DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

Metadata Scraping Using Programmable Customized Search Engine

[1]Esraa Q. Naamha, [2]Matheel E. Abdulmunim[1]
[1,2]*Computer Sciences Department, University of Technology, Baghdad, Iraq*
[1]*cs.20.29@grad.uotechnology.edu.iq,* [2]*matheel.e.abdulmunim@uotechnology.edu.iq*

Abstract*— The World Wide Web (WWW) is a vast repository of knowledge, including intellectual, social, financial, and security-related data. Online information is typically accessed for instructional purposes. On the internet, information is accessible in a variety of formats and access interfaces. Because of this, indexing or semantic processing of the data via websites may be difficult. The method that seeks to resolve this issue is web data scraping. Unstructured web data can be converted into structured data using web data scraping so that it can be stored and examined in a central local database or spreadsheet. This paper offers a metadata scraping using a programmable Customized Search Engine (CSE) system, which can extract metadata from web pages (HTML pages) in the Google database and save it in an XML format for later analysis and retrieval. Documents that contain metadata are a relatively recent phenomenon on the web and increase the likelihood that users will find the information they need.*

Index Terms— *Programmable (CSE), JSON API, API key, metadata scraping.*

I. INTRODUCTION

Any research, whether it be academic, marketing, or scientific, must have data. It may be desirable for people to gather and examine data from several websites. The various websites that fall under the given category present information in various ways. Even using a single website, all the information might not be available at once. The information may be spread across several pages in various areas. The vast majority of websites do not maintain a local copy of the data they offer on their webpages. Data from the webpage must be manually copied and pasted into a local computer file. This is a very tiresome procedure that takes a lot of time. Web scraping is a method for combining data from various websites into a single spreadsheet or database, making it simple to evaluate and even visualize the information [1]. Web scraping is the process of extracting data from the web programmatically and transforming it into a structured dataset. Web scraping allows for larger amounts of data to be collected in a shorter span of time and in an automated fashion that minimizes errors. There are two types of web scraping. The first is screen scraping, where data can be extracted from the source code of a website with an HTML parser or regular expression matching. The second is using application programming interfaces, commonly referred to as APIs. This is where a website offers a set of structured HTTP requests that return JSON or XML files [2]. The web scraping procedure is divided into three stages, which are as follows [3]:

- Fetching stage: The desired website with the relevant information must first be accessed in what is known as the fetching phase. This is accomplished via the HTTP protocol, which is an Internet protocol for sending and receiving requests from web servers. Web browsers utilize similar methods to get material from web pages. In this step, libraries such as curl 2

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

and wget 3 can be used by sending an HTTP GET request to the target address (URL) and getting the HTML page as a response.

- Extraction stage: After retrieving the HTML page, the important data should be extracted. Regular expressions, HTML parsing libraries, and XPath queries are utilized in this step, which is referred to as the extraction stage. The XML Path Language (XPath) is a tool for finding information in documents.
- Transformation stage: Now that just the relevant data remains, it may be converted into a structured format for presentation or storage. Using the stored data, information can be gathered that can help the business intelligence team make a better decision and much more.

The main contributions of this paper are: Create a programmable CSE based on an XML file to scrape the metadata of the links from various web pages (HTML pages) in the Google database using an API key and save them in a JSON format for later analysis and retrieval.

II. RELATED WORKES

Achmad M. et al. [4] describe a method for automatically retrieving the title, publication date, author, clean text article, and URL address of a news article from the HTML page of three news websites, namely Detik, ribunnews, and Liputan 6, without manually copying and pasting the information. This method consists of three steps: analyzing the structure of news websites, creating Regex patterns, and implementing the patterns as a set of rules for web scraping.

Fatmasari et al. [5] collect weather data from websites in South Sumatera and the surrounding area using web scraping technology. Web scraping technology is a technique for specifically retrieving the contents of a web page. The data collected by the Beautiful Soup and Requests library will form a database or data warehouse that can be used for further research on weather forecast data mining in South Sumatra, which in the future can be developed into a weather based decision support application.

Lanny A. and Srujan K. [6] use the web scraping mechanism to extract financial statements from diverse websites and make investment decisions based on further research and analysis. In this thesis, the outcome of the web scraping is to keep the extracted data in a database. The gathered data of the resultant database can be implemented for the required goal of further research, education, and other purposes with the further use of the web scraping technique.

III. META DATA

The phrase "metadata," which was allegedly first used in 1969, is also known as "information about information" or "data about data"[7]. The word "meta" is a Greek word that can indicate "higher order," "more fundamental kind," or "above," "beyond," and "of something in a different context." A metadata record is made up of a number of pre-defined elements that each have one or more values and indicate different characteristics of a resource. These components could be the features of the data, such as its origin, quality, and other attributes, or other pieces of information that provide the best approach to communicating information about the data without actually providing it. In other words, metadata is organized data that identifies, describes, locates, or otherwise makes it simpler to access, utilize, or manage an information resource. Additionally, it asserts that metadata is essential for assuring the survival and accessibility of digital resources in the future

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

[8][9]. The use of meta tags, the Dublin Core Metadata Element Set (DCMES), and the Resource Description Framework (RDF) are examples of metadata standards for characterizing Internet resources (i.e., the encoding of Web-related metadata)[10][11].

IV. SEARCH ENGINE

Users access information from the WWW via search engines like Google. In a search engine, there are three crucial elements. They function as the crawler, indexer, and ranking algorithm. A crawler is another name for a robot or spider that browses the internet and downloads web pages. Sending the downloaded pages to an indexing module allows them to be parsed and create an index based on the keywords they contain. Usually, keywords are used to keep an index current. The query processor component of a search engine matches a user's query's keywords with the index and provides the user with the URLs of the sites that match their search query. However, search engines perform a ranking mechanism before displaying the pages to the user in order to display the most relevant pages at the top and the least relevant ones at the bottom [12][13][14].

