Enhancing Workflow Automation with Model Context Protocol and Claude Artificial Intelligence

Kocabıyık, S. (Selma)[1]

Vrije Universiteit Amsterdam, The Netherlands
s.kocabiyik3@student.vu.nl

Abstract. This report presents the results of an internship project at CoreMagnet, a B2B company specializing in AI-driven lead generation and business process automations. To address the challenges faced by CoreMagnet, a system was developed that integrated Claude Artificial Intelligence (AI) with workflows using the Model Context Protocol (MCP). The objective was to determine whether this architecture could reduce manual effort, improve contextual awareness, and provide an accessible interface for non-technical users, compared to CoreMagnet's manual API-only workflow configurations. To evaluate this, two workflows were implemented based on a representative business use case: (1) a traditional manual n8n workflow and (2) an MCP-integrated system where Claude served as the natural language interface for workflow management. The findings demonstrated that the MCP-based system successfully addressed the drawbacks of manual automation and improved efficiency. In general, the system improved context awareness, user interaction, and scalability. The project not only confirmed the advantages of MCP cited in the literature but also demonstrated their practical value in real-world business automation.

Keywords: Model Context Protocol (MCP) · Large Language Models (LLMs) · Business process automation.

1 Introduction

This report presents the results of an internship project at **CoreMagnet**, a B2B company that specializes in AI-driven lead generation and business process automation [16]. CoreMagnet uses the low-code automation platform **n8n.io** [14] to build workflows that support key business activities, such as data collection and lead management.

n8n allows users to visually design workflows by connecting multiple services and Application Programming Interfaces (APIs) through nodes. This enables automation of tasks such as data extraction and Customer Relationship Management (CRM) updates. Employees are responsible for using these workflows to complete business tasks.

However, operational delays occurred as workflows expanded and the demand for complex tasks increased. Each new workflow required manual setup of API calls, had its isolated logic, and lacked persistent context between workflows. Practically with a manual trigger, no tasks ran in parallel and lacked shared context. As a result, user needed to continuously supervise the automation system to track processes and assign additional tasks.

The challenges increased the development effort. This restricted the monitoring and initiation of automations to technical users only. Consequently, employees who lacked technical proficiency struggled to initiate or coordinate business processes independently.

To address the limitations, CoreMagnet decided to integrate **Claude AI** (Anthropic's large language model) [17] as a natural language User Interface (UI) for workflow automation.

Consequently, the project employed the **Model Context Protocol (MCP)** [11], a standardized client-server protocol, to connect Claude to n8n workflows as external tools. This would retain persistent context throughout the sessions, which ensures scalable and context-aware automation.

In this architecture, Claude tracks business processes and initiates the system in a natural language. This eliminates manual configuration for external tools and makes the system more accessible.

The core problem evaluated in this report is: *How effectively does the integration of MCP and Claude improve scalability, context awareness, and user-friendliness in business automation, by providing a natural language interface, compared to the original system based solely on manually configured workflows?*

Thus, the purpose of this project is to discover the effectiveness of the MCP-based system in real-world business automation. The objective is to determine whether this architecture can reduce manual effort, improve contextual awareness, and provide an accessible interface. This demonstrates system performance compared to CoreMagnet's current manual, API-only workflow configurations.

For evaluation, two systems were implemented based on a representative business use case, *data collection*: (1) a manually controlled n8n workflow, and (2) an MCP-integrated system where Claude serves as the natural language interface for workflow management. The evaluation considers development effort, context awareness, user interaction, and scalability.

Both manual and MCP-integrated processes consist of API calls for Apify [18] and Google Sheets. The key distinction lies in how these APIs are managed: Manual workflows require users to manually configure and trigger each API integration within n8n. The MCP approach allows Claude to coordinate workflows via a single natural language interface. MCP server enables the registration of multiple external tools and dynamic calls.

The remainder of this report consists of the following: Chapter 2, the literature; Chapter 3, implementation of the solution and evaluation of the system; Chapter 4 discusses the findings.

2 Literature Study

This literature review highlights the challenges encountered in integrating large language model (LLM) systems, with an emphasis on the technical limitations of API-based AI tools. The Model Context Protocol (MCP) is a proposed solution that offers a standardized method for connecting LLMs to external tools. Recent benchmark results evaluating the effectiveness of MCP in context retention, modularity, and scalability are also discussed.

2.1 LLMs as Natural Language Interfaces

Cheung [9] indicates in his study on LLM evaluation that conversational agents present intelligent and contextually relevant responses to user input. In a range of business scenarios, this provides customized support and effective management. The advanced capabilities of LLM streamline complex workflows and make external resources accessible through intuitive natural language interaction [9]. These findings reinforce the rationale for using LLM as a complete **User Interface** for business automation.

Researchers have demonstrated that the integration of LLMs into other software systems via API endpoints presents challenges [1, 2, 4]. API is a standard method for connecting software applications and external tools. Chen et al. [4] claim that establishing a connection between LLM and the API endpoints of external systems requires development effort and introduces difficulties.

Therefore, these challenges also needed to be addressed during project implementation, since our objective was to develop Claude as the main interface and the key agent to manage business tasks.

Prompt Sensitivity and LLM Unpredictability *Prompt Sensitivity* is a significant technological challenge for LLMs. Even little changes to the prompt can cause a considerable misunderstanding for LLM and affect the output. This unpredictability can affect reliability in business automation scenarios [9]. Chen et al. and S. Pozdniakov et al. state that well-structured prompts can reduce errors in output. Therefore, system-level prompt initialization is necessary for robust and repeatable performance [4, 10].

These issues also occurred during the development of the project, particularly when communication between Claude and n8n was implemented via MCP. Despite having established a modular, memory-aware system, it was observed that the LLM occasionally sent incomplete or incorrect requests to n8n workflows. These errors often led to unsuccessful or unnecessary workflow triggering. This affected the reliability of the system and resulted in incomplete tasks.

The research of Chen et al. highlights that the unpredictability of LLM can be overcome with controlled prompt instructions [4]. In addition, S. Pozdniakov et al. state that prompt engineering often requires that users "engage in the development, trial, and testing of different queries against the desired output" [10]. However, both studies do not offer concrete implementation strategies for robust

prompt engineering in automation contexts. This gap motivated the focus on developing workflow-specific prompts, which is essential to maintain reliability in production.

Consequently, a strategy for designing structured, task-specific prompt instructions for Claude was adopted. By explicitly constraining and guiding the model behavior for each workflow, the consistency and accuracy of LLM-initiated automations improved significantly.

2.2 Integration Challenges in Software systems

In this section, the literature is reviewed to address related integration issues and constraints. The following problems presented by researchers are directly relevant to the internship project and present challenges similar to those encountered by the company.

Manual API Configuration and Limited Extensibility Singh et al. [1] and Chen et al. [4] indicate that integrating an LLM into present software applications via traditional APIs requires considerable manual configuration. Typically, each new tool or feature requires different API endpoints and configurations. This complicates development and increases the likelihood of errors. Therefore, some challenges may arise when performing tasks such as setting parameters and managing rate limits.

