FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION

INTERNSHIP REPORT

Melika Abedikoupaei

Element61

03/02/2025-14/03/2025

Tine Claeys

Faculty of Economics and Business Administration
Master Business Engineering
Data Analytics

Academic Year 2024-2025

Contents

1 Introduction To Element61

Element61 is a leading consulting firm specializing in Business Analytics, Performance Management, and Data Science. Founded in Belgium, the company has established itself as a trusted partner for organizations looking to leverage data-driven insights for strategic decision-making. With a strong focus on advanced analytics, artificial intelligence, and big data solutions, Element61 provides tailored services that help businesses optimize performance and gain a competitive edge. The company collaborates with clients across various industries, including finance, healthcare, retail, and manufacturing, ensuring that their analytics strategies align with business goals. Their expertise spans areas such as data warehousing, financial consolidation, predictive modeling, and cloud-based analytics solutions. Element61 prides itself on having a team of highly experienced consultants with deep domain knowledge and technical proficiency in tools like Microsoft Power BI, SAP, IBM Cognos, and other leading analytics platforms. Through a combination of technical excellence, industry best practices, and customer-centric consulting, Element61 has built a strong reputation for delivering high-value analytics solutions that drive business transformation.

2 Challenges In Manual Employee Scheduling

Before the implementation of the automated system, assigning employees to projects at Element61 was a complex and time-consuming process. Users had to manually query the database to verify multiple factors, including the existence of a project, the associated customer, employee availability, employee type, wage details, and other relevant conditions. Each assignment required cross-checking various data points, making the process prone to human errors such as scheduling conflicts, misallocations, and missing critical constraints. Since this approach lacked automation, it required significant effort and increased the likelihood of inconsistencies in workforce planning.

Additionally, determining an employee's available workdays while ensuring alignment with project timelines was another major challenge. Users needed to manually analyze schedules and match them with project durations, which often led to inefficiencies and delays. The absence of an automated system not only made decision-making slower but also limited the ability to optimize resource allocation. These challenges highlighted the need for a more efficient and accurate scheduling solution to reduce manual workload and improve overall planning.

3 Positioning The Internship Assignment And Research Objective

During my internship at Element61, I worked within the AI (Artificial Intelligence) team, where my primary responsibility was to explore and evaluate various AI-based solutions aimed at automating the employee scheduling process. The manual scheduling system at Element61 was not only complex and time-consuming but also prone to human errors and inefficiencies. My task was to

research and assess different AI techniques and models that could streamline this process, reduce mistakes, and optimize resource allocation.

In the broader context of the company, this project played a critical role in enhancing operational efficiency and aligning with Element61's strategic objective of integrating advanced technologies to optimize business processes. The AI solutions under consideration aimed to automate the assignment of employees to projects while accounting for key constraints such as project timelines, employee availability, and required skill sets. This automation was designed to reduce manual effort, minimize scheduling errors, and improve overall accuracy. The central research question guiding my internship was:

"How can AI-based solutions improve the accuracy and efficiency of employee scheduling at Element61?"

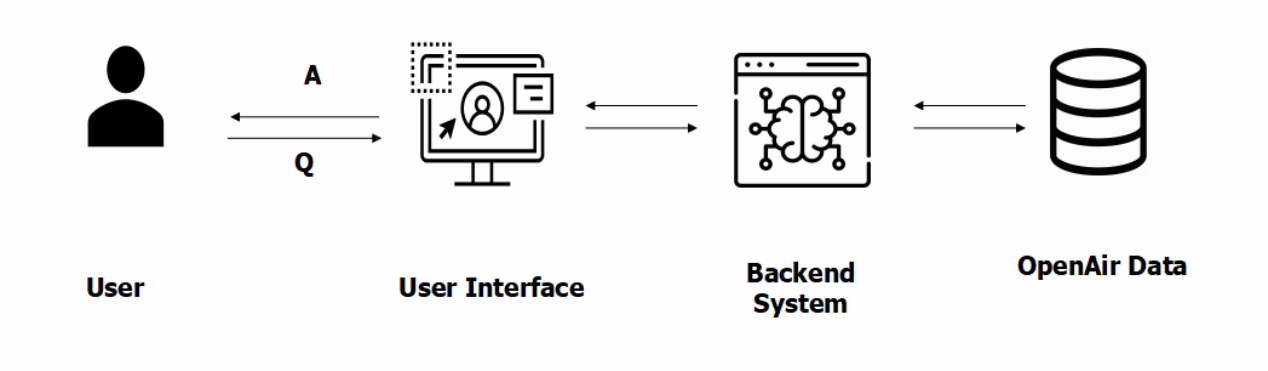
The ultimate goal of my internship was to identify the most effective AI-driven solution to optimize the scheduling process, thereby improving both performance management and resource allocation within the company.