V. WEB DATA SCRAPING

Web scraping is a method for extracting data from the WWW and storing it in a file system or database for subsequent retrieval or analysis. It is sometimes referred to as "web extraction" or "web harvesting". The Hypertext Transfer Protocol (HTTP) or a web browser are frequently used to scrape web data. A user can do this manually, or a bot or web crawler can do it automatically. Web scraping is widely recognized as an effective and strong strategy for gathering vast amounts of data due to the fact that a tremendous amount of heterogeneous data is constantly generated on the WWW [15][16]. The use of completely automated algorithms that can transform entire webpages into well-organized data sets has replaced smaller ad hoc, human-assisted methods in contemporary web scraping strategies to adapt to a variety of settings. Modern web scraping can parse JSON files or markup languages in addition to integrating with computer visual analytics and natural language processing to mimic how people explore websites [17][18]. Obtaining web resources and then extracting relevant information from them are the two consecutive processes that make up the process of gathering data from the Internet. A web scraping tool specifically starts off by creating an HTTP request to obtain resources from a selected website. This request can either be presented as a URL with a GET query or as a portion of an HTTP message with a POST query. The requested resource will be obtained from the website and then delivered back to the web scraping program after the targeted website has successfully received and processed the request. The resource can be in multiple formats, such as web pages that are built from HTML, data feeds in XML or JSON format, or multimedia data such as images, audio, or video files. The web data is downloaded, and the extraction process continues to parse, reformat, and organize the data in a structured way [19][20].

VI. THE PROPOSED SYSTEM

A programmable CSE will be designed and implemented to extract and store the metadata of the links in XML format. The related links will be automatically collected from the search engine's database using the Application Programmable Interface (API). In general, the proposed system has four main steps, as listed below:

Step 1: Design a programmable CSE.

Step 2: Enable search queries in the programmable CSE.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

Step 3: Design a programmable CSE using an API.
Step 4: Design a programmable CSE using a JSON file.
The proposed system has been implemented based on the following algorithm (1) that has four main steps as it has been illustrated in *Fig. 1*.

Algorithm (1): Metadata Scraping

Input: CSE Query, API Key, CSE-ID.
Output: The CSE-XML File that Contains Metadata.
Begin
Step 1: Design a Programmable CSE.
  Step 1.1: Create a programmable CSE based on XML file.
    Step 1.1.1: Defining a Programmable CSE Structure (Context XML file).
    Step 1.1.2: Defining Sites to Search (Annotations XML file).
  Step 1.2: Design a Programmable CSE Query in XML Configuration.
    Step 1.2.1: Design the CSE XML Document Description
    Step 1.2.2: Design the CSE XML-URL Template.
    Step 1.2.3: Design a Programmable CSE Query Element.
  Step 1.3: Design Advanced Programmable CSE.
    Step 1.3.1: Determine the appropriate XML file format for our needs.
    Step 1.3.2: Determine the Specification Tags of the Programmable CSE.
    Step 1.3.3: Defining the Programmable CSE export results.
  Step 1.4: Create and Edit Programmable CSE Files
Step 2: Enable Search Queries in the Programmable CSE.
  Step 2.1: Design and Implement the Search Box in the XML File.
  Step 2.2: Enabling the Autocomplete XML Search File.
  Step 2.3: Customizing XML File Search Results.
Step 3: Design a Programmable CSE Based on API Definition.
  Step 3.1: Determine the API key for the programmable CSE.
  Step 3.2: Use of an API Engine key and a Design Data Model in Programmable CSE.
Step 4: Design a Programmable CSE using a JSON File.
  Step 4.1: Extract metadata from the Programmable CSE -JSON API.
  Step 4.2: Extract the Metadata Objects from the XML file in the Programmable CSE.
End