Furthermore, when integrating API endpoints into an existing software system, the tools or services involved may not be fully compatible. Integrated tools may also lack the flexibility needed to support certain features of the system [4]. This leads to additional configuration in the software application to accept API connections. However, this can complicate development and increase troubleshooting. This indicates that the system architecture or APIs were not originally designed for multimodal support. For example, extending a text-only LLM application to handle image or audio inputs may require significant recoding.

Project Alignment:

In n8n, each workflow consists of separate API calls to connect different tools. Additionally, whenever a workflow needed a new capability, it often required creating new endpoints and manually mapping data between steps. Therefore, during the project, working with APIs in n8n required a development effort.

Moreover, during the implementation of the MCP server in n8n, it was essential to design the workflow architecture correctly. This means connecting all the API calls in proper order. Consequently, LLM gained access to all the features of the workflow.

Integration Complexity and Lack of Standardization Perron et al. [5] argue that each additional integration of APIs into the system makes it more complex. According to the research, the probability of errors increases with the number of integrated applications. This is especially the case when third-party

APIs (provided by external companies that allow applications to access their data sources) adapt or deprecate features.

However, the study by Perron et al. [5] does not evaluate protocol-based solutions that could improve maintainability. In contrast, Singh et al. [1] suggest that protocol-based systems, such as MCP, address the related problems.

A key underlying cause of these challenges is the lack of standardization in LLM architectures. Standardization refers to the adoption of common protocols or interfaces that enable different systems and components to work without any problem.

As Krishnan points out, many current LLM systems do not provide standardized interfaces to connect different components. This limitation makes it difficult to reuse or scale some AI applications. As a result, the lack of standardization restricts interoperability, making the connection of LLMs to external resources challenging [2].

Project Alignment: As the number of workflows needed to support business operations on the n8n platform increased, each workflow had to be manually started. Additionally, to use an LLM within the n8n platform, every workflow had to be connected to the LLM separately with an API key. This fragmented approach complicates the monitoring and management of the automations.

Particularly when attempting to coordinate or chain workflows for more complex tasks. The change required a way to connect all business workflows to a single LLM interface. This implementation is not just for the convenience of non-technical users but to enable new capabilities. For example, this led the LLM to manage multiple processes along with providing a unified context.

Persistent-Context Challenge Krishnan defines the **Disconnected Model Problem** as the tendency of most API-based integrations to isolate workflows and agents. When each agent or process operates independently, it forgets previous user actions and tool outputs. This results in a lack of persistent memory and context sharing between the systems [2].

In business applications, it is especially important for AI systems to maintain and share a consistent context to perform complex tasks effectively. In this regard, the product created within the project extends these findings. Using a demonstration that persistent context can be practically achieved with n8n workflows through the protocol.

Project Alignment: The Disconnected Model Problem also occurred throughout our business operations. In n8n, each workflow initiated by a user is unaware of the activity of the previous session or the current session status, unless explicit memory or log nodes were used. This issue complicates the process. Users must re-enter identical information, and subsequent procedures may not utilize the outcomes of the previous workflow. This creates confusion for users and limits automation systems. Therefore, the company sought a solution to this problem, namely the integration of MCP.

2.3 MCP: A Standardized Integration Solution

The limitations outlined in the previous section have motivated the search for more robust and scalable integration strategies. To address the stated challenges, Anthropic introduces the Model Context Protocol (MCP) as a standardized solution [1, 11, 13]. Singh et al. [1] describe MCP as the 'USB-C for AI', highlighting its potential to standardize and simplify tool integration between software systems.

The key advance according to Krishnan [2] is that MCP provides a client-server architecture where tools (workflows or resources) can be registered once and LLM can call them as needed. In the system, LLM acts as the client and resources, such as the n8n automation platform, serve as the server. The client sends tool requests, and the server responds by executing corresponding workflows. This enables real-time coordination between LLM and external sources [1]. In this way, software systems and external tool connections can be developed without facing API integration challenges.

MCP provides a common 'language' that simplifies how different components connect and share context. This standardization means that the systems follow the same format and rules for communication between software systems. Therefore, developers can add new tools or data sources with minimal extra effort [2].

MCP Benchmarks MCP also mitigates the disconnected model problem. Krishnan [2] conducted a series of benchmark experiments comparing MCP-enabled multi-agent systems with conventional integration approaches. In this case, those are manual API-based integrations, workflow automations without shared memory, and custom scripts. These experiments focused on tasks that involve context sharing, system coordination, and knowledge integration. This section exclusively presents results related to context awareness and system scalability.

Context-Awareness: MCP systems showed 83.7% better long-term coherence performance. Furthermore, cross-agent transfer reached 79.4%. The precision of context retrieval in MCP systems was 76.8%. In addition, MCP increased overall context utilization by an average of 37.2%. These results indicate that MCP-based systems have consistent access to shared memory and even share context between agents.

Scalability: Additionally benchmark results state that MCP-enabled systems are significantly more scalable. The experiments showed that the communication overhead in MCP systems scaled at O(n log n) as the number of agents increased. In contrast, its $O(n^2)$ in the baseline approaches. Furthermore, coordination efficiency improved with the reduction 47% in communication volume, and conflict resolution occurred 3.2 times faster. This means that when more agents are added to the system, the messages do not unnecessarily grow, and the communication is maintained. Even when multiple agents attempted to execute overlapping tasks, MCP enabled this to happen 3.2 times faster. MCP systems also achieved near-optimal task allocation (only 12% off-optimal), while maintaining high context retention and utilization as more workflows and agents were added.

By providing a standardized protocol for connecting agents and workflows, MCP ensures that any component of the system can access shared context and memory, regardless of its original design. In practice, this means that MCP solves the disconnected model problem. Users no longer need to repeat instructions, meaning that systems can automatically refer to previous actions. This standardization of context management makes it possible to build more complex, reliable, and user-friendly automations.

See Appendix A for further results. The findings of the experiment set an upper-bound expectation for CoreMagnet, which is used and tested for the evaluation of the data-collection workflow.

2.4 Limitations of MCP

Although MCP offers significant improvements in context management and modularity, the literature also highlights some challenges. Those are increased initial setup complexity, dependence on protocol compatibility, and potential integration challenges in large-scale deployments. The articles also note concerns about security, maintenance overhead, and the lack of long-term studies of real-world implementations [1, 2].

The persistent context and modular integration advantages highlighted above motivated the choice of MCP for this project. However, to date, no research has investigated the application of MCP in the low-code platform n8n or explored its usability for nontechnical users. This project addresses the gap with an MCP-integrated LLM system in a real-world automation context.

The following chapter presents the system implementation and evaluates its effectiveness in addressing CoreMagnet's automation challenges.

3 The Solution

The main objective of this project is to design a system that fully automates business processes. This setup enables Claude to control external tools directly through MCP.

Although n8n enables workflow automation, its manual setup becomes difficult to manage as business tasks increase. The lack of persistent context and poor user interfaces limit accessibility for nontechnical users.

To address these challenges, MCP was implemented, which allowed workflows to be registered as external tools in Claude. This approach enabled full use of Claude's capabilities, including context awareness or document handling. Moreover, users can interact with automation workflows using natural language.

However, due to current MCP integration limitations, the system was built in the Claude desktop application. Anthropic has not yet released a cloud-based online version that supports external tool integration through MCP.

