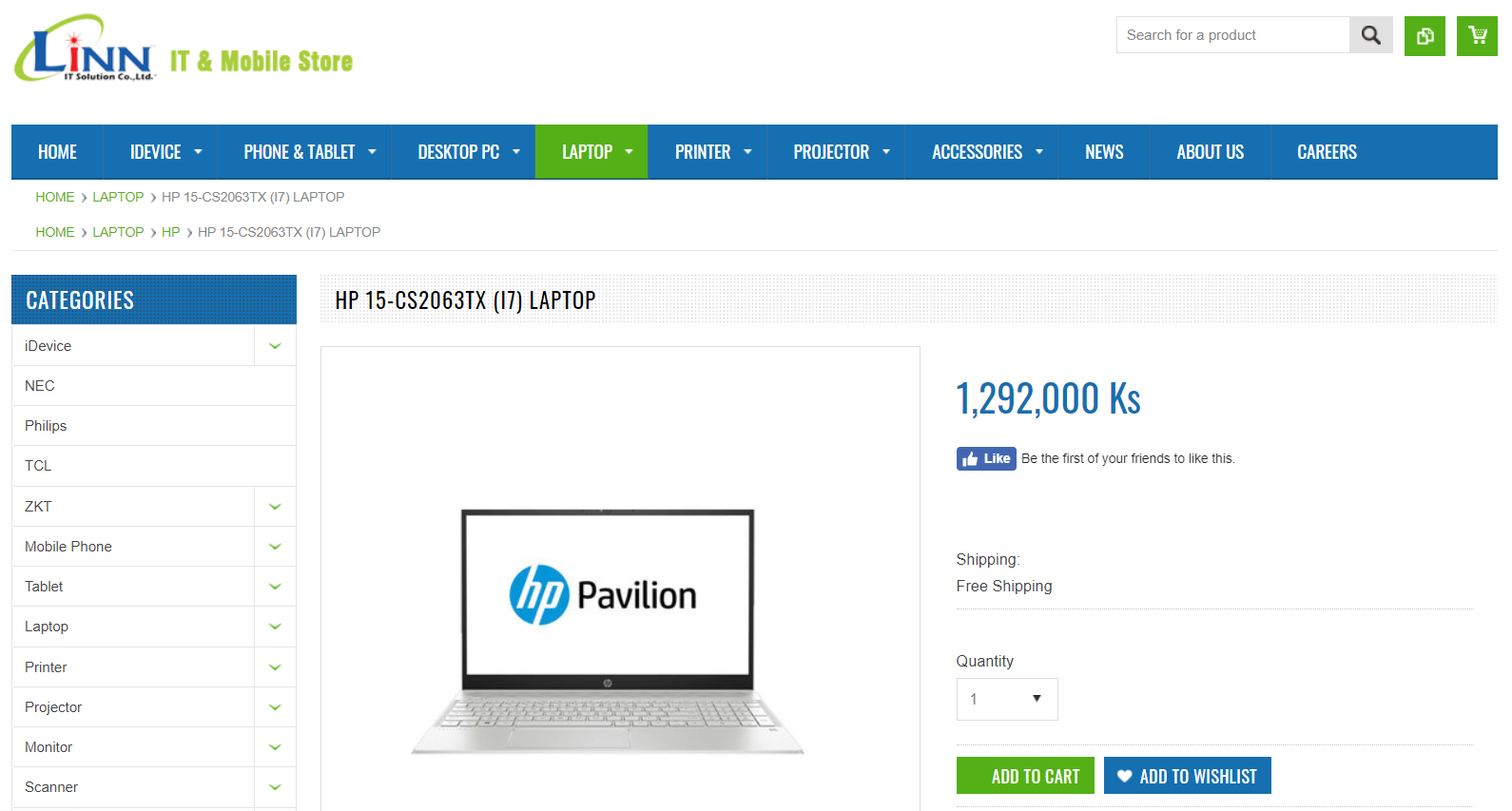
###### Figure 3.16 Brand and Specifications Options

In Figure 3.16, if user chooses brand and specification options, the processor, ram and storage size is added as input and also target price amount. User also need to input the number of laptops for the system to find that match with the inputs. Finally, User can see the most relevant laptops that match the input requirements with the list of input that the user entered previous page with user input and the best suitable laptops table with links. That links are representing the actual item location of each laptops.



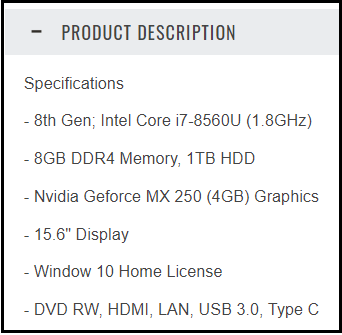
**Figure 3.17 Final Result**

The most suitable laptop is given that match with the user input from the websites as shown in Figure 3.17. And user can select on the available links under the “Best Suitable laptops for you” title and can buy via actual e-commerce websites.



**Figure 3.18 E-commerce Websites Page**

Figure 3.18 shows the web page when user selects link on Figure 3.17 where user can actually buy that product. And, the specifications chosen by user which is match according to the actual website as specifications are shown in Figure 3.19.



**Figure 3.19 Product Specifications from Website**

**3.4 Output Result**

The output result is a user-friendly web crawling and scraping Agent which is useful for searching the laptops that match the user input. User can also study in detail how web crawler works by viewing links available on the links.php.