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

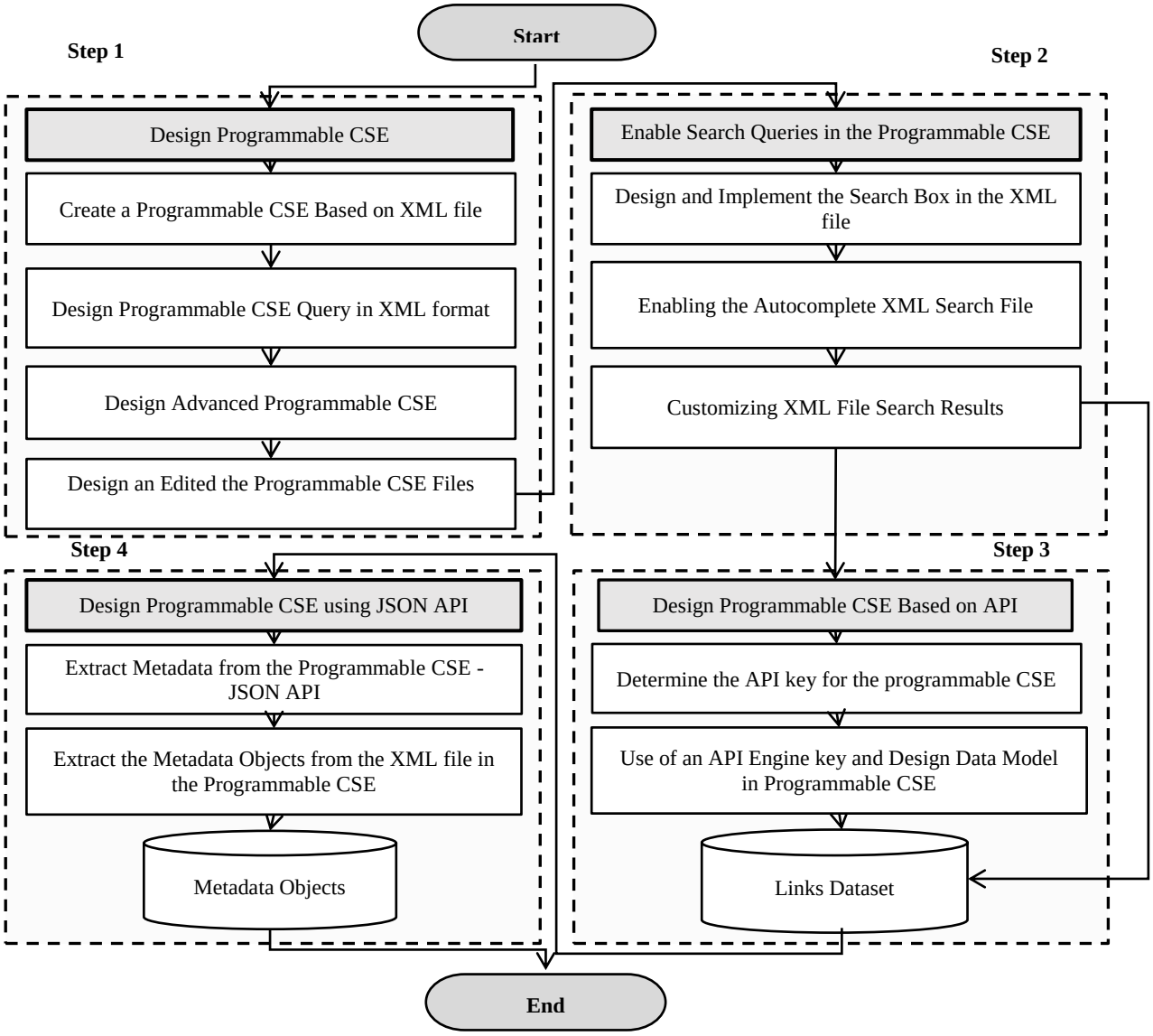

FIG. 1. BLOCK DIAGRAM OF THE PROPOSED SYSTEM.

A. Design Programmable CSE

This step, a virtual CSE will be created based on the API. In this step, a search engine is created to implement a programmable CSE. A programmable CSE is defined based on using the XML file, which in this case, the JSON file will be accessed and deal with the metadata. Generally, an XML file is created through the Python platform to define the search engine. To do that, the following steps are implemented:

1. *Create a Programmable CSE Based on XML File:* In this step, the programmable CSE-based JSON file using an API is defined and created. Once the API key is defined and created, the programmable CSE and the XML file for the programmable CSE are downloadable, as well as the annotations and context of the CSE XML file. In general, programmable CSE based on the XML file has two components. Each is controlled by the XML file designed and constructed to run and control the programmable CSE. The programmable CSE created using an XML file is illustrated in algorithm (2).

Received 30/October/2022; Accepted 10/January/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

Algorithm (2): XML File Configuration
Input: Initial XML File**.** **Output:** XML File Configuration**.** **Begin** **Step 1:** Create the root element. **Step 2:** Create annotation tag. **Step 3:** Associate the website with its search engine using the following steps: **Step 3.1:** Configure the search label tag. **Step 3.2:** Specify how that site should be treated by the search engine by getting the labels for the search engine from the context section of the Advanced tab in the Control Panel. **Step 3.3:** Set the name of the search engine label in the context file. The label for including sites is in the form of cse_xxxxxxxxxxx, where x is a character, and the label for excluding sites is in the form of cse_exclude_xxxxxxxxxxx. **Step 4:** To add more sites, create and define another annotation element. **Step 5:** Save and export the XML file. **End**

1.1. *Defining a Programmable CSE Structure (Context XML File)**:*** The first component (context) of the XML file is the control tag. This context in the XML file describes the basic features of the programmable CSE, such as wherever using the image searching option or promotions in the programmable CSE is enabled. The XML file's context outlines the search engine's structure and establishes its behavior.

1.2. *Defining Sites to Search (Annotations XML File)**:*** The second component of the XML file that describes and runs the programmable CSE is the Annotations tag, which includes which webpages or websites (indicates the preferences) that the CSE wants to be included in the search engine. Each website has associated information that is called an annotation. Programmable CSE allows for building a large search engine database. A large website collection can be achieved by managing a lot of sites that can be added or listed in the annotations file that is built into the XML file. In general, an annotation XML file is an XML file that lists a set of annotations. There are two parts to every annotation: the site and its associated labels. When a site is accessed, the label in the XML file instructs the programmable CSE how to handle it, such as by excluding, promoting, or demoting the site. The site labels must be defined in the context XML file and must be tagged with the proper labels in various forms in the annotations file. Any of the following formats are acceptable for annotation files:

- A Markup Language for Outlines Used by Outliner Processors (OPML): Use the OPML format to develop a search engine using an existing OMPL file (feed-based search engine), so annotations do not need to be created from scratch if there are already OPML files with URL patterns.
- Text files with Tab-Separated Values (TSV): The TSV format can be used to create a search engine that does not require all of the advanced features because the annotations can be managed in a more readable manner. It is also possible to use a spreadsheet editor and benefits from many advanced features, such as applying labels, associating scores, and adding comments.
- Programmable Search XML: The Programmable Search XML format can be used because it is the most powerful format to build a sophisticated and highly configurable search engine. It is suitable for programmers who want to build sophisticated, feature-rich search engines. It allows developers more freedom and power over how the search results are ranked.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

2. *Design a Programmable CSE Queries in XML Format:* In this step, the implementation of the programmable CSE Query using an XML file is illustrated in algorithm (3).