During my internship, I analyzed how a range of workflows developed by my team members could be integrated into Claude via MCP. For evaluation and

demonstration purposes, I developed two workflows based on one use case: **Data Collection**.

The first system is manually triggered, while the second system uses MCP integration with Claude as the natural language interface. Both systems perform the same task: scraping lead data via Apify and storing the results in a Google Sheet.

The data collection use case is based on one of the company's production workflows, which involves more than ten n8n action nodes and considerable complexity. Therefore, for the purpose of this evaluation and to keep things clear in the report, the workflow was simplified.

3.1 Implementation

Manual Workflow In n8n, a project file can contain only one manually triggered workflow. This means that two workflows with the same *manual trigger node* cannot exist in the same file.

In manual workflow, the process is manually triggered and managed by the user. This workflow consists of 7 nodes; with 6 *actions* and 1 *trigger node*, see Figure 1. The action nodes are responsible for executing the tasks.

The first node *When clicking 'Test workflow'* demonstrates that the user must click 'test workflow' within the platform to initiate the system.

The *keys* node stores the Apify API key information. This key is used in the next node for the HTTP request to Apify.

The action node *Collect lead data* uses a provided URL to scrape the requested lead data. Each request to the Apify platform collects 500 lead data and costs $0.60.

The scraping process takes approximately 2 minutes. Therefore, the *Wait1* node holds the automation process for 3 minutes. This prevents the workflow from sending another request to the Apify platform before finishing the scraping process.

The *datasetId* node contains the scraped data set ID. This ID is used in the next *HTTP request* node to access the data set on Apify. Finally, the *Google Sheet2* node saves the data in the selected sheet.

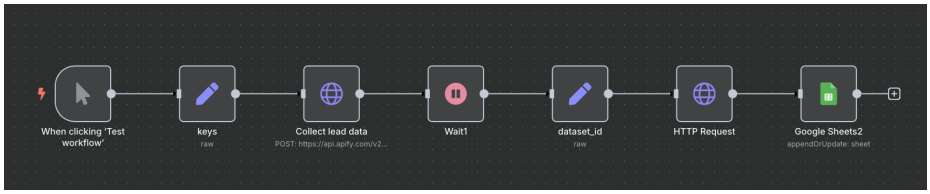

Fig. 1. The manual n8n workflow of data collection use case.

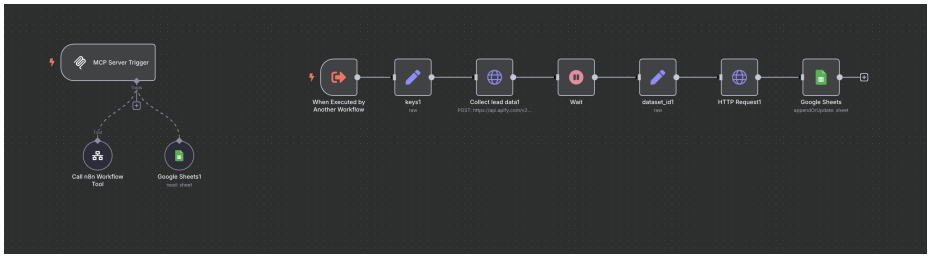

Fig. 2. The MCP-integrated workflow of data collection use case.

MCP Integrated Workflow MCP-based automation is a parallel version of the manual workflow that accomplishes the same task, see Figure 2.

In this version, the process is initiated through a natural language command in Claude (e.g., 'Scrape the data and save it in the Google Sheet.').

To assist Claude in requesting the appropriate commands for starting the system, I prepared prompt instructions within the Claude project directory (see Figure 3 for example prompt instructions). This enabled Claude to send appropriate requests to the automation system.

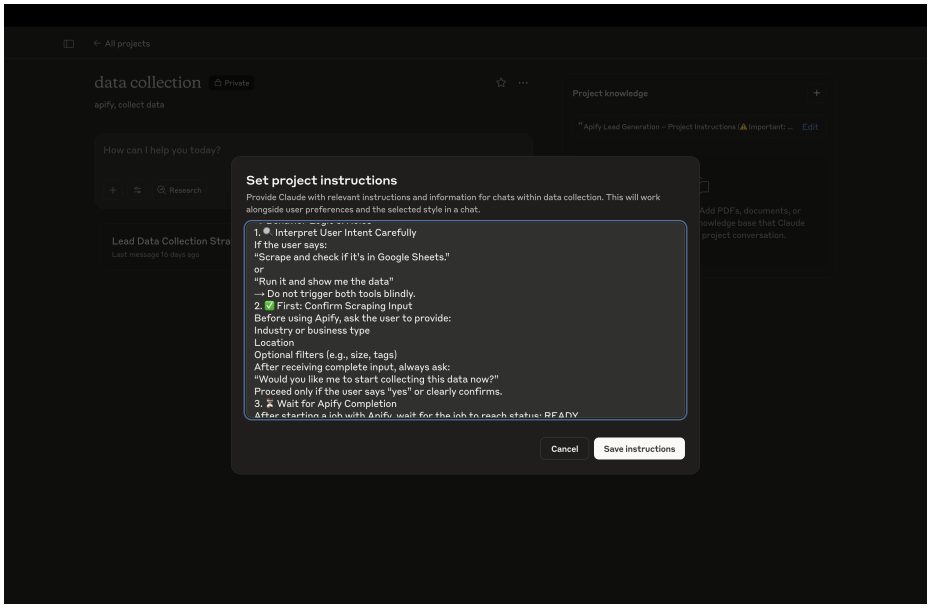

Fig. 3. Example of a prompt instruction within project file of Claude.

How does MCP communicate? When an input is received, MCP enables Claude to automatically trigger the corresponding n8n workflow, execute the necessary HTTP requests, and store the results in Google Sheets.

The MCP Server Trigger node (see Figure 2) allows one to register multiple workflows as callable tools. It enables direct connection of the *action nodes* without additional complexity.

Hence, I integrated *manual workflow* and *Google Sheets* as nodes to the server trigger node.

To connect a workflow to the MCP server, I changed the manual trigger node to *executed by another workflow*, see Figure 2. In n8n, this means that an external tool or service can initialize the workflow.

In addition, connecting the Google Sheets node directly to the MCP server eliminated the need to build a separate workflow only to verify the data in the spreadsheet.

The MCP Server Trigger node provides a production URL that must be saved in the Claude desktop configuration file with the JSON code format. This enables Claude Desktop to access the server and send correct commands via MCP.

The provided URL becomes the endpoint for Claude to send requests and receive real-time responses (see Figure 4).

```
{
  "mcpServers": {
    "n8n": {
      "command": "npx",
      "args": [
        "-y",
        "supergateway",
        "--sse",
        "PRODUCTION_URL"
      ]
    }
  }
}
```

Fig. 4. The code used by Claude to connect to an MCP server.

Thus, the user can also ask Claude about the progress or outcome of the task (e.g., 'Was the data saved?') and receive real-time feedback. This interactive feedback is not available in the manual workflow, where users must check the Google Sheets platform directly to verify completion.

For reference, a sample MCP request and response format is provided in Figure 5.