4 Solution Approach To Address The Research Question

To address the research question, **"How can AI-based solutions improve the accuracy and efficiency of employee scheduling at Element61?"**, the solution requires the design and development of a comprehensive system that integrates three key components:

- **A well-structured Database**
- **A robust Backend System**
- **An intuitive User Interface (UI)**

Each of these components plays a crucial role in delivering an AI-based solution for automating the employee scheduling process. Below, I will outline the specific requirements for each component and provide a detailed explanation of how we can approach the development of each part.



Figuur 1: Key components of the solution: User, User Interface, Backend System, and OpenAir Data.

4.1 Data Sources And Structure Documentation

As part of my internship project, I worked with the data provided by Element61 through the OpenAir application, which was used for project management, employee coordination, and scheduling. The information stored in OpenAir encompassed critical data points related to projects, employees, work schedules, and other relevant factors necessary for the employee scheduling process.

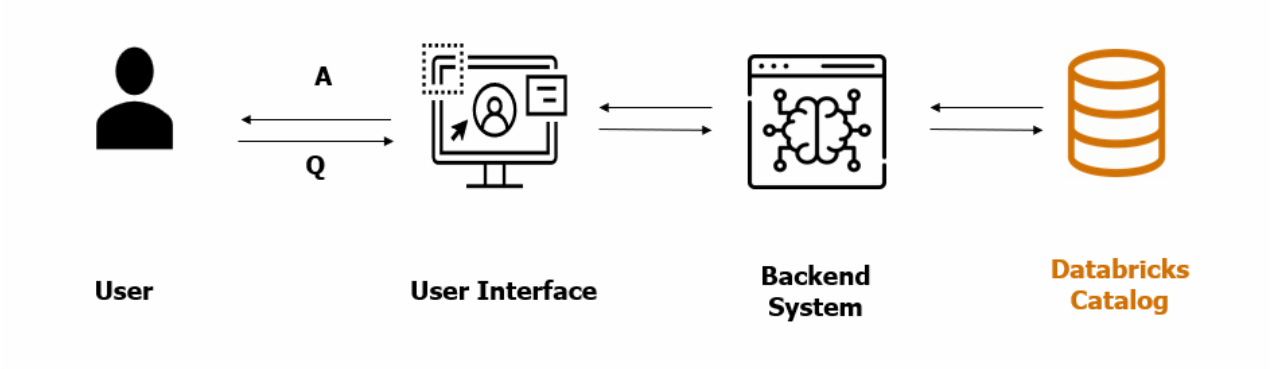
The data from OpenAir was initially imported into the **Azure Storage Account**, where it was stored in a raw format. Once in Azure, the data underwent a cleaning process to ensure that it was accurate, complete, and consistent before being moved to **Databricks** for further processing and analysis. This step was critical in preparing the data for efficient use in AI-based scheduling solutions.