Algorithm (3): Design Query in XML File

Input: XML file Configuration, CSE-URL.
Output: XML File.
Begin
Step 1: Create the file namespace in the XML file using the URL for the XML data format.
Step 2: Set the OpenSearch engine's description information.
  Step 2.1: Set the type of the OpenSearch document description.
Step 3: Set the OpenSearch engine description elements in the XML file**.**
  Step 3.1: Set the root node of the OpenSearch XML description file.
  Step 3.2: Set the short name elements in the OpenSearch XML description file.
  Step 3.3: Set the description element in the OpenSearch XML description file.
  Step 3.4: Set the url element in the OpenSearch XML description file.
  Step 3.5: Set the url rel. value as "results" in the OpenSearch XML file.
  Step 3.6: The Contact element in the OpenSearch XML file.
  Step 3.7: The Tags element in the OpenSearch XML file.
  Step 3.8: The LongName element in the OpenSearch XML file.
  Step 3.9: The "Image element" in the OpenSearch XML file.
  Step 3.10: The "Query element" in the OpenSearch XML file.
  Step 3.11: The "Developer element" in the OpenSearch XML file.
  Step 3.12: The "Attribution element" in the OpenSearch XML file.
  Step 3.13: The "SyndicationRight element" in the OpenSearch XML file.
  Step 3.14: The "AdultContent element" in the OpenSearch XML file.
  Step 3.15: The "Language element" in the OpenSearch XML file.
  Step 3.16: The "InputEncoding element" in the OpenSearch XML file.
  Step 3.17: The "OutputEncoding element" in the OpenSearch XML file.
Step 4: Set the searching engine that is published in the OpenSearch description element in the XML file.
Step 5: Set the OpenSearch url template syntax format that can be used to represent the url description element in the XML file.
End

To define the programmable CSE Query-based XML file format, the following XML file needs to be created and designed based on the following format:

2.1. *Designing a CSE XML Document Description:* Designing the OpenSearch description document that may be used to describe a search engine's online interface is the first step in creating a CSE XML file.

2.2. *Designing a CSE XML-URL Template:* The second step of building the CSE XML file is based on designing the OpernSerach URL format that can be used to determine the URL of the search engine that will be used in the programmable CSE. Here, the customized search client will interpret the URL template according to the parameters supplied by the template: {name}.The parameter names are hardcoded into the XML file by the default OpenSearch 1.1 template. The XML specification provides the definitions of a group of search parameter names.

2.3. *Designing a Programmable CSE Query Element:* A specific search request that can be carried out by a search client can be defined using the OpenSearchQuery element. The search criteria in a URL template are matched by the characteristics of the query element. When necessary, additional custom parameters can be introduced via namespaces in addition to the basic set of search parameters, which are explicitly described as query attributes. In order for search clients to evaluate the search engine, authors should include at least one Query element with role="example" in each OpenSearch description document. In

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

order for search clients to do the same query again, search engines should include a Query element with the role="request" in each search response.

3. *Design of an Advanced Programmable CSE:* The third step of designing the programmable CSE is to create an advanced programmable CSE, which can be achieved by using the following algorithm (4).

Algorithm (4): Advanced CSE-XML

Input: XML File**.**
Output: Labeled Context XML File.
Begin
Step 1: Determine the XML file format that is appropriate for our needs.
Step 2: Determine the specification tags of the programmable CSE.
  Step 2.1: Defining a programmable CSE Structure (Context):
  Step 2.2: Defining Sites to Search (Annotations).
Step 3: Defining a programmable CSE export results.
  Step 3.1: Boosting Results with Keywords.
  Step 3.2: Changing Search Results with Labels.
  Step 3.3: Modulating the Effects of Labels.
End

3.1. *Determine the XML File Format Appropriate for Our Needs:* Before creating the programmable CSE, decide which format best suits our needs (based on step 1.2).

3.2. *Determine the Specification Tags of the Programmable CSE***:** In this step, the context of the XML file is defined to create a programmable CSE (based on Step 1.1).

3.3. *Defining a Programmable CSE Export Results:* Three different tuning options are available with programmable CSE: scores, weighted labels, and keywords. While the annotations file specifies the scores, the context file specifies the keywords and weights.

3.3.1. *Boosting Results with Keywords:* Using keywords is a quick way to increase the visibility of specific web pages in search results and to increase the number of search results for a given topic.

3.3.2. *Changing Search Results with Labels:* Weighted labels inform the Programmable Search Engine whether to promote, degrade, or exclude a site. Depending on the weights that are applied to the labels, it is possible to determine how much a site is boosted or demoted.

3.3.3. *Modulating the Effects of Labels:* Annotations are given scores that can reduce or even eliminate the impact of the weighted labels. They increase the ranking's level of granularity by another level.

4. *Create and Edit Programmable CSE Files***:** The fourth step of designing a Programmable CSE is editing the programmable CSE Files. In this step, to design an edited XML file, the following algorithm (5) is implemented:

Algorithm (5): Edited CSE - XML

Input: labeled Context XML File**.**
Output: Edited labeled Context XML File.
Begin
Step 1: Using the advanced XML_CSE algorithm, create a context XML file.
Step 2: Change the UNIX-style in the XML file by saving it with the XML extension.xml (for example, cx_global.xml).
Step 3: Make a copy of the created XML file and debug it by recreating the programmable CSE.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

Step 4: Edit the XML file by changing the Unicode text file format.
End

B. Enable Search Queries in the Programmable CSE

The second step of the proposed system is implementing the search engine query after the programmable CSE is created. In this step, the element of the programmable CSE is added to the XML file. To do that, some code must be copied and pasted into the HTML file where the search engine is called. The following steps illustrate the main algorithm (6) used to implement the searching query in the programmable CSE.

Algorithm (6): CSE Query

Input: Edited labeled Context XML File, CSE- ID, and User Query**.**
Output: HTML Page.
Begin
Step 1: In the Control Panel, click on the search engine that is used as a programmable CSE.
Step 2: On the sidebar, click the Setup section, then the Basic tab.
Step 3: In the details section, click on the Get code section and paste the designed code into the HTML page source code where the programmable CSE Element appears.
End

1. *Design and Implement the Search Box in the XML File:* The following example shows how the search element, including the search query box and search results, will be displayed:

```
<div class="gcse-search"></div>
```

2. *Enabling XML Search Autocomplete:* In addition to changing the colors, fonts, and link styles in the Control Panel, custom HTML properties may be used to modify the search box's appearance and functionality. This enables some of the default Control Panel settings to be modified. It is especially helpful when one search box on the website operates differently from the others (like the one on the homepage). For instance, the enableAutoComplete parameter can enable or disable the autocomplete capability. If autocomplete is turned on in the Control Panel, it is set to true by default. The way the element acts can be altered by setting the value to false.