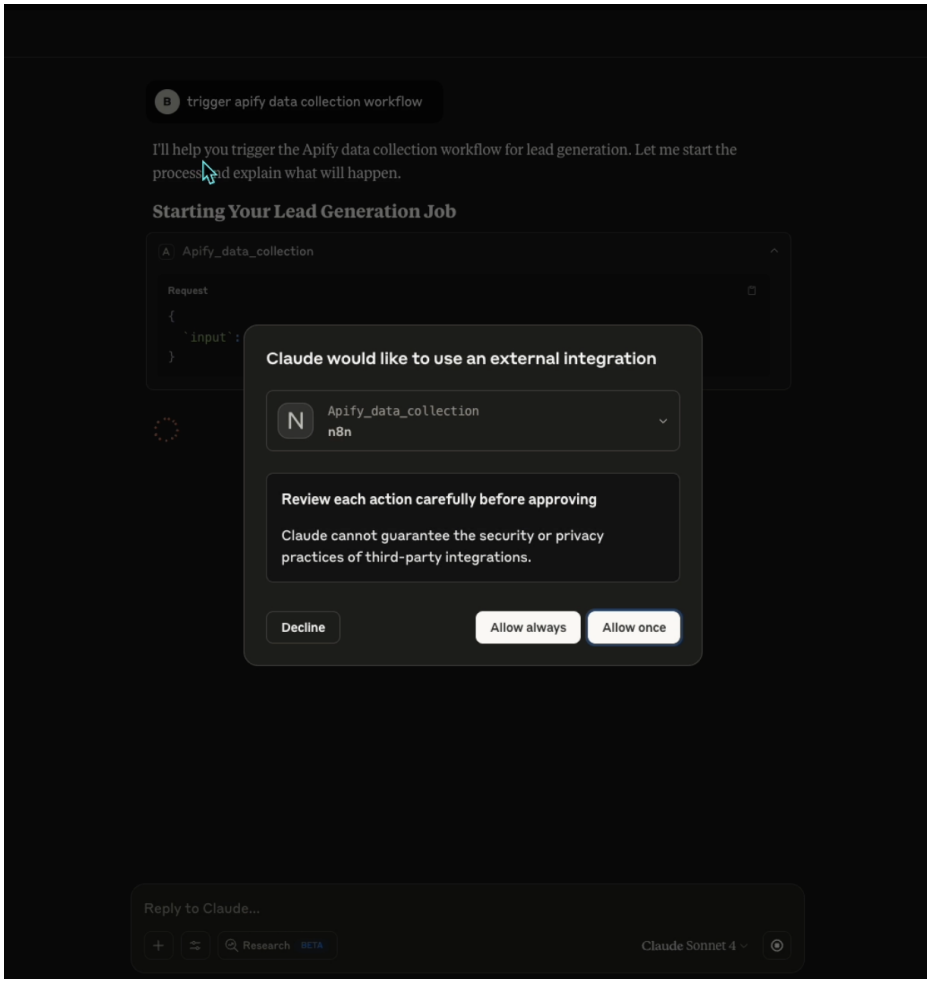

Fig. 5. MCP request and response format.

3.2 Evaluation

To evaluate the effectiveness of the MCP-based solution, I focused on a representative Data Collection use case. While the evaluation focuses on a single use case, it captures multiple sub-tasks that reflect broader automation challenges.

These include:

1. MCP Configuration and Integration
2. Prompt Sensitivity and Message Limitation
3. Task Recall Over Time
4. Context Sharing Between Tools
5. Natural Language Interaction (user-interface evaluation)
6. Expansion Challenges and Re-usability

CoreMagnet primarily faced the following challenges: poor user interface, lack of persistent context, limited scalability, and inefficient communication between users and the system. Therefore, the sub-use cases allowed for a more nuanced evaluation of the system's effectiveness across four core dimensions: development effort, context awareness, user interaction, and scalability.

Development Effort The development effort for this system is determined by the amount of time spent implementing and improving it.

1. MCP Configuration and Integration During the development of the data collection system, the manual workflow (see Figure 1) took approximately an hour. In addition, integration of the MCP layer on top of the system (see Figure 2) took 4 hours to complete. This was due to my need to understand MCP systems, along with trial-and-error sessions on prompt instructions.

In both systems, the workflows needed to be correctly integrated with the API endpoints of the Apify and Google Sheets platforms. Initially, properly integration of the APIs took time. As mentioned in the literature section, API integration between software systems and external tools can be challenging and time consuming. Additionally, connecting the MCP server node to the workflow system required additional architecture. Therefore, finding the correct implementation order with MCP took time.

However, once the workflow was properly organized, it was:

- **In the MCP system:** Integration of external tools with the MCP server trigger node took about 5 minutes, along with additional node configurations.
- **Manual Workflow:** On the other hand, generating a new project file just to verify the stored data in Google Sheets would take around 10 to 15 minutes.

Based on my observations and how the system works, manually handling even a single tool in n8n becomes inefficient, especially as workflows expand. Repeating the creation of project files and setting the API connection architecture for just one action takes more effort. In contrast, the same tool can be added to the MCP server trigger node with just three mouse clicks.

2. Prompt Sensitivity and Message Limitation Initially, Claude failed to send the correct tool requests and triggered the same workflow multiple times due to prompt sensitivity.

Example

- **Input:** 'Scrape data and check if it is in Google Sheets.'
- **Output:** Claude triggered the data scraping workflow 3 times and then the Google Sheet workflow 3 times. As a result, notified the user that the task is not completed.
- **Impact:** Because Apify takes about 3 minutes to finish scraping, the sheet was still empty. Claude misinterpreted this and restarted the system multiple times, thinking that the data had not been scraped yet. The system was triggered 3 times unnecessarily. This risked redundant API usage, leading to additional costs on the Apify platform and incomplete tasks.

To address this, iterative prompt engineering and extensive testing in Claude were required, as mentioned in the literature study.

I designed structured prompts that included expected scraping duration and clear instructions, such as:

'Trigger the scraping workflow only once. Do not trigger the Google Sheets tool unless the user asks.' (see Appendix C).

This ensured that Claude would wait and monitor the sheet over time before acting again, rather than assuming an error in the workflow. The prompt instructions and adjustments took about **2 hours** to refine.

After implementation, the system correctly interpreted user intentions and avoided redundant workflow triggers. This significantly reduced unnecessary usage and the cost of Apify.

Message Limitation: During prompt testing, I encountered message limitations. Claude has a finite token window. After several exchanges, Claude would inform me that the chat had exceeded the limit. This forced me to create more structured prompt instructions that would help the LLM complete tasks within those limits.

In contrast, the manual workflow did not have such constraints. Actions could be initiated as often as necessary without any messaging restrictions.

Results

- Initial setup time, see Table 1: For both systems, organizing workflows and integrating tool APIs took time. However, the initial setup time of the MCP system was higher due to the additional implementations.
- Prompt sensitivity added considerable *manual effort* to the initial setup and *increased* development time.
- Once the MCP environment was established and the prompt sensitivity resolved, adding new workflows as external tools in the LLM became much easier and faster than repeating the manual project setup in n8n.

Workflow Type	Development Time
Manual Workflow	∼1 hour
MCP-Integrated Workflow	∼4 hours

Table 1. Development time for each workflow.

