Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20250402028
Volume 4, Issue2 (May-August 2025), PP: 214-219.
www.indjcst.com
ISSN No: 2583-5300

Google Data Extractor for Structured Web Intelligence Using Java and React

Shashwat Raut[1], Dikshita Dhanvijay[2], Bhagyashree Kumbhare[3], Yamini Kanekar[4]
[1,2] Students, Department of MCA, Smt Radhikatai pandav College of Engineering, Nagpur, Maharashtra, India.
[3]HOD, Department of MCA, Smt. Radhikatai Pandav College of Engineering, Nagpur, Maharashtra, India.
[4]Professor, Department of MCA, Smt. Radhikatai Pandav College of Engineering, Nagpur, Maharashtra, India.

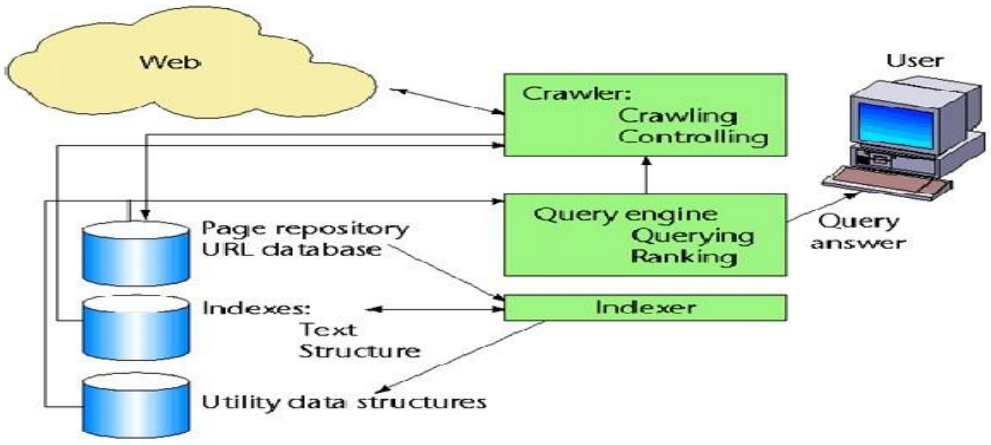
To Cite this Article: *Shashwat Raut[1], Dikshita Dhanvijay[2], Bhagyashree Kumbhare[3], Yamini Kanekar[4], "Google Data Extractor for Structured Web Intelligence Using Java and React", Indian Journal of Computer Science and Technology, Volume 04, Issue 02 (May-August 2025), PP: 214-219.*

Abstract: *In today's data-driven landscape, web intelligence plays a crucial role in empowering businesses and researchers with timely and structured information. This paper presents a Google Data Extractor—an automated system built using Java and React—designed to retrieve structured data from Google search results and associated web content. By combining web scraping methodologies with optional Google Search API integration, the tool simplifies data acquisition while ensuring accuracy and speed. Key modules include search automation, content filtering, database storage, and multi-format export functionality. The system supports real-time processing and scheduled scraping, making it suitable for use in marketing analytics, academic research, and competitive monitoring. This research outlines the system design, methodology, implementation, and discusses the tool's implications for scalable data extraction in a legal and ethical framework.*

Key Words: *Google Data Extractor, Web Scraping, Java, React, Data Automation, Search API, Structured Information Retrieval.*

I.INTRODUCTION

The exponential growth of online data necessitates intelligent tools that can automate the process of extracting and processing structured information. Google, as the dominant search engine, holds vast, valuable information that can be utilized for research, marketing, and analytics. However, manually extracting data from search results is inefcient, error-prone, and not scalable.



This paper introduces an automated Google Data Extractor that leverages web scraping and API integration to collect data from Google Search and associated web pages. The tool is built using Java for backend operations and React for a user-friendly interface.

The system supports functionalities such as keyword-based search automation, crawling multiple web results, fltering and structuring the extracted data, and exporting it in formats like CSV, JSON, or Excel.