3. *Customizing XML File Search Results:* The search box can be customized in a manner similar to adding extra options to the search results element. For instance, the defaultToImageSearch attribute can be used to switch the search engine from one that relies on online results to one that relies on images. The Control Panel for the search engine must first have Image Search enabled. Click the Image search option to the "on" position on the Basics tab of the Setup section. Then, on the website, add the defaultToImageSearch attribute to the element.

C. Design a Programmable CSE Based on API-Definition

In this step, the API will be called. API refers to the context of the word "application," which means any software with a distinct function. Using the API to build the programmable CSE has a very important role because it acts as an interface between the created programmable CSE and the services provider. In other words, it is a contract of service between the two applications. The contract illustrates how the two applications can deal with each other based on the requests and responses. In terms of designing a programmable CSE-based API, the designed programmable CSE application should store the tokens (API objects) that allow access to the Google API. Once the access token (key)

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

has expired, the designed programmable CSE will not be able to access the Google API service anymore, and a new one (key) has to be obtained. The general scenario of the web server application-based API is illustrated in *Fig. 2*.

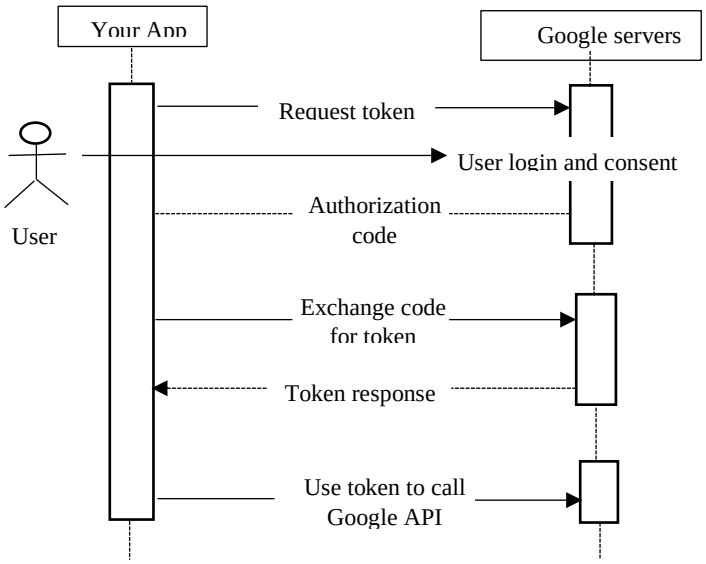

FIG. 2. API WEB SERVER APPLICATION.

- The authorization process starts with the application.
- The application starts with a browser (Google URL). The Google URL includes the query parameters, which indicate the access type.
- Google handles the user's authentication request by selecting the user's consent.
- The authentication code is a response to a request for authentication. Using an access token and a refresh token, the access application can now make an exchange.
- A Google API must be accessed using the application's access token, and the refresh token must be saved for future usage. A new access token is obtained by the program when the old one has expired using the refresh token.

By accessing the API, the user can ask the current search engine to access its database and carry out requests with it, such as importing the link data, carrying out searches, and exporting the link data, etc. The following steps must be taken in order to determine the API ID number before using the API in the programmable CSE:

Algorithm (7): API Key Request

Input: CSE Name, CSE-ID**.**
Output: API Key.
Begin
Step 1: Create the programmable CSE-based API key.
Step 1.1: Use the Control Panel and sign in using your Google account.
Step 1.2: In the Sites to Search section, include the programmable CSE that needs to access the database. The URL pattern can be included in the XML file.
Step 1.3: From the programmable CSE Navigate, click on Add to request a new API key.
Step 1.4: Select the configuration file of the programmable CSE Navigate by clicking on Adding to request a programmable CSE and clicking on the Create button.
Step 1.5: confirm the configuration file of the programmable CSE and extend it to search the entire web.
Step 2: Enable the entire webpage search by modifying the programmable CSE.
Step 2.1: Delete the existing URL, and toggle on the "Search the entire web" option.
Step 3: Get your search engine API key.
Step 3.1: An API key is needed to use the Tailored Searching JSON API.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

Step 3.2: Navigate to the customized searching JSON API page and click Get a Key.
Step 3.3: The search engine ID can be found at the end of the page.
Step 3.4: Choose an existing project or create a new one and click Next.
Step 3.5: Copy and export your API key.
End

1. *Determine the API Key for the Programmable CSE:* The Programmable CSE-based JSON API requires an API key described in the algorithm (7). A client can be identified by Google via an API key. The application can add the query parameter key=your API key to all request URLs after you have an API key. The API key does not require any encoding and is secure for embedding in URLs.

2. *Data Modeling for Search Engine API Customization:* This step is based on making a request by invoking the API-based XML file using Representational State Transfer (REST). This step describes the data modeling of the programmable CSE as well as how the JSON API based on the programmable CSE is used with REST.

2.1. *Making a Request:* REST is a platform that describes the software architecture style and a uniform interface between the decoupled components in the internal and client-server architecture. Instead of offering access to resources, the programmable CSE-based JSON API offers services for accessing resources. As a result of this step, the API returns a single URL that serves as a serving endpoint. With access to API services, we can send an HTTP tag with a GET request in the programmable XML file to receive the results for any search. For every search request, the following three queries [parameters] are necessary:

- API key: To identify the application, use the key query parameter
- Programmable Search Engine ID: Enter the programmable CSE's ID in the cx format to carry out this search. The Control Panel must be used to create the search engine. Note: There are various formats available for the Search Engine ID (cx) (e.g. 8ac1ab64606d234f1).
- Search query: The q query parameter can be used to specify a search expression.

2.2. *Design Query Parameters:* When designing any query through the XML file, there are two different types of parameters that can be included in the request:

- API-specific parameters: The search's parameters must be specified, such as the expression, the number of results, the language, etc.
- Standard query parameters**:** Specify the request's technical details, such as the API key.

2.3. *Data Response:* The server sends back a 200 OK HTTP status code and the response data in JSON format if the request was successful.