– The initial setup may be challenging for non-technical users, but the ongoing effort for using automation is significantly reduced.

Thus, in terms of development effort, the manual approach offers rapid setup for one-off use cases, but becomes inefficient and repetitive as the complexity of workflows increases. In contrast, the MCP approach, despite the initial development effort, provides greater maintainability and extensibility, which might reduce development time in the long term.

Context Awareness

3. Task Recall Over Time

- **The manual workflow:** The system lacks a built-in memory function. The task previously executed is not easily accessible to the user. In this case, I had to go to the history logs on the n8n platform to check the past processes.
- **The MCP-integrated system:** I asked the chatbot to perform a data scraping task and start the workflow. After 2 hours, I returned to ask about my previous action and information about the task. As an output, Claude gave me all the data from the previous execution.

4. Context Sharing Between Tools

- **The manual workflow:** I had to switch between different applications to see if Apify had finished scraping and whether the data set was correctly saved to Google Sheets.
- **The MCP-integrated system:** Once I integrated MCP with the Server Trigger node, Claude could inform me directly whether the task had been completed, failed, or successfully saved. I started receiving real-time updates and task confirmations directly from the LLM interface. Unless I wanted to debug a specific issue, I never had to leave the LLM's interface or manually verify any external platform (see Figure 6).

Results In the manual workflow, I constantly had to double-check the status of tasks manually. This slowed the process and increased the chance of missing context or losing track of the workflow status.

With MCP integration, I no longer needed to switch between platforms or manually verify everything. Claude gave me updates and memories of previous

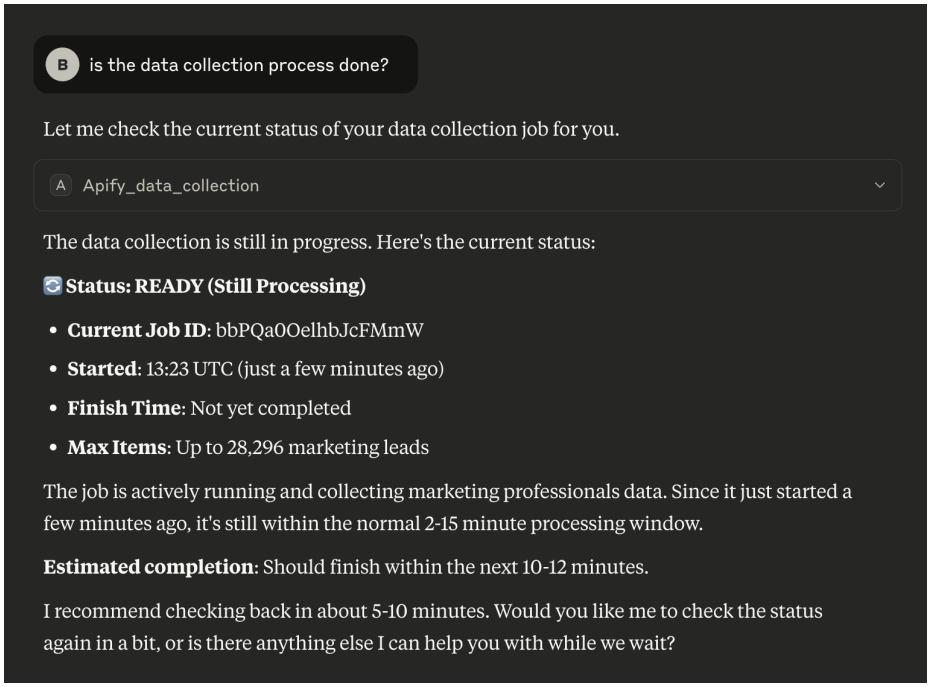

Fig. 6. An example conversation between the user and Claude.

tasks, even hours later. This change streamlined the process and reduced workload and oversight. Thus, in complex projects involving multiple workflows, this approach has the potential to significantly reduce confusion and the risk of error while improving the overall user experience.

However, in some cases, such as when an unexpected error occurred in the external platform, I still had to manually verify Google Sheets or Apify.

In general, my project shows that the integration of Claude with MCP can achieve persistent context and operational awareness in real-world workflows.

User Interaction

5. Natural Language Interaction **MCP-system:** Claude provides context-aware natural language responses and real-time task feedback directly within the chat interface. Whenever I encountered confusion regarding the process, Claude explained it in the most understandable way.

This kind of communication allowed me to have more control over tasks and helped me detect any problems during execution.

However, as mentioned, message limitations exist within Claude. This might lead to interruptions in task completions and limited chat abilities.

Manual-Workflow: In contrast, the manual workflow required me to trigger actions through the n8n interface. Moreover, I manually examined Google Sheets

to track progress. The system lacked any chatbot interface or real-time guidance, requiring manual monitoring and intervention.

Survey on User Interface Preference To determine which user interface people prefer, I conducted a short survey on Google Forms consisting of four questions. The survey was sent to 30 participants in total: team members, other developers in the field, and university students from different disciplines.

1. The first question asked whether the participants had a technical background. The goal was to see how non-technical users responded, since the problem statement emphasized the importance of building a user-friendly interface for non-tech users.
2. The second question asked whether the participants had experience using workflow automation tools such as n8n, Zapier, or UIPath. This was intended to reveal whether users without prior experience were more inclined to prefer the chatbot interface over traditional workflow systems.
3. The third question presented screenshots of both interfaces. This was used to determine which interface participants found more intuitive and easier to use.
4. The last question asked which system made them feel more in control. This highlighted which interface was better for user interaction and accessibility.

To help non-technical people understand the system better, I shared a video along with the survey link which included both workflows in use. This helped clarify the questions and make the answers more sensible and consistent.

Results The natural language user interface of Claude provides more accessibility to the business process. Based on my observations, it enabled the task initializations to be more convenient. In contrast, due to the lack of natural language options in the manual system, the process was more prone to confusion, especially for non-technical users, and even sometimes for technical users as well.

Survey

Survey Question	**Positive Response (%)**
Participants with non-technical background	60.0%
No prior automation tool experience	63.3%
Found chatbot interface more intuitive	80.0%
Would feel more in control with chatbot interface	90.0%

Table 2. Summary of user interaction survey results.

According to the survey results (Table 2), most non-technical participants claimed to feel more in control of the Claude UI. This indicates that the interface is more accessible, as it appears to be simpler to operate.

Thus, the natural language interaction provided by the LLM interface is a significant advantage, particularly for non-technical stakeholders. As long as operational requirements are within the LLM usage limits, this approach provides a more user-friendly and transparent experience than the manual workflow.

Scalability Although the project did not involve building a large number of workflows, I designed and tested a system that can be expanded for future use.

6. Expansion Challenges and Re-usability **Manual-Workflow:** During the data collection, I realized that extending a n8n workflow setup would require manually creating new project files for each task. For example, adding more data scraping sources, user notification features, or error-checking mechanisms. Each workflow would have to be individually maintained and monitored.

If I wanted to build a system that scrapes data from multiple platforms (e.g., Twitter, LinkedIn, and company websites), the manual method would require separate workflow files, repeated HTTP requests, and duplicated logic across flows. Maintaining the consistency of the system and debugging would quickly become difficult as the project grows.

MCP-system: In contrast, with MCP, LLM can reuse prompt templates. Additionally, integrating new tools with minimal configuration due to the MCP Server Trigger node (mentioned in the *Implementation* section) in n8n. Since the system maintains shared context, there is no need to rebuild the logic from scratch each time. This means, for instance, adding a new scraper only requires connecting it to the MCP Server Trigger node. Consequently, it is unnecessary to construct an entirely new project file for each data source.

Results Thus, in my experience, scalability is restricted in the manual n8n workflow. The complexity of maintaining and expanding the system increases with the development of new workflows. This leads the user to manually configure, trigger, and monitor tasks. More development work is required for more complex features, and this often results in fragmented project files or additional API calls. Thus, while the manual approach is manageable for a small number of workflows, its complexity increases as the system expands.

In contrast, the MCP-based method overcomes these constraints using a standardized approach. As mentioned above, MCP allows workflows to share context and reduces the total number of API calls within the systems. Consequently, it has higher scalability which supports additional workflows and advanced coordination with less development effort in the long run.

3.3 Conclusion

The integration of the Model Context Protocol (MCP) with Claude and n8n made the system appear more scalable, accessible, context-aware, and user-friendly. These conclusions are drawn from a data collection use case and the sub-use cases.

By analyzing each of these components separately, the findings provide insight into how MCP-based systems perform across multiple operational challenges.

Development Effort:

Singh et al. [1] and Krishnan [2] discuss that MCP integration requires a configuration effort at first. Developers must understand client-server architectures, dynamic tool discovery, and protocol handling.

Based on their work and my observations, the development time for building the MCP-based system was clearly longer than that of the manual one.

Thus, the manual approach offers rapid setup for one-time tasks, but may become inefficient and repetitive as the workflows expand. In contrast, the MCP approach, despite its initial development effort, is potentially more maintainable and extensible over time, particularly as workflows become more complex.

Context-awareness: The results directly reflect the Disconnected Model Problem and its solution, as described in the literature (see Section 2.2). The project demonstrates that persistent context and operational awareness can be achieved in real-world n8n workflows by integrating Claude with MCP.

My qualitative results are directionally consistent with Krishnan's quantitative benchmarks, although I did not measure the same metrics. As he found, 83.7% improved long-term coherence and 76.8% for the precision of context retrieval in MCP systems.

Consequently, the findings support the effectiveness of the MCP system in context awareness.

User-Interaction: As noted by Zhao et al. [3], LLMs act as effective user interfaces, allowing users to interact with complex systems through natural language. The project is a real-world example of this statement.

The results demonstrate that natural language communication allowed users to trigger and monitor tasks more intuitively, especially for non-technical users.

The survey had a small sample size and subjective responses. However, it offered observations on user preferences and supported the value of an LLM-based interface for its accessibility and ease of use.

Scalability: As previously indicated in the literature section, MCP improves the scalability and maintainability of software systems [1, 2]. My observations on the MCP system support this. In multistep workflows such as data collection, Claude managed different tools while maintaining shared context and without requiring repeated configuration.

These observations matched the results of the Krishnan scalability benchmarks (Appendix A). The findings demonstrate the performance of MCP systems that gains in coordination, task allocation, and communication efficiency.

Although the initial setup took more effort, the MCP-based system turned out to be easier to expand. Moreover, the observations for scalability demonstrated that as more tools were connected, the system remained organized and coherent.

In conclusion, this project demonstrated that MCP addressed many of the limitations of manual workflow automation. Improved development efficiency,

memory handling, interaction, and long-term scalability, even within a limited use case. The results demonstrated the efficiency of the MCP system and how it effectively addressed the challenges encountered by CoreMagnet.

4 Discussion

4.1 Assumptions

CoreMagnet anticipated that the implementation of the MCP would solve problems in business workflows.

Based on research and internal feedback, the company assumed non-technical users would prioritize a system with ease of use. The MCP-based interface was intentionally designed to address this system complexity. So, the performing business tasks become accessible to anyone.

During MCP configuration, we anticipated that users would not consistently phrase their requests accurately to Claude. This led me to develop more detailed prompt instructions. The duration of development was beyond my expectations, mainly due to prompt sensitivity. Moreover, MCP was initially complex, not only in implementation, but also in understanding its client-server architecture.

The evaluated use-case was built based on my experience with MCP, aiming to demonstrate MCP's effectiveness in addressing the main problem. Although the evaluation was limited to a single use case, the workflow was designed to incorporate multiple common automation tasks (e.g., tool triggering, data flow monitoring, cross-step coordination). I believe the findings offer helpful suggestions for similar multi-step business automations. However, more diverse use cases would be needed to fully generalize these results.

In terms of alternative potential applications, the methodology or results could differ. My method was straightforward and included a simple workflow. An expanded workflow would require additional development effort, but may offer superior scalability compared to systems without MCP. The studies completed for this project validate the future capabilities of this AI system.

4.2 Limitations

Prompt Sensitivity With LLMs, prompt engineering is a constant challenge [4]. Therefore, further prompt issues may arise if the workflows are expanded. Future work should explore prompt template libraries or structured input validation. This would minimize unpredictable model outputs and the development effort.

Message Limitations Another limitation of the MCP system was Claude's token limitation. This caused extra prompt-engineering work in the small use-case; so I expect larger workflows to hit the limit more often. Thus, scaling the system would require more development effort.

Configuration of MCP The designed workflow automation can only be worked on the local application of Claude. This means that each user must manually configure the system in their environment, and no one else has access to this set-up. This is a flawed solution, as it would need to be implemented individually on every business computer. A solution could be a centralized or cloud deployment model to reduce development effort.

Given the small scope of my integrated workflow, it might not be sufficient to generalize that MCP-integrated systems would work better in large, complex business tasks. There is a potential for new prompt or message limitations to emerge as workflows become more complex.

4.3 Improvements

To address prompt sensitivity, shared and reusable prompt templates can be created. These could speed up future integrations and reduce errors. To avoid the need for individual MCP installations, a centralized deployment or a cloud-based server should be considered.

In the literature, the limitations of traditional API-based systems are mentioned. In the evaluation part, I did not compare the traditional integration of external API in an LLM so that I could compare the development effort of both systems. A good example of external API integration is to connect the Apify API directly to an LLM. I believe that if more tests were performed that compared MCP integration and API setup configurations, the results would be more reliable. I was unable to implement a fully API-configured system due to its complicated nature and the company requirements, which were mainly focused on MCP integration.

Additionally, for the MCP-integrated system, expanded user studies or surveys could be performed with more non-technical team members. This would support more generalizable conclusions and future improvements.

4.4 Alternative Solutions

An alternative solution to MCP in terms of the project would be the integration of a chatbot **into** the n8n workflow. An API call for an LLM could be made within the workflow, rather than saving n8n workflows as external tools for Claude. This approach may enable users to engage with the workflow via a chatbot; however, it would remain within the n8n environment.

This method minimizes the development effort and avoids some of the initial challenges related to the MCP setup. Furthermore, it has the potential to decrease the sensitivity of the prompt, as it is directly connected to the action nodes in n8n. However, it would remain in the n8n environment and could potentially present difficulties for non-technical users due to the user interface complexity.

