# Development Team Project: Project Report

## 1. Introduction

The increasing reliance on digital platforms for academic research necessitates automation, to enhance efficiency (Krotov and Johnson, 2023). Researchers often spend significant time manually searching, extracting, and organizing academic resources (Gusenbauer and Haddaway, 2021). This proposal outlines the design and implementation of an automated system that retrieves, processes, and stores research data from online sources based on researcher/user queries. The system will streamline academic research by automating searches, extracting relevant metadata, and storing results in a structured format for offline use.

The proposed solution will automate research article data search by extracting key metadata such as title, authors, and publication details, while ensuring ethical compliance with data access policies. The system is designed to be scalable, reliable, and compliant with privacy regulations , making it a valuable tool for academic researchers.

## 2. System Requirements

The system will be developed in Python due to its powerful libraries for data extraction, processing, and storage (Kumar et al., 2024). The following libraries will be used:

- SerpApi: this library will be used to retrieve results from Google Scholar in a structured way. By selecting this library, compliance with anti-scraping policies will be ensured, preventing any restriction on automated data extraction;
- BeautifulSoup: Used for parsing HTML documents when needed. BeautifulSoup was chosen over Scrapy because it is lightweight, easy to implement, and better suited for handling static HTML content (Kumar et al., 2024). While Scrapy is more powerful for large-scale web crawling, it comes with a steeper learning curve and is better suited for handling multiple concurrent requests, which is unnecessary for this project (Mitchell, 2018)
- Pandas: this library will be used for the data structure and storage.
- Streamlit will be used for building the web-based GUI for the system.
- SQLite will be used as the datastore, for structured storage and retrieval.

## 3. Development methodology

### 3.1. Agent-Based System Approach

For this project, we propose building the ResearchAgent as a hybrid agent that combines structured decision-making with real-time adaptability. This approach allows the agent to intelligently manage research tasks while reacting dynamically to external conditions, such as API failures or variations in web page structures. The ResearchAgent follows a structured process to determine the best method for retrieving data - first attempting API integration and switching to web scraping if needed. This makes it more flexible than a purely rule-based system, while still ensuring that data retrieval is structured and predictable.

### 3.2. Justification for a Hybrid Agent Approach

We need an agent that can handle uncertain and dynamic environments - some research sources provide APIs, while others require web scraping. A purely symbolic reasoning agent would rely on predefined rules and struggle with unexpected failures, while a purely reactive agent would lack structured decision-making. For our agent to function effectively, it must be both reactive and proactive. One approach is to distribute these behaviours across distinct subsystems, enabling adaptability and goal-directed reasoning (Wooldridge, 2009).

Our approach balances both:

- Symbolic Reasoning: The agent decides between API calls and web scraping based on availability.

- Reactive Implementation: If an API fails or a web page structure changes, the agent adjusts its extraction method accordingly.

### 3.3. System Design Techniques

To ensure the system remains scalable, modular, and adaptable, we opted to adopt the following design approaches:

- Multi-Agent Communication

  The ResearchAgent acts as a central controller, managing search queries, data extraction, and storage.

  Each component (Search, Scraping, Extraction, and Storage) operates independently, allowing for easy updates, debugging, and optimisation. This modular design follows the principle of separation of concerns, which enhances maintainability (Ferber, 1999).

- Event-Driven Execution

  The system does not rely on predefined datasets but reacts dynamically to available data. If an API request succeeds, the system uses structured data. If the API fails, the agent automatically falls back to web scraping, ensuring continuity. By tracking its internal state and adapting decisions based on percepts, the agent continuously refines its understanding of data availability and structure, aligning with model-based reflex agent principles (Russell and Norvig, 2021).

# 4. System Design

## 4.1. Architecture Overview

The proposed system consists of multiple components, with the ResearchAgent acting as the central controller that coordinates tasks between different modules.