D. Design a Programmable CSE Using a JSON File

The JSON file will be extracted in this step. An application may be developed to retrieve and display search results from the programmable CSE using the JSON file that provides the web pages metadata. The API allows us to submit a full request to obtain a JSON file format containing the web pages metadata.

1. *Extract Metadata from the Programmable CSE Based on JSON API:* A JSON file is imported as a result of the search query after the search query is implemented on the programmable CSE based on JSON API. The JSON file typically contains three categories of metadata:

- Metadata describing the requested search (and, possibly, related search requests)
- Metadata describing the programmable search engine

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

- Metadata describing search results.

2. *Extract the Metadata Objects from the XML File in the Programmable CSE:* In this step, the important metadata objects are labeled and extracted from the XML file. To understand this step, three sub-steps must be implemented:

2.1. *Searching for Request Metadata Extraction:* In this step, the main objects of the search request of the metadata are extracted, which are represented by the following properties:

- URL property: This field contains details about the OpenSearch template that was applied to the results returned by this request.
- Queries property: which is an array of objects outlining various search criteria. One of the two custom roles defined by this API, previous page or next page, or the name of an OpenSearch query role, appears as the name of each object in the array. Several potential query role items are:

❖ Request: Metadata about the query that will be used to retrieve the current page's results.
❖ Next Page: Metadata about the query that will be used to retrieve the next page's results.
❖ Previous Page: Metadata about the query that will be used to retrieve the previous page's results.

2.2. *Searching for Search Engine Metadata Extraction:* In this step, the metadata describing the search engine is extracted. In this object of the metadata, the context property is extracted from the XML file, which provides the refining of the search engine.

2.3. *Searching for Search Results Metadata Extraction:* In this step, the main object of the item array contains the actual search results. In the XML file, the metadata of the search results includes the URL title as well as the text snippets that describe all the search results. In the search results, the metadata object promotions property in the XML file that contains the searching promotion is extracted too.

VII. THE IMPLEMENTATION OF THE PRPOSED SYSTEM

The implementation of the proposed system has two steps: The first will be used when the user wants to select a query, and the second will be used to scrape the metadata.

A. Query Selection Step

This step will be as follows: The main window of the query selection will be shown in *Fig. 3*. In this figure, once the user types the desired keyword, such as "data mining," into the text box, the metadata scraping step will be carried out.

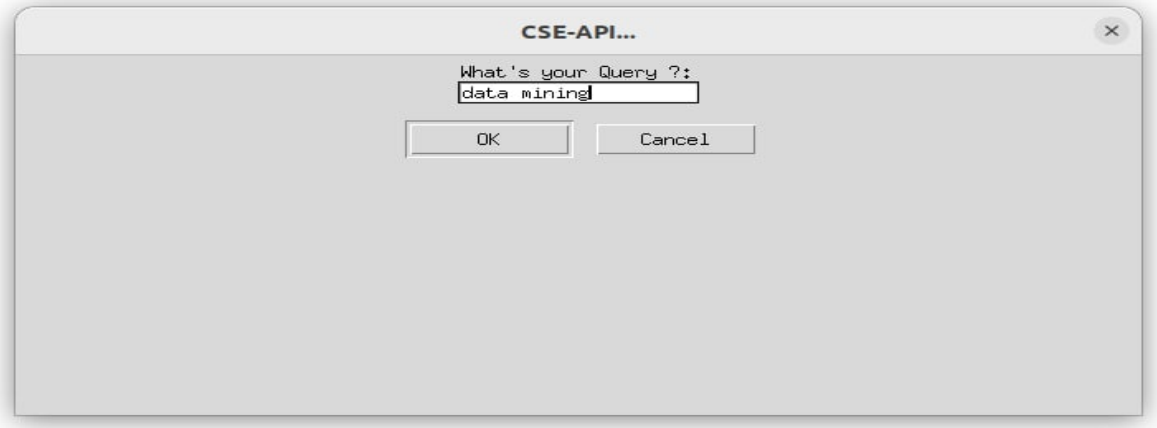

FIG. 3. THE MAIN WINDOW OF THE QUERY SELECTION.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

B. Metadata Scraping Step

This step will be as follows: The main window of the metadata scraping will be shown in *Fig. 4.*

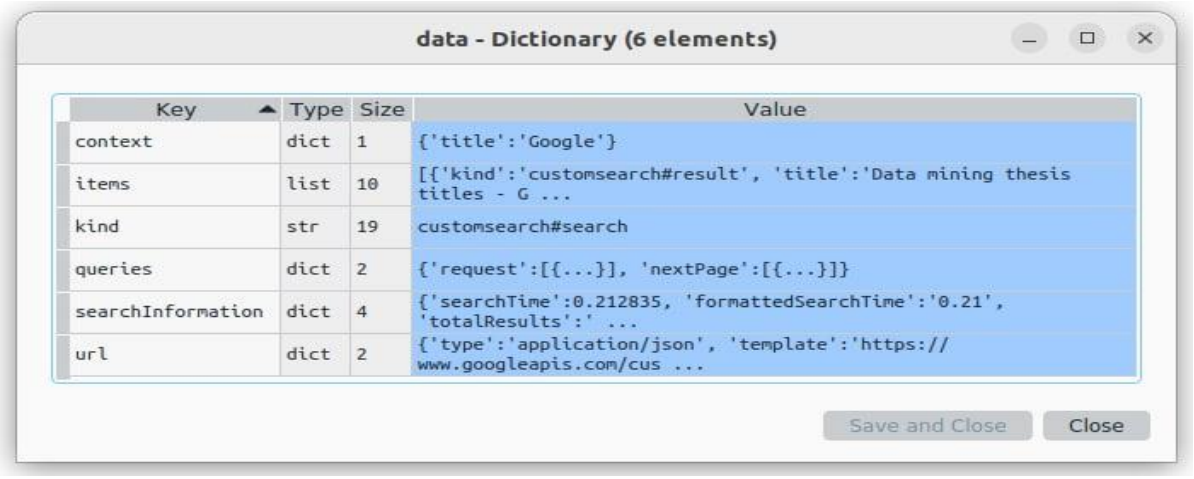

FIG. 4. THE MAIN WINDOW OF THE METADATA SCRAPING.