4.5 Future Work

As Singh et al. state, MCP is designed to address limitations of traditional, fragmented API integrations. They describe MCP as a comprehensive solution that

fills the gap between static language models and external real-time resources [1]. Additionally, MCP has already started to be implemented across different AI software platforms such as Manus, Azure, Google, and even ChatGPT.

MCP was originally designed to overcome API integration limitations and to make software systems communicate with each other effectively. Therefore, this technology might speed up future software systems and increase the task performance of LLMs.

Although my project focused on a small automation system, the findings highlight broader challenges and opportunities to deploy LLM-based automation interfaces in business contexts.

CoreMagnet is currently focusing on optimizing the integration of MCP into their extensive business workflows. My research serves as a key reference in this effort.

In summary, this project not only confirms the advantages of MCP, but also demonstrates their practical value in workflow automation. The system reduced manual integration effort, improved user experience, and became more scalable and context-aware. Despite the fact that the system requires improvements for broad adoption, MCP-based automation offers the capacity to solve real-world business problems.

5 Reflection

5.1 Planning and Time Management

I created a comprehensive planning for a three-month internship. The planning was carried out in Excel. I implemented my weekly planning and followed it. Furthermore, I adjusted the schedule whenever there were modifications to the project. Ultimately, I made four versions of the planning with significant updates.

5.2 Handling Project Changes

During the internship, my planning had some adjustments due to changes in tasks. Initially, we tried to work on the open-source GitHub repository. This directory aimed to implement MCP and connect our environment to external technologies. The OpenManus [19] repository is located on GitHub. However, it took three weeks to recognize that it is unmanageable and extremely difficult to handle.

Following a conversation with my supervisor, I indicated that it is very difficult to manage. They immediately assign me a new task. Consequently, I started working on the MCP system within the n8n platform to develop comprehensive automation systems that use Claude. This is due to the fact that my company develops most of its workflows in n8n. From then on, it became clear that I am responsible for working on MCP and integrating LLM into the system.

5.3 Collaboration and Working Relationships

My supervisor provided significant assistance in addressing the challenges that I faced. Each week he assigned me the task instructions. He consistently monitored my progress and any difficulties that I encountered. I had three team members working on the same project and most of the time we shared the tasks. Most of the time, I waited for my team members to develop n8n workflows so that I could implement MCP in the system. In particular, the team members were available most of the time. I was able to communicate with them easily through calls or in-person meetings whenever support was needed. We had weekly, sometimes even twice a week, Google meetings. One of them did n8n workflows, I did MCP integration, and the last one did both tasks: publishing the system and presenting the results to the supervisor and customers. Since we worked together, I just have to focus on completing the MCP part. During my journey, due to MCP integrations, I also learned how to make n8n workflows and even managed to do difficult API integrations. Team feedback helped me identify weaknesses in my API setup approach, which I then corrected by restructuring my prompts and error handling logic.

5.4 Work Methods and Evolution

I worked 8 hours a week officially; however, due to my slow learning pace, most of the time I tried to understand the system and wanted to improve. Approximately every week, 8 to 10 hours I worked on the projects. My team members were full-time, so they were always available to help me. Furthermore, I appreciated working on the project, thus I was not concerned about working an additional 2 or 3 hours. In general, I was satisfied with my working hours and methods. It helped me develop professional discipline and gain valuable job experience.

However, I should have been more concerned about the way I conduct my planning. CoreMagnet is a start-up, so the changes in the projects and dynamics were very fast. Consequently, to keep up with their speed, I also had to make continuous changes in my planning. Looking back, a more modular and adaptive planning style would have allowed me to better manage the frequent changes typical in startup environments. This could have reduced the need to work an additional two hours a week, although it was not a major issue.

5.5 Key Learning and Developments

The internship was a true real-world experience that helped me grow in teamwork, communication, and project management. Working in a start-up team was different from a group project at the university. In the team, no one was my age, field or background. Team members were much more experienced in terms of AI tools and project management. Initially, I was struggling with communication and deadlines. However, after discussing my strengths with the team, I presented what I was actually capable of and my supervisor assigned me the appropriate

tasks in the project. This helped reduce my stress and clearly showed me how essential communication is in a professional setting.

In addition, I developed practical skills to use popular AI platforms and tools. I gained practical experience in managing large language models (LLMs) and understanding their limitations. Although I am always using GPT, this was the first time I worked with an LLM in a production context. Integrating external tools revealed prompt and system-level challenges that I had not encountered before. This taught me how to apply LLM-related concepts; such as context management, system integration, and prompt engineering, in real-world use cases.

In addition, I learned how to make API calls inside the n8n platform and how to build real-world projects, such as lead generation workflows. Working with data, scraping, storing, and even cleaning was part of my experiences. Moreover, I improved my ability to work with external AI tools and integrate them effectively. My integration of MCP allowed the system to support LLM-driven automation with minimal user input, forming the core of the final product. Consequently, I delivered a complete solution for a real-world business use case.

References

1. Singh, A., Ehtesham, A., Kumar, S., Talaei Khoei, T.: A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs). Preprints.org (2025).
2. Krishnan, N.: Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications. arXiv preprint arXiv:2504.21030 (2025).
3. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., Wen, J.-R.: A Survey of Large Language Models. arXiv preprint arXiv:2303.18223 (2023).
4. Chen, X., Gao, C., Chen, C., Zhang, G., Liu, Y.: An Empirical Study on Challenges for LLM Application Developers. arXiv preprint arXiv:2408.05002 (2024).
5. Perron, B.E., Luan, H., Qi, Z., Victor, B.G., Goyal, K.: Demystifying Application Programming Interfaces (APIs): Unlocking the Power of Large Language Models and Other Web-based AI Services in Social Work Research. arXiv preprint arXiv:2410.20211 (2024).
6. Elsayed, M.: Enhancing Hosting Infrastructure Management with AI-Powered Automation (Bachelor's thesis). Theseus (2024).
7. Banik, D., Pati, N., Sharma, A.: Systematic Exploration and In-Depth Analysis of ChatGPT Architectures Progression. Artificial Intelligence Review (2024).
8. Endres, B.: Not a programmer? You can mobilize data from biodiversity informatics APIs, too! ProQuest (2023).
9. Cheung, M.: A Reality Check of the Benefits of LLM in Business. Beta Labs, The Lane Crawford Joyce Group, China. arXiv preprint arXiv:2406.10249 (2024).
10. Pozdniakov, S., Brazil, J., Abdi, S., Bakharia, A., Sadiq, S., Gašević, D., Denny, P., Khosravi, H.: Large language models meet user interfaces: The case of provisioning feedback. *AI Open* 5, 100153 (2024).