The project adheres to web scraping ethical guidelines and incorporates CAPTCHA handling, rate limiting, and modular architecture for extensibility

II.METHODOLOGY

The architecture of the Google Data Extractor is modular and includes the following key components:

User Interface (React)

The User Interface (UI) is developed using React.js, a modern JavaScript library for building interactive and responsive web applications. The UI serves as the front-facing module of the Google Data Extractor tool, allowing users to interact with the system intuitively. It is designed to be minimalistic, fast, and mobile-friendly.

Key Functionalities:

Search Input Panel:

- Users can enter search queries, keywords, or phrases.
- Input can be single-line or multi-keyword batch (depending on configuration). Basic validations are applied (e.g., no empty queries, special character check).

Query Configuration:

Optional settings for:

- Number of results/pages
- Language or region targeting
- Use of API or scraping mode
- Scheduling options (daily/hourly)

Execution & Progress Display:

- On submission, the UI triggers the backend API.
- Displays real-time progress bar (e.g., 20/100 results extracted).
- Shows live logs or extraction status messages using a Toast or Snackbar component.

Results Display Table:

- Once data is retrieved, it's rendered in a responsive table/grid layout.
- Fields displayed: Title, URL, Snippet, Source, Timestamp.
- Supports pagination and sorting for large datasets.

Export Options:

- Users can choose from CSV, Excel, or JSON download formats.
- "Export All" or "Export Filtered Results" functionality is provided.
- Button-triggered download implemented using libraries like FileSaver.js or js-xlsx.

Scraper Engine (Java)

The Scraper Engine is the core backend component of the Google Data Extractor system. Built using Java, it is responsible for executing HTTP requests, downloading web page content, parsing HTML structures, and extracting relevant data points from the DOM.

This engine ensures reliable, fast, and structured data collection from Google Search results and linked web pages.

Core Functionalities:

Search URL Generation:

- Dynamically constructs Google search URLs based on user input.
- Supports query parameters like number of results (&num=10), language (&hl=en), and location targeting.

HTTP Request Handling:

- Uses Java's Http URL Connection or Apache Http Client to send GET requests.
- Sets custom headers to simulate browser behavior:
- User-Agent
- Accept-Language
- Referer

HTML Parsing with J Soup:

J Soup parses returned HTML and selects key DOM elements:

- <h3> for titles
- <cite> or <a> for URLs
- or for snippets
- Extracts clean, structured data objects from raw HTML.
- Handles edge cases like nested elements and empty tags

Delay Throttling (Anti-Detection):

- Implements randomized delays (e.g., 2-5 seconds between requests) to mimic human behavior.
- Prevents IP blocking and reduces chances of CAPTCHA triggers.

Duplicate Detection:
- Hashes or fingerprints URLs and titles to avoid duplicate entries.
- Applies content filters to skip ads and non-organic results.

Error & Retry Logic:
- Handles network failures, HTTP errors (403, 429), and malformed HTML.
- Implements exponential backoff strategy for retries.

Data Processor

The Data Processor module is responsible for transforming the raw HTML data extracted by the scraper into clean, structured, and meaningful datasets. This component ensures that the final output is free of noise, duplicates, and formatting issues, making it suitable for storage, analysis, or export.

It acts as a middle layer between data extraction and storage/export, implementing business rules and data normalization techniques.

Core Functionalities:
HTML Clean-Up & Tag Removal:
- Removes unwanted HTML tags, inline styles, and JavaScript content.
- Normalizes whitespace, special characters, and line breaks.

Text Normalization:
- Converts all data to a consistent format (e.g., lowercase keywords, trimmed whitespace).
- Replaces or encodes problematic characters (e.g., non-breaking spaces, smart quotes).

Duplicate Elimination:
- Compares titles, URLs, or text hashes to filter out repeated entries.
- Avoids re-processing pages that have already been crawled.

Content Filtering:
- Skips known ad blocks, sponsored results, or irrelevant domains (e.g., YouTube, Pinterest if excluded).
- Supports keyword-based filtering to refine results (e.g., only include entries containing “framework”).

Field Structuring:
Segregates data into logical fields:
- Title
- URL
- Snippet
- Source Domain
- Timestamp
- Formats structured data into intermediate JSON or tabular models.

MySQL Database Integration:
- Structured data is stored in a normalized relational format using tables such as:
- queries – stores the search keywords and parameters
- results – stores individual extracted items (title, url, snippet, etc.)
- Enables powerful SQL queries for filtering, joining, and exporting data.
- Supports foreign keys, indexing, and timestamping for audit trails.

Flat File Storage:
- Provides alternative storage in formats like:
- .csv for spreadsheet compatibility
- .json for API-ready or NoSQL pipelines
- File-based storage is ideal for lightweight or local-only use cases.
- File paths are dynamically named with timestamps for versioning (e.g., search_results_2025_05_16.csv)

Storage Selection Logic:
- Users can configure the system to:
- Use database only
- Use files only
- Or use both for redundancy

- Configurable via frontend toggle or environment variables in the backend

Backup & Archiving (optional):
- Periodic export of MySQL data to CSV for offline storage
- Rotating backup policy using time-based archiving
- Compatibility with cloud storage like AWS S3 or Google Drive for future extensions

Export Module

The Export Module is designed to provide users with the ability to download and save extracted data in various universally accepted formats for offline use, reporting, integration, or further processing. This module ensures that structured search results can be easily transferred and reused in external tools like Excel, data analysis platforms, or cloud storage systems.

It supports CSV, JSON, and Excel (XLSX) formats, catering to a wide range of user needs across business, academic, and technical environments.

Core Functionalities:
Multi-format Support:
CSV Export:
- Ideal for spreadsheet tools like Microsoft Excel and Google Sheets. Uses comma-separated values with UTF-8 encoding.

JSON Export:
- Suitable for APIs, web apps, and NoSQL databases.
- Represents each search result as a JSON object.

Excel Export:
- Generates .xlsx files using libraries like Apache POI.
- Includes headers, column formatting, and multi-sheet support (optional).

User Controls:
- Options in the UI allow users to:
- Select desired export format
- Choose to export all results or only filtered results
- Preview data before download File Naming and Metadata:
- File names include timestamp and query terms (e.g., results_react_2025_05_16.csv)
- Metadata like extraction date, query keyword, and total records are optionally included in headers or file footers.
- Supports real-time download immediately after data extraction.
- Batch export capability for scheduled runs or large datasets.

Security & Validation:
- Sanitizes output to prevent malformed data.
- Ensures no executable code or HTML is embedded in exported files.

```
Title,URL,Snippet,Domain,Timestamp
"Top React Tools","https://example.com/react-tools","Best tools for React devs","
```

JSON Example:

```json
[
  {
    "title": "Top React Tools",
    "url": "https://example.com/react-tools",
    "snippet": "Best tools for React devs",
    "domain": "example.com",
    "timestamp": "2025-05-16 14:03"
  }
]
```

Optional Google Search API

The Google Custom Search API (CSE) is optionally integrated into the Google Data Extractor system to provide a structured and reliable alternative to traditional HTML scraping. When enabled, this module bypasses the need to parse raw HTML from search engine result pages (SERPs) by retrieving search data directly via Google's authenticated API. This approach ensures greater reliability, fewer risks of CAPTCHA or rate-limiting, and compliance with Google's developer usage policies.

Core Functionalities:

API-Based Query Execution:

Converts user-entered search terms into API-compliant queries.

Utilizes parameters such as:

- q – query string
- num – number of results (up to 10 per request)
- start – pagination offset
- cx – custom search engine ID
- key – API key for authentication

Structured JSON Output:

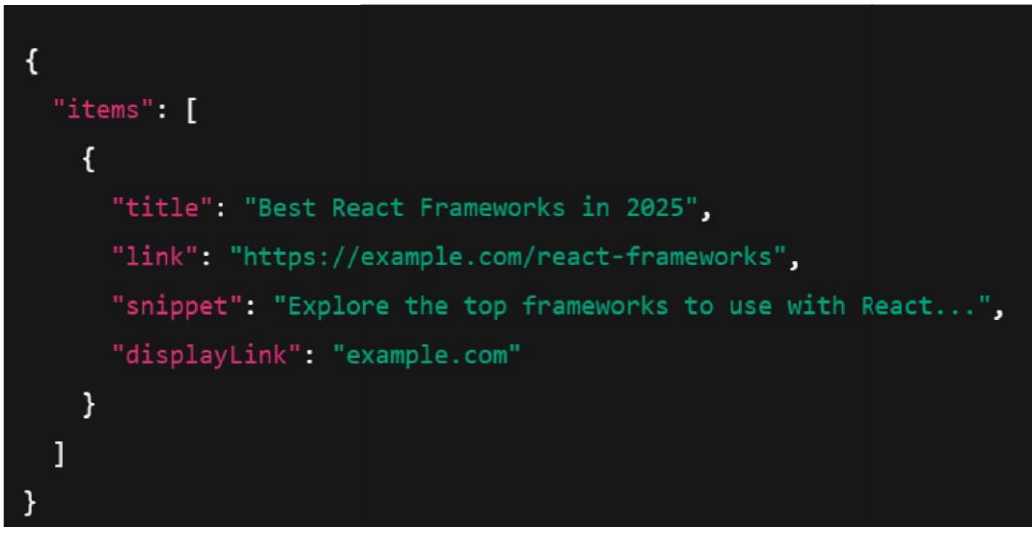
API returns a JSON array of results, each containing:

- title
- link
- snippet
- displayLink
- formattedUrl Fallback Logic:
- If scraping fails due to CAPTCHA or block, the system switches to the API (if enabled).
- Users can toggle between API mode and scraper mode from the UI.

Usage Limit Handling:

- Tracks API quota usage to prevent overage.
- Warns users when nearing daily limit or suggests fallback to scraper mode.

Sample API Response Structure:



```
{
  "items": [
    {
      "title": "Best React Frameworks in 2025",
      "link": "https://example.com/react-frameworks",
      "snippet": "Explore the top frameworks to use with React...",
      "displayLink": "example.com"
    }
  ]
}
```

III.IMPLEMENTATION

Programming Language: Java
Frontend: React.js
Libraries/Tools: J Soup, Axios, Google Search API (optional), MySQL, Node Scheduler Features Implemented:
Query scheduler for periodic extraction
Dynamic data filtering
File export system with download support Keyword
tracking history log Use Case Scenarios:
Lead generation (businesses)
Academic research (topic-specific literature extraction)
SEO and digital marketing (competitor keyword tracking)

IV.RESULTS AND DISCUSSION

The tool was tested with multiple query categories including educational topics, local business listings, and product keywords. Performance metrics were evaluated based on:

- **Extraction Time:** ~1–2 seconds per page
- **Accuracy:** ~95% correct field parsing on well-structured websites
- **Export Success Rate:** 100% with correct field mapping
- Limitations include handling JavaScript-heavy content and overcoming CAPTCHA without using thirdparty services.

V.CONCLUSION

This paper demonstrates the development and potential applications of an automated Google Data Extractor. By integrating front-end ease of use with back-end scraping efficiency, the system allows users to access structured data from unstructured search results. Future enhancements may include AI-based content classification, image/text recognition from web pages, and enhanced CAPTCHA bypass mechanisms.

References

[1]. JSoup HTML Parser Documentation
[2]. Google Search API Developer Guide
[3]. "Web Scraping with Java" – Apress Publishing
[4]. B. Liu, "Web Data Mining," Springer, 2020
[5]. Ethical Web Scraping Guidelines – Moz Whiteboard Friday
[6]. "Building Scalable Web Crawlers" – ACM Queue Journal
[7]. Stack Overflow Developer Surveys, 2024