After the search query is implemented by the user, a JSON file is imported as a result of the search query. The JSON file that contains the websites' metadata lets the user retrieve and display search results. By clicking on the items key in *Fig. 4*, the custom search results will be displayed as shown in *Fig. 5*.

FIG. 5. THE MAIN WINDOW OF THE CUSTOM SEARCH RESULTS.

If the user clicks on any link of the custom search results in *Fig.* 5, then the metadata of the custom search results will be displayed as shown in *Fig.* 6.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

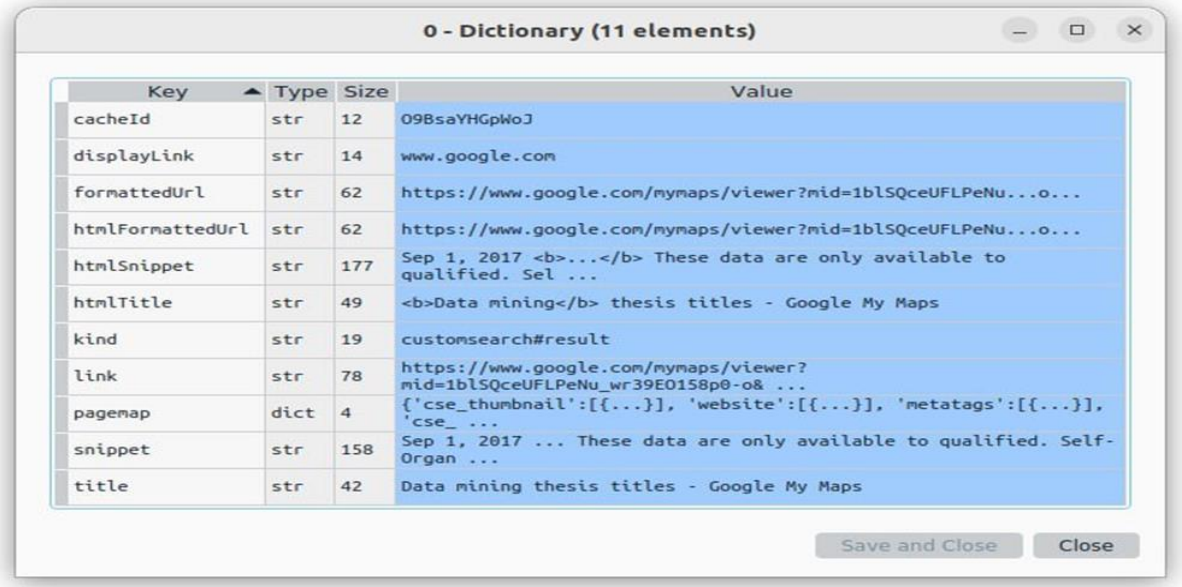

FIG. 6. THE METADATA OF THE CUSTOM SEARCH RESULTS.

When the page map key is clicked by the user in *Fig.* 6, the further metadata of the custom search results will be presented as shown in *Fig.* 7.

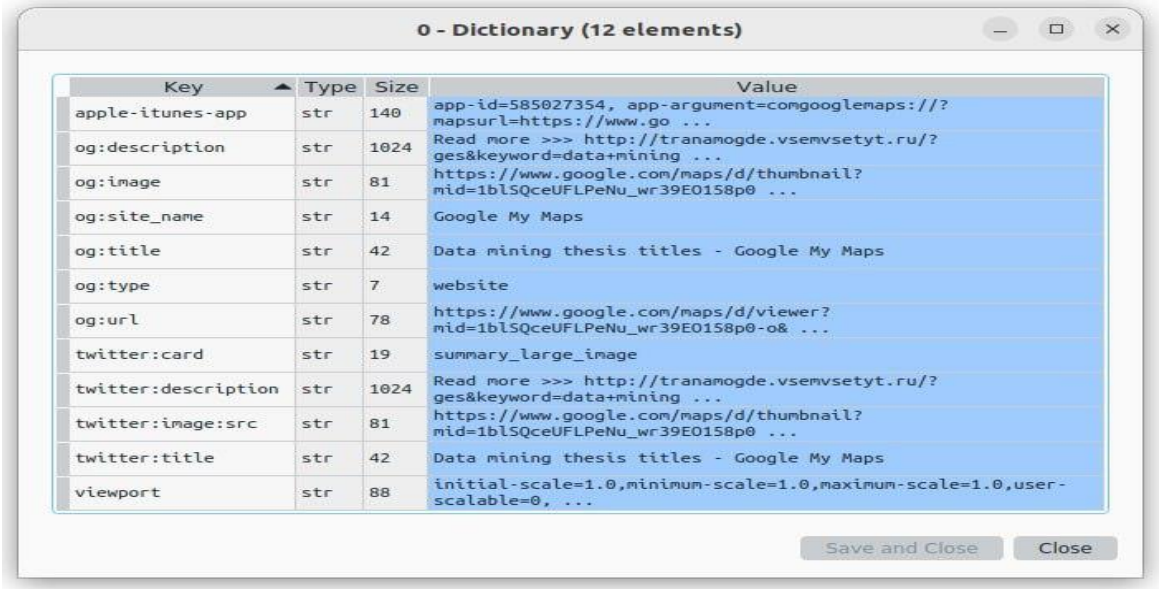

FIG. 7. THE FURTHER METADATA OF THE CUSTOM SEARCH RESULTS.