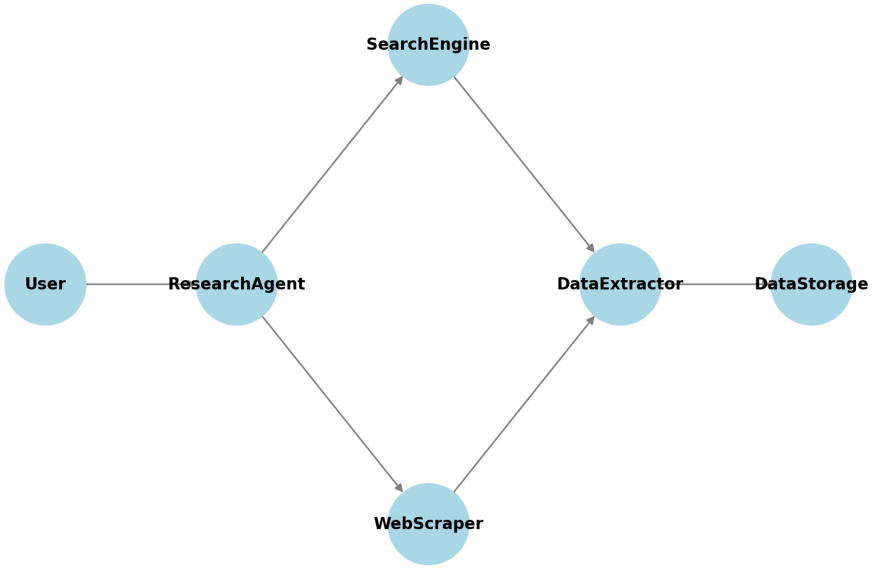


*Figure 1: System Architecture*

The architecture follows a modular approach, ensuring separation of concerns and ease of modification. The key components include:

- ResearchAgent: The core system component that orchestrates all tasks.
- SearchEngine Module: Interfaces with search APIs to find relevant sources.
- WebScraper Module: Extracts raw data from web pages.
- DataExtractor Module: Processes and structures extracted data.

- DataStorage Module: Saves the processed information for offline access.

## 4.2. UML Diagrams

- Class Diagram: Illustrates the relationships between key system components.

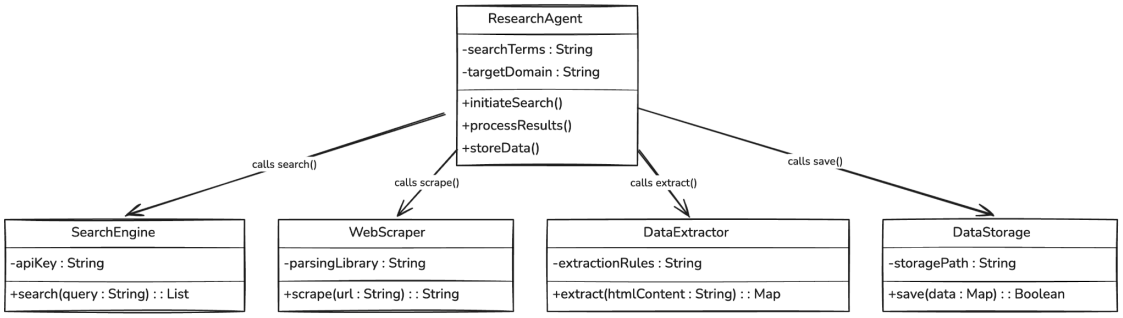


*Figure 2: Class Diagram*

- Sequence Diagram: Illustrates the flow from when a user initiates a research task.

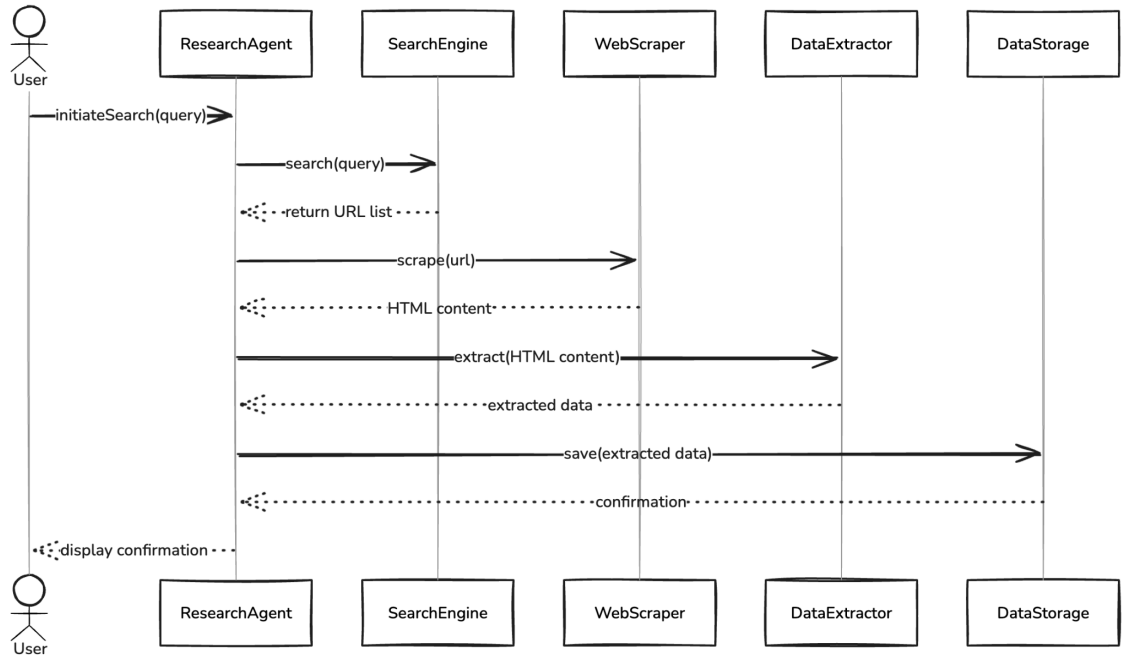


*Figure 3: Sequence Diagram*

This design ensures that each component operates independently while working collaboratively to achieve the system's research objectives.

## 5. Challenges and Mitigation Actions

Restrictions to automated data extraction will be one of the main challenges faced by the system. Data scraping is not allowed by the majority of the websites, and such activity may lead to access restrictions or even IP bans. To address this issue, the system will use the SepApi library that provides access to Google Scholar in compliance with their relevant policies (Krotov and Johnson, 2023).

Missing, incomplete or incorrectly formatted data is another challenge that will be addressed by the system. To do so, data cleaning and validation will be applied

in order to identify missing or incorrectly formatted fields; the Pandas library will be used for this.

To ensure compliance with data privacy laws, in particular the UK GDPR, the system will store only the metadata of the extracted data. The data will also only be accessible for academic research use, thus the system will have requirements for agreement to use within compliance to privacy regulations, before users can access the data. Additionally, references to the original sources will be provided to the users in order to preserve academic integrity (Buck and Ralston, 2021).

## 6. Conclusion

This research proposal presents a solution for automating academic research by enabling users to retrieve and store data in a structured format for subsequent analysis. By leveraging Python, SerpApi, and Pandas, the system improves research efficiency and effectiveness while ensuring compliance with applicable ethical and data protection regulations.

Considering the number of studies a researcher has to go through when producing his/her own piece of research, as well as the fact that academic research has become primarily digital, automation of the process can enhance its efficiency and effectiveness, this proposed system can contribute significantly towards that direction.

**References**


Buck, A., & Ralston, D. (2021). I didn't sign up for your research study: The ethics of using “public” data. Computers and Composition, 61, 102655.

Gusenbauer, M., & Haddaway, N. (2021). What every researcher should know about searching –clarified concepts, search advice, and an agenda to improve finding in academia. Research Synthesis Methods, 12(2), 136-147.

Krotov, V., & Johnson, L. (2023). Big web data: Challenges related to data, technology, legality, and ethics. Business Horizons, 66(4), 481-491.

Kumar, S., Varha, V., Sharup, Y., Ahamd, S., & Chandy, K. (2024). Leveraging Python for Web Scraping and Data Analysis: Applications, Challenges, and Future Directions. Frontiers in Collaborative Research, 2(1), 65-73.

Massimino, B. (2016). Accessing Online Data: Web-Crawling and Information-ScrapingTechniques to Automate the Assembly of Research Data. Journal of Business Logistics, 37(1), 34-42.

Mitchell, R. (2018) Web Scraping with Python: Collecting More Data from the Modern Web. Second edition. Sebastopol: O’Reilly Media, Incorporated.

Russell, S. and Norvig, P. (2021) Artificial Intelligence: a Modern Approach, Global Edition. Harlow, UNITED KINGDOM: Pearson Education, Limited. Available at: http://ebookcentral.proquest.com/lib/universityofessexebooks/detail.action?docID=6563568.

Wooldridge, M. (2009) ‘An Introduction to Multi-Agent Systems’. United Kingdom: John Wiley & Sons, Incorporated.