However, there was a lack of an Entity-Relationship Diagram (ERD) for the existing database, which made it difficult to understand the full structure and relationships within the data. To address this, I took the initiative to map out the data relationships and document the data structure by creating an ERD. This process involved the following steps:

1. **Data Exploration:** I began by exploring the raw data from OpenAir, which included employee details, project timelines, employee schedules, and other critical information. I closely analyzed how the data was stored and the relationships between different entities.
2. **Entity Identification**: I identified key entities in the system, including Employee, Project, Schedule, Customer, and Wage. These entities were critical for understanding how to automate the scheduling process, as they contained the relevant attributes for assignment, availability, and project needs.
3. **Relationships Mapping:** I mapped out the relationships between the entities. For example, an Employee could be associated with multiple Projects, and each Project could have one or more Schedules attached to it. Additionally, Employee data needed to be cross-referenced with availability and wage details to ensure that assignments met the project's requirements.
4. **ERD Creation:** After identifying the entities and their relationships, I created an ERD that visually represented the data structure. The ERD included all the necessary entities and their relationships, helping me better understand how to build an automated scheduling system that would align with Element61's needs.
5. **Data Cleaning and Import:** Once the ERD was documented, I proceeded to clean the data and prepare it for processing within Databricks. This step was necessary to ensure that the data was usable and that any inconsistencies were resolved before moving on to the AI-based scheduling solution.

I also made sure to document the ERD for Element61's future use, allowing the team to easily reference the database structure and relationships as they continue to develop and refine their scheduling and analytics systems. This documentation will serve as a valuable resource for future team members or any further improvements to the system.

The ERD and data cleaning process laid the foundation for building an AI-driven scheduling system. By having a clear understanding of the database structure, I was able to design and develop an effective solution that would automate the assignment of employees to projects while minimizing errors and optimizing resource allocation.



Figuur 2: Databricks Catalog as a storage solution for OpenAir Data

4.2 User Interface: Datbricks Apps

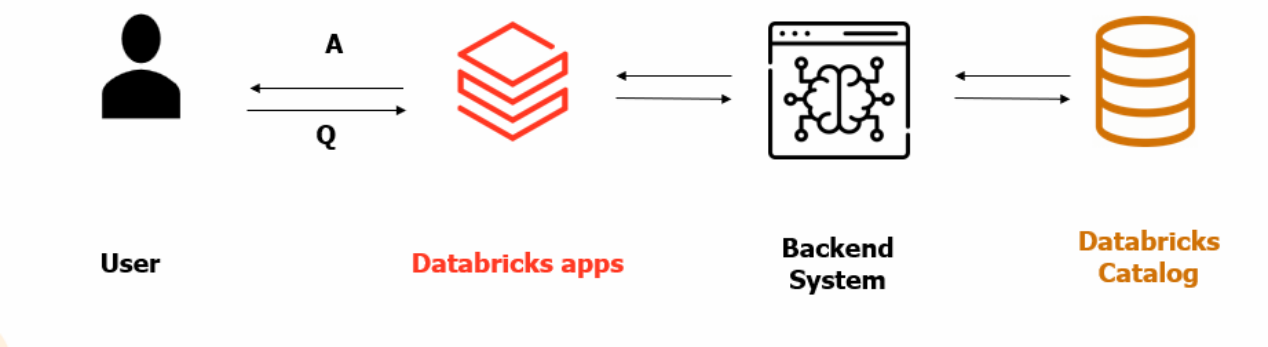
For the interface part of the app, we explored a new feature within Databricks called Databricks Apps. Databricks Apps allows us to build powerful data and AI applications directly within the Databricks platform, without the need to worry about managing complex infrastructure. This feature provides an environment where developers can quickly create, deploy, and manage apps using familiar programming languages like Python, and frameworks such as Dash, Streamlit, or Gradio. The primary advantage of Databricks Apps is its simplicity and integration with the broader Databricks ecosystem, making it a seamless choice for automating business processes like employee scheduling.

Databricks Apps works within the Databricks environment, connecting users, data, and cloud computing resources. It simplifies the process by handling the hosting, security, and data governance aspects automatically. The apps can run on a serverless compute platform, which means Databricks automatically allocates the necessary computing resources without requiring manual intervention. This serverless setup is particularly beneficial because it eliminates the need to manage the underlying infrastructure, making it easier to focus on building the application itself. Moreover, Databricks Apps integrates with Databricks SQL, Databricks Jobs, and other advanced AI features, ensuring that the app can access and utilize the data effectