**Word count** (without Appendices and References): 892.
# **Report: Proposal for Automating Academic Research Solution.**

## 1. Introduction
Our biotechnology company has developed a new medical device concept. To scientifically prove its feasibility, a research team seeks an automated search solution to streamline the online academic research process. This proposal outlines an end-to-end solution to improve the research workflow created by our technical team.

## 2. Problem Statement
The current manual approach to academic research poses significant challenges and limitations. It is time-consuming and susceptible to errors, impeding the efficient validation of our medical device concept.

## 3. Proposed Solution Overview
Our proposed solution is a comprehensive and integrated system designed to enhance the speed and efficiency of academic research. Through the use of automation technologies and intelligent data processing methods, we aim to automate research tasks, employ advanced algorithms, and improve overall efficiency, accuracy, and productivity. This will enable faster and more efficient outcomes in the academic research process.

## 4. Solution Design
### 4.1. Design Description
Our Multi-Agent System (MAS) (Wooldridge, 2009) is a system composed of multiple autonomous agents that interact with each other to achieve specific goals. Our Solution Architecture presents three agents: Data Retrieval and Extraction Agent (Shah et al., 2022), Data Processing Agent, and Data Storage Agent. All agents are of the goal-based type (Russell & Norvig, 2021) working together in a coordinated manner, exchanging information, and collaborating to achieve the common goal of retrieving and processing relevant data for storage and further use (Bansall, 2023)

### 4.2 Solution Architecture
1. Data Retrieval and Extraction Agent
**Goal:** Retrieve and extract relevant information related to a user query.
**Actions:**
- Receives a user query as input.
- Utilises the Google Search API and Scale SERP API to search for data related to the query.
- Retrieves search results from the APIs.
- Extracts relevant information from the search results, including URL, title, summary, authors, publish date, keywords, and full text of articles.
- Handles errors and continues processing even if an error occurs during extraction.

**Behaviours:** The agent acts as a mediator between the user query and the external APIs, facilitating the retrieval and extraction of data.

2. Data Processing Agent:
**Goal:** Process the extracted information and enhance its quality and usefulness.
**Actions:**
- Receives the extracted information from the Data Retrieval Agent.
- Downloads articles referenced by the URLs.
- Performs processing tasks specific to the extracted data using the extract_content_v2 function.
- Parses the downloaded articles and extracts additional relevant information.

- Modifies the extracted data, such as joining author’s names, formatting publish dates, etc. (utils.py).
- Applies any required processing or transformations to enhance the data's quality and usefulness.

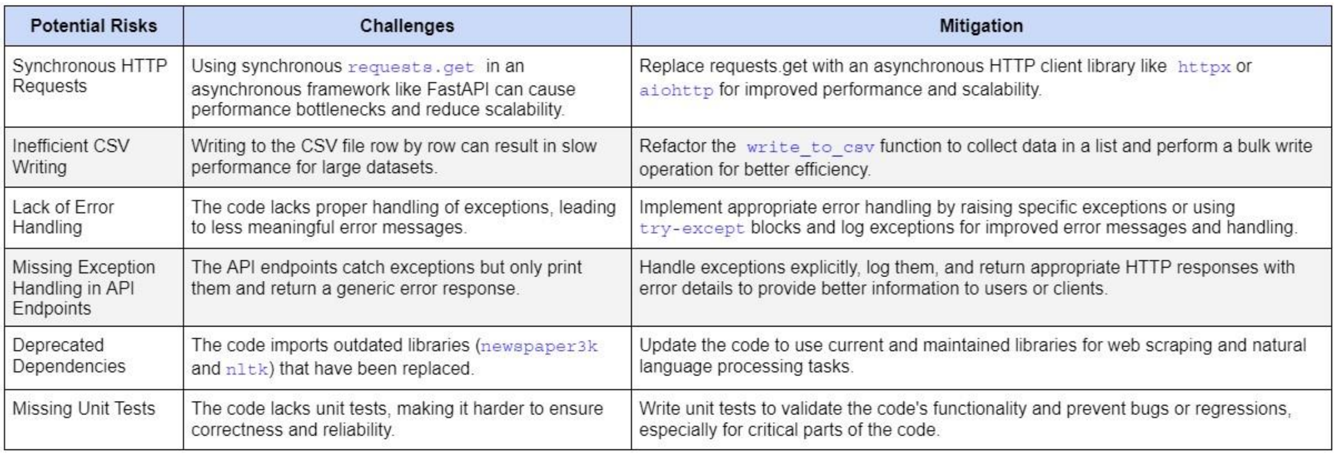
**Behaviours:** The agent takes the extracted information as input, performs various processing tasks, and applies transformations to improve the quality and usability of the data.

3. Data Storage Agent:
**Goal:** Store and provide access to the processed information.
**Actions:**
- Receives the processed information from the Data Processing Agent.
- Uses the write_to_csv function to save the processed data to a CSV file named "search_results.csv."
- Structures the data in a way that allows for easy access, retrieval, and analysis (URL, title, summary, authors, publish date, keywords, and full text of articles.)
- Provides an API with endpoints (/search) to access the processed information.
- Enables the presentation or consumption of the processed information by other applications.

**Behaviours**: The agent receives the processed information, stores it in a structured manner, provides API endpoints for access, and facilitates the utilisation of the processed data by other applications.

### 4.3 System Requirements
Programming language: Python
Libraries: os, pedantic, functools, dotenv, configs, scholarly, requests, json, fastapi, factories, models, utils, uvicorn, newspaper3k, nltk, csv, BeautifulSoup.

### 4.4. Visualisation
Sequence diagram presented in Appendix 1; Class diagram presented in Appendix 2; Activity diagram presented in Appendix 3.

## 5. Risk Assessment


| Potential Risks | Challenges | Mitigation |
|---|---|---|
| Synchronous HTTP Requests | Using synchronous `requests.get` in an asynchronous framework like FastAPI can cause performance bottlenecks and reduce scalability. | Replace requests.get with an asynchronous HTTP client library like `httpx` or `aiohttp` for improved performance and scalability. |
| Inefficient CSV Writing | Writing to the CSV file row by row can result in slow performance for large datasets. | Refactor the `write_to_csv` function to collect data in a list and perform a bulk write operation for better efficiency. |
| Lack of Error Handling | The code lacks proper handling of exceptions, leading to less meaningful error messages. | Implement appropriate error handling by raising specific exceptions or using `try-except` blocks and log exceptions for improved error messages and handling. |
| Missing Exception Handling in API Endpoints | The API endpoints catch exceptions but only print them and return a generic error response. | Handle exceptions explicitly, log them, and return appropriate HTTP responses with error details to provide better information to users or clients. |
| Deprecated Dependencies | The code imports outdated libraries (`newspaper3k` and `nltk`) that have been replaced. | Update the code to use current and maintained libraries for web scraping and natural language processing tasks. |
| Missing Unit Tests | The code lacks unit tests, making it harder to ensure correctness and reliability. | Write unit tests to validate the code's functionality and prevent bugs or regressions, especially for critical parts of the code. |

To address these challenges, we will incorporate the Asynchronous Programming Paradigm (Mozilla, 2023), Bulk Write Pattern, Error Handling Best Practices (Elise, 2020), Exception Handling Patterns (Microsoft, 2023), Dependency Management, and Test-Driven Development (IBM, 2023) in order to improve the performance, efficiency, error handling, maintainability, and reliability of your MAS.

## 6. Justification and Benefits
The design of our solution identified requirements and provide several benefits:
1. Improved Efficiency: Multiple autonomous agents with distinct responsibilities allow for workload distribution and task parallelization. Benefits: This enhances efficiency in academic

research by enabling concurrent and independent operations, reducing the time needed for data retrieval, processing, and storage.
2. Enhanced Collaboration: Agents in the MAS interact seamlessly, with the Data Retrieval and Extraction Agent supplying data to the Data Processing Agent, and the Data Processing Agent collaborating with the Data Storage Agent for storage and structuring. Benefits: This collaborative approach streamlines workflow and facilitates the achievement of research goals.
3. Modularity and Scalability: The MAS design allows for independent development, modification, and replacement of agents without affecting the system. Benefits: Easy maintenance, future enhancements, and scalability to handle increasing data volumes and evolving research requirements.
4. Intelligent Data Processing: The Data Processing Agent employs advanced algorithms to enhance data quality and usefulness. Benefits: Improved research accuracy and informed decision-making based on reliable data.
5. Structured Data Storage and Access: The Data Storage Agent stores processed information in a structured format and provides API access. Benefits: Organized data for easy retrieval, seamless integration with other applications or systems.

## 7. Conclusion

In summary, our MAS solution ensures a robust and effective system for automating the academic research process, ultimately leading to accelerated progress, accurate results, and increased productivity.

## 8. Appendices

Appendix 1.

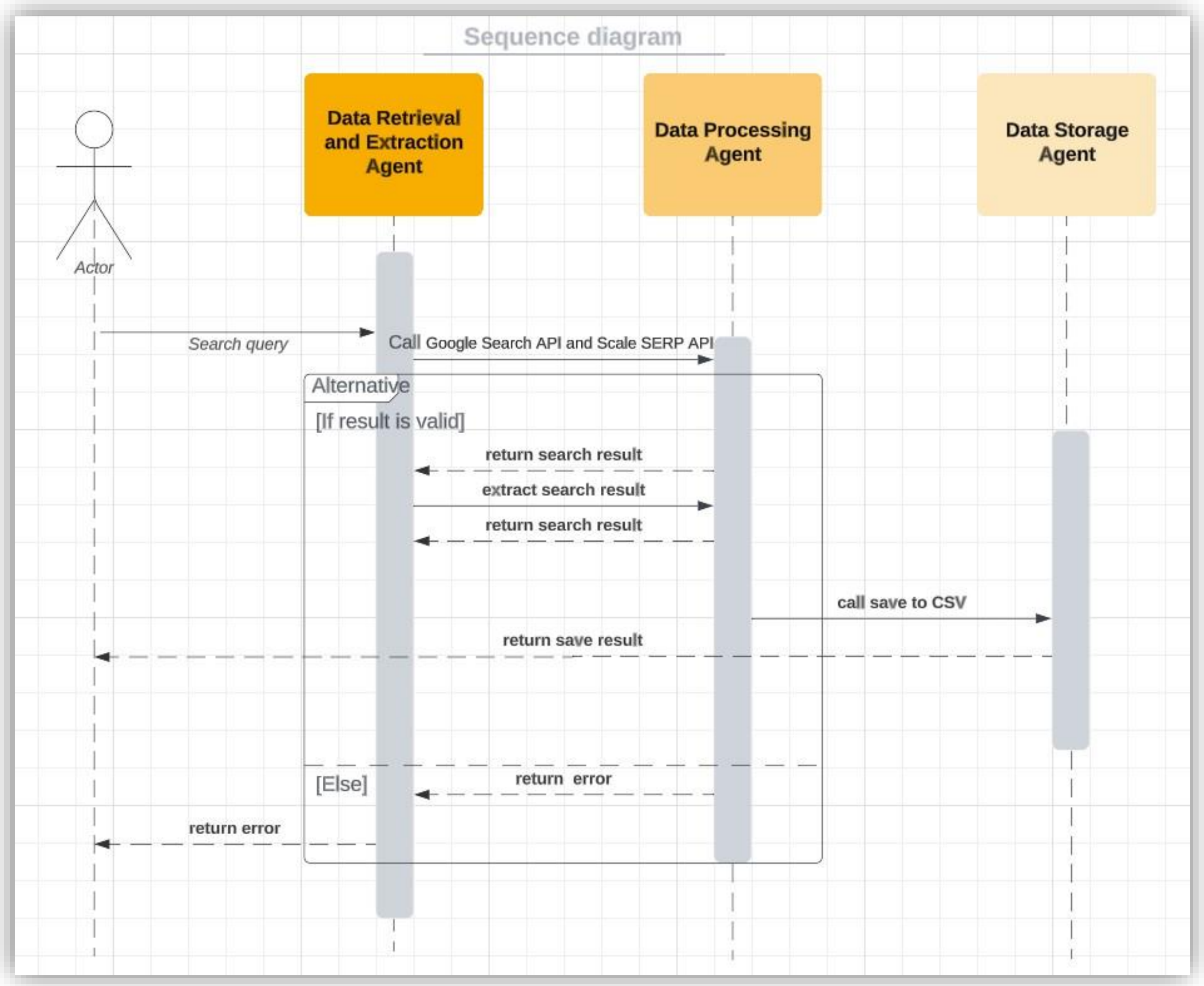


*Figure 1 Sequence diagram*

Appendix 2.

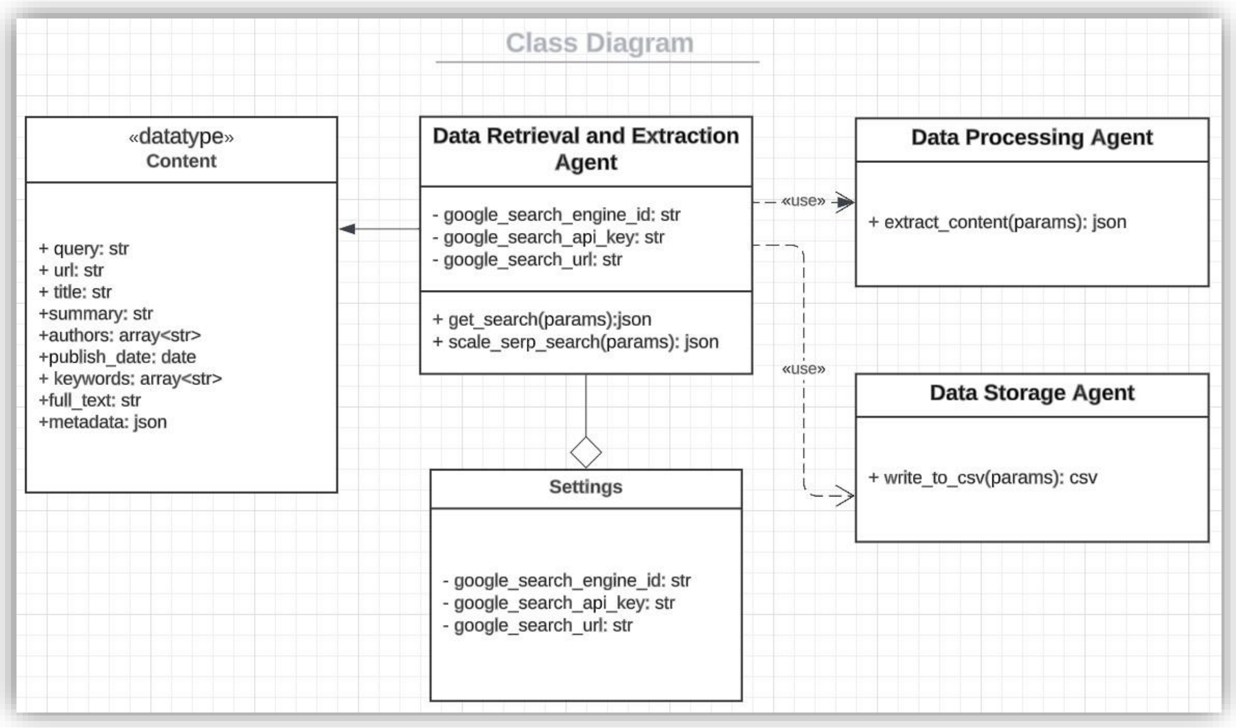


*Figure 2 Class diagram*

Appendix 3.

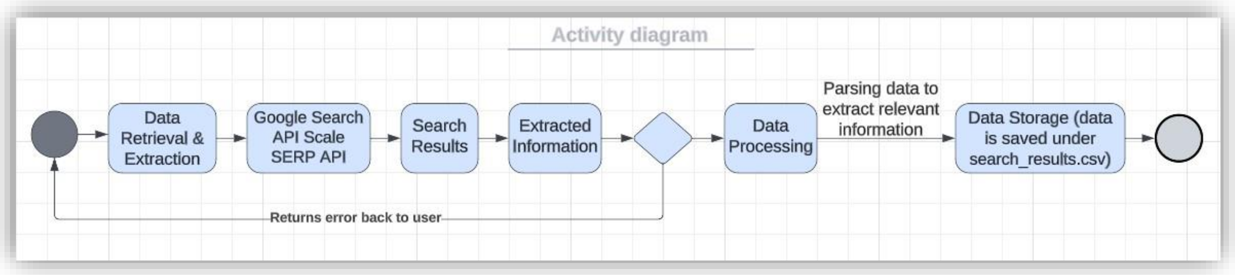

Activity diagram
Data Retrieval & Extraction
Google Search API Scale SERP API
Search Results
Extracted Information
Data Processing
Parsing data to extract relevant information
Data Storage (data is saved under search_results.csv)
Returns error back to user

*Figure 3 Activity diagram*

## 9. References

Bansall S. (2023) Agents in Artificial Intelligence. *Available from:* https://www.geeksforgeeks.org/agents-artificial-intelligence [Accessed 08 June 2023].

Elise J. (2020) Handling Errors in Python. *Available from:* https://betterprogramming.pub/handling-errors-in-python-9f1b32952423 [Accessed 08 June 2023].

IBM (2023) Test-driven development. *Available from:* https://www.ibm.com/garage/method/practices/code/practice_test_driven_development/ [Accessed 08 June 2023].

Microsoft (2023) Best practices for exceptions. *Available from:* https://learn.microsoft.com/en-us/dotnet/standard/exceptions/best-practices-for-exceptions [Accessed 08 June 2023].

Mozilla (2023) Introducing asynchronous JavaScript. *Available from:* https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing [Accessed 08 June 2023].

Russell, S. & Norvig, P. (2021) *Artificial Intelligence: A Modern Approach.* (4th ed). Pearson Education.

Shah, U.S. et al.(2022) Agent-Based Data Extraction in Bioinformatics. *Hindawi. Available from:* *https://www.researchgate.net/publication/359500106_Agent-Based_Data_Extraction_in_Bioinformatics* [Accessed 08 June 2023].

Wooldridge, M. J. (2009) *An introduction to multiagent systems.* (2nd ed). New York: John Wiley & Sons.