VIII. CONCLUSIONS

Data about data is known as "metadata." Google and other search engines do not "read" the content (data). They index it by looking for information in predefined fields of metadata. This paper proposed a methodology for metadata scraping using a programmable (CSE) system which can extract metadata from web pages (HTML pages) in the Google database and save it in an XML format for later analysis and retrieval. There are four steps in the proposed system. A virtual CSE will be built in the first step by defining the engine in an XML file. The second step involves implementing the search engine query in programmable CSE. The programmable CSE requires an API key in the third step. Google

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

can identify the client by using the API key. The user can ask the current search engine to do requests with it, such as importing the link data, performing searches, exporting the link data, and other operations, by requesting access to the API. The final step of the proposed system will import a JSON file as a result of the search query; the important metadata objects are labeled and stored in the XML file for later analysis and retrieval. Although the majority of currently available resources lack this kind of information, documents that contain metadata are a relatively recent phenomenon on the web and increase the likelihood that users will find the information they need.

IX. COMPARISON WITH RELATED WOEKS

The benefits and drawbacks of each scraping method is highlighted in Table I.

TABLE I. COMPARISON PROPOSED METHOD WITH RELATED WORKS

Name of Author	Scraping approach	Ease to install	Performance	Ease of use
Achmad M. et al.[4] in 2018	Regular expressions	Easy (built-in module)	Fast	Hard
Fatmasari et al.[5] in 2018	Beautiful Soup	Easy (pure Python)	Slow	Easy
Lanny A. and Srujan K. [6] in 2021	Lxml	Moderately difficult	Fast	Easy
Proposed method	API	Easy	Fast	Easy

REFERENCES

[1] I. S. H. Almaqbali, F. M. A. Al Khufairi, M. S. Khan, A. Z. Bhat, and I. Ahmed, "Web Scrapping: Data Extraction from Websites," *J. Student Res.*, pp. 1–4, 2020, doi: 10.47611/jsr.vi.942.

[2] M. Dogucu and M. Çetinkaya-Rundel, "Web Scraping in the Statistics and Data Science Curriculum: Challenges and Opportunities," *J. Stat. Educ.*, vol. 0, no. 0, pp. 1–24, 2020, doi: 10.1080/10691898.2020.1787116.

[3] P. Thota and E. Ramez, "Web Scraping of COVID-19 News Stories to Create Datasets for Sentiment and Emotion Analysis," *ACM Int. Conf. Proceeding Ser.*, pp. 306–314, 2021, doi: 10.1145/3453892.3461333.

[4] A. Maududie, W. E. Y. Retnani, and M. A. Rohim, "An Approach of Web Scraping on News Website based on Regular Expression," *Proc. - 2nd East Indones. Conf. Comput. Inf. Technol. Internet Things Ind. EIConCIT 2018*, pp. 203–207, 2018, doi: 10.1109/EIConCIT.2018.8878550.

[5] Fatmasari, Y. N. Kunang, and S. D. Purnamasari, "Web Scraping Techniques to Collect Weather Data in South Sumatera," *Proc. 2018 Int. Conf. Electr. Eng. Comput. Sci. ICECOS 2018*, no. December, pp. 385–390, 2019, doi: 10.1109/ICECOS.2018.8605202.

[6] L. A. Rodrigues and S. Kumar Polepally, "Degree Project Level: Second-Cycle Creating Financial Database for Education and Research: Using WEB SCRAPING Technique," 2021.

[7] N. Perlin, "Introduction to metadata," *IEEE Int. Prof. Commun. Conf.*, pp. 153–155, 2020, doi: 10.1109/IPCC.2006.320378.

[8] L. Woolcott, *Understanding Metadata: What is Metadata, and What is it For? , Primer Publication of the National Information Standards Organization* , vol. 55, no. 7–8. 2017. doi: 10.1080/01639374.2017.1358232.

[9] W. Underwood, "Automatic Extraction of Dublin Core Metadata from Presidential E-records," *Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020*, pp. 1931–1938, 2020, doi: 10.1109/BigData50022.2020.9377943.

[10] E. Méndez, F. Crestani, C. Ribeiro, G. David, and J. C. Lopes, "Metadata Synthesis and Updates on Collections Harvested using the Open Archive Initiative Protocol for Metadata Harvesting," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11057 LNCS, no. September 2018, pp. v–vi, 2018, doi: 10.1007/978-3-030-00066-0.

[11] R. Turgunbaev, "Metadata in Data Search,", International Conference on Applied Sciences,no. September, 2022.

[12] C. SINGH, "Dynamic clustering for web mining," p. 2019, 2019.

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.23.3.2

[13] Jyoti Mor, "WEB CRAWLER AND SEARCH SYSTEM ARCHITECTURES FOR INFORMATION RETRIEVAL FROM WEB," no. 12001971, p. 12001971, 2021.

[14] Kompal, "Improving the Performance of Web Crawler For Effective Information Retrieval," no. August 2020, p. 2020.

[15] H. Phan, "Building Application Powered by Web Scraping," no. March, 2019, [Online]. Available: https://www.theseus.fi/handle/10024/166489

[16] M. Gheorghe, F.-C. Mihai, and M. Dârdală, "Modern techniques of web scraping for data scientists," *Rev. Rom. Interactiune Om-Calculator*, vol. 11, no. 1, pp. 63–75, 2018.

[17] E. Persson, "Evaluating tools and techniques for web scraping," *Degree Proj. Comput. Sci. Eng.*, vol. SECOND CYC, pp. 1–97, 2019, [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1415998/FULLTEXT01.pdf

[18] A. V Saurkar and S. A. Gode, "An Overview On Web Scraping Techniques And Tools," *Int. J. Futur. Revolut. Comput. Sci. Commun. Eng.*, vol. 4, no. 4, pp. 363–367, 2018, [Online]. Available: http://www.ijfrcsce.org/index.php/ijfrcsce/article/view/1529

[19] B. Zhao, "Web Scraping," *Encycl. Big Data*, no. December, 2020, doi: 10.1007/978-3-319-32001-4.

[20] R. Vording, "Harvesting unstructured data in heterogenous business environments ; exploring modern web scraping technologies, 34thTwente Student Conference on IT, January. 29th, 2021, pp. 1–9,2021

Received 30/October/2022; Accepted 10/January/2023
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*