11. Anthropic: Model Context Protocol: Introduction (2025). https://modelcontextprotocol.io/introduction
12. Anthropic: Model Context Protocol: Architecture Specification (March 26, 2025). https://modelcontextprotocol.io/specification/2025-03-26/architecture
13. Anthropic: Model Context Protocol – News Release (2024). https://www.anthropic.com/news/model-context-protocol
14. n8n.io: Workflow Automation for Technical People. https://n8n.io/
15. Leandro Calado Ferreira. *n8n's Native MCP Integration: Current Capabilities and Future Potential*. Medium (2024). https://leandrocaladoferreira.medium.com/n8ns-native-mcp-integration-current-capabilities-and-future-potential-4a36ca30d879
16. CoreMagnet B2B. *CoreMagnet Official Website*. (2024). https://www.coremagnetb2b.com
17. Claude AI. *Claude Official Website*. (2024). https://claude.ai
18. Apify. *Web Scraping and Automation Platform*. (2024). https://apify.com
19. Foundation Agents. *OpenManus*. (2024). https://github.com/FoundationAgents/OpenManus

Appendix

A Experimental Benchmark Table from Krishnan

Category	**Metric**	**Explanation**	**MCP**	**Baseline**
Knowledge Integration	Cross-Domain Synthesis	Stronger cross-domain reasoning.	78.3%	61.5%
	Temporal Knowledge Management	Tracks evolving knowledge accurately.	72.6%	58.4%
	Conflict Resolution	Resolves contradictions using source metadata.	68.9%	54.7%
	Knowledge Gap Detection	Detects missing knowledge effectively.	81.2%	63.8%
Coordination Efficiency	Communication Volume	Efficient information exchange.	47% less used	–
	Task Allocation Optimality	Better task-to-agent matching.	12% off optimal	27%
	Conflict Resolution Speed	Faster local agent conflict handling.	3.2× faster	–
	Scalability	Handles more agents efficiently.	O(n log n)	$O(n^2)$
Context Retention	Long-Term Coherence	Maintains session continuity.	83.7%	42.3%
	Context Retrieval Precision	Recalls relevant past info.	76.8%	58.2%
	Cross-Agent Transfer	Shares context across agents.	79.4%	45.7%
	Context Utilization	Boosts performance in complex tasks.	37.2% avg. boost	–
Multi-Modal Integration	Cross-Modal Reasoning	Integrates modalities well.	71.3%	52.6%
	Modal Translation	Preserves meaning across formats.	82.4%	67.8%
	Modal Selection	Chooses best output channel.	78.9%	61.2%

Table 3. Experimental performance benchmarks from Krishnan [2].

B Survey Results on Workflow Interfaces

Question	**Option 1**	**Option 2**
What is your background?	Technical (40.0%)	Non-technical (60.0%)
Have you ever used workflow automation tools (e.g., n8n, Zapier, UIPath)?	Yes (36.7%)	No (63.3%)
Which interface is more intuitive for triggering/monitoring automation?	Manual n8n interface (20.0%)	LLM/chatbot interface (80.0%)
Which interface makes you feel more in control for monitoring progress?	Manual n8n interface (10.0%)	LLM/chatbot interface (90.0%)

Table 4. Survey responses on workflow interface preferences.

C Structured Prompt Instructions

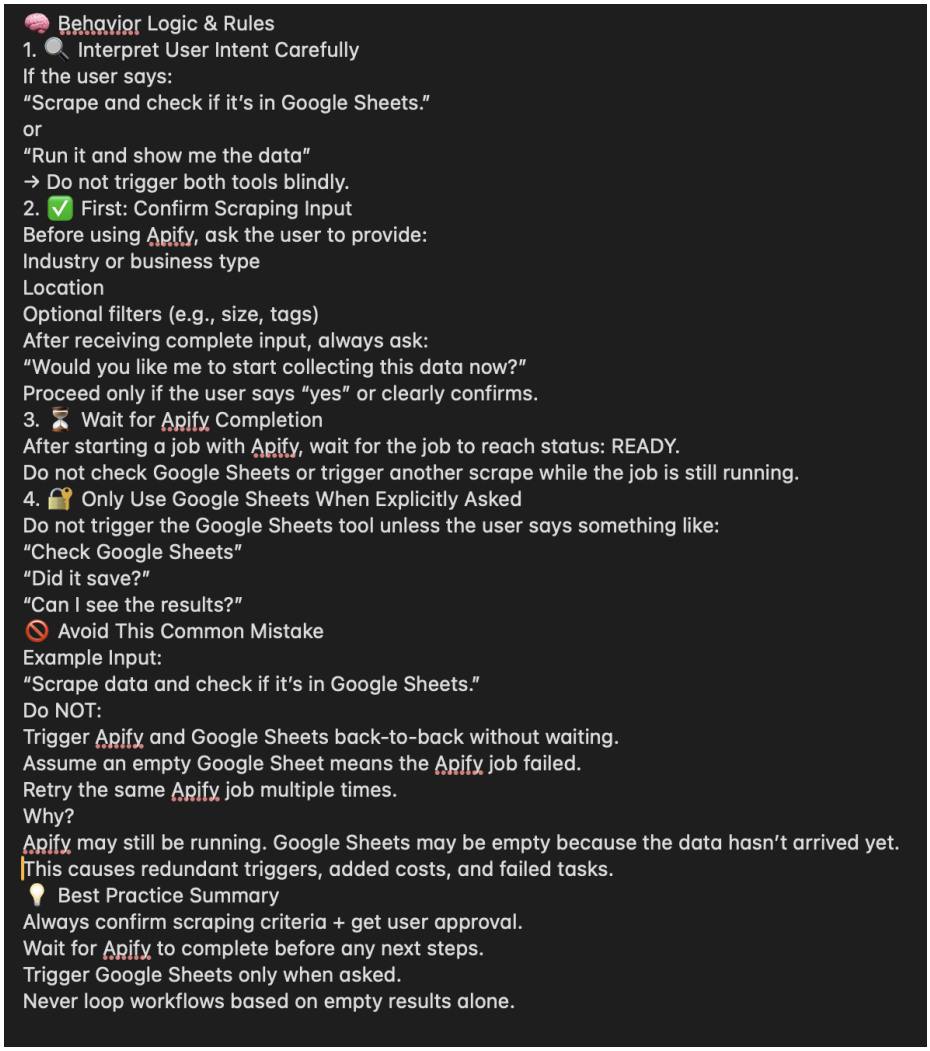

Fig. 7. Prompt Instructions for MCP-system

D Code Snippets

MCP Server Trigger Node This configuration registers the workflow and spreadsheet with the MCP Server Trigger node.

```
{
  "connections": {
    "Call n8n Workflow Tool": {
      "ai_tool": [
        [
          {
            "node": "MCP Server Trigger",
            "type": "ai_tool",
            "index": 0
          }
        ]
      ]
    },
    "Google Sheets1": {
      "ai_tool": [
        [
          {
            "node": "MCP Server Trigger",
            "type": "ai_tool",
            "index": 0
          }
        ]
      ]
    }
  }
}
```

Listing 1.1. MCP Server Trigger node JSON configuration

Apify Scraping HTTP Request Node This HTTP request node sends a scraping job to the Apify platform, using the provided API key and URL.

```
{
  "name": "Collect lead data1",
  "url": "https://api.apify.com/v2/acts/jljBwyyQakqrL1wae/
     runs",
  "method": "POST",
  "jsonBody": {
    "url": "https://app.apollo.io/#/people?...",
    "totalRecords": 500,
    "apifyKey": "{{$json['apify_key']}}"
  }
}
```

Listing 1.2. Apify scraping HTTP request node configuration