**3.5 Summary**

In this chapter, detail design, implementation and usage of the system is explained. In the next chapter, limitations and benefits of using the system will explain and discuss some upgradable technical information.

**CHAPTER 4**

**CONCLUSION**

Web Crawling and Web Scraping are the most fundamental aspect in every Search Engine. Every crawler can visit every available links without approval. But there are some regulations that should obey by Crawlers. The proposed system is designed to obey all rules of Web Crawling and Scraping. The first rule of scraping the web is not to be a burden to the website. The second rule of web crawling is not to harm the website. So, the frequency and volume of queries make by this system not interfere with the website’s normal operations or burden the website’s servers as a Distributed Denial of Service (DDoS) attack.

The system is implemented by limiting the number of concurrent requests to the same website from a single IP and respecting the delay that crawlers of wait between requests by following the crawl-delay directive outlined in the robots.txt file usually available at the root of a website - www.example.com/robots.txt. This document describes what a crawler should or shouldn’t crawl according to the Robots Exclusion Standard which is also specifies what is considered as good behavior on that site, such as areas that are allowed to be crawled, restricted pages, and frequency limits for crawling.

It is always essential to identify and put contact details in the crawler’s User Agent header by adding contact form or abuse report where a system admin can contact to developer of a web crawler. And, the system is not breach General Data Protection Regulation by its implementation and also inspect the web data that are planning to extract is copyrighted or not as the exclusive legal right over a physical piece of work like an article, picture, movie, etc. The system is implemented to obey all the rule of web scraping.

**4.1 Benefits**

The system is fully designed to understand about how search engine works because it is implemented with the combination of basic and advanced technique of web crawling and scraping technology. It also help to study about web scraping use case and applied areas by developing a simple web scraping tool for choosing suitable

product from e-commerce websites.

In this system, Scrapy; Python Web Scraping Framework is used to implement the entire Crawling process. For the searching, PHP is used and MySQL is used as Data storage. The system is helpful for people who are needed to search the most relevant laptops for their miscellaneous purpose in a short period of time and for those who are the owner of laptops shops to make some kind of price comparison about their products.

**4.2 Limitation**

This system is mainly focus on Web Scraping technology not for user convenient but also for specific items that is only laptops are available for searching. So, user cannot be used the system to get information about other items. And the website that are used for crawling is only three websites. For the user convenient, the more website can be processed in crawling.

**4.3 Further Extension**

The system can be extended to get all items such as mobile phones, printers, projectors and so on. The system is Multi-threaded but not distributed. So, the entire system can be upgraded using Distributed Web Crawling Techniques.

**REFERENCES**

[1] Broucke, S. V. & Baesens, B (2018). *Practical Web Scraping for Data Science.* [*https://doi.org/10.1007/978-1-4842-3582-9*](https://doi.org/10.1007/978-1-4842-3582-9). Leuven, Belgium.

[2] Hajba, G. L. (2018). *Website Scraping with Python.* <https://doi.org/10.1007/978-1-4842-3925-4>. Sopron, Hungary.

[3] J. Pei, “Information Retrieval and Web Search -- Web Crawling”. Retrieved from https://dl.acm.org/citation.cfm?id=1835525.

[4] Kouzis-Loukas, D. (2016). *Learning Scrapy*. Retrieved from [www.packtpub.com](http://www.packtpub.com).

[5] Loukas, D. K. (2016) *Learning Scrapy - Learn the art of efficient web scraping and crawling with Python*. Retrieved from https://www.amazon.com/Learning-Scrapy-Dimitrios-Kouzis-Loukas/dp/1784

[6] Michael Schrenk, M. (2007). *Webbots, Spiders, and Screen Scrapers*. Retrieved from <https://zh.b-ok.cc/book/973163/d3f132>

[7] Mitchell, R. (2018). *Web Scraping with Python*. Retrieved from <https://shop.oreilly.com>

[8] Pallavi & Sharma, R. (2014). *An Adaptive, Selective and Incremental Web Crawler.* Retrieved from https://www.ijser.in/archives/v3i7/20071502.pdf.

[9] Trupti V. Udapure, T.V & Kale, R.D. & Rajesh C. Dharmik, R. C. (2014). *Study of Web Crawler and its Different Types.* Retrieved from https://pdfs.semanticscholar.org/3260/1ca5ac22427b6ad56938e44f88098035593a.pdf

[10] https://towardsdatascience.com/https-medium-com-hiren787-patel-webscraping-applications-a6f370d316f4

[11] <http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/>

[12] <http://blog.christianperone.com/2011/10/machine-learning-text-feature-extraction-tf-idf-part-ii/>

[13] <http://blog.christianperone.com/2011/10/machine-learning-text-feature-extraction-tf-idf-part-iii/>

[14] <https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/>

[15] <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

[16] <https://librarycarpentry.org/lc-webscraping/04-scrapy/index.html>

[17] <https://medium.com/@djerahahmedrafik/web-scraping-fundamentals-using-scrapy-84a1e64b5ec>

[18] <https://towardsdatascience.com/tfidf-for-piece-of-text-in-python-43feccaa74f8>

[19] <https://www.promptcloud.com/blog/evolution-of-web-crawling-how-crawling-emerged-as-mainstream-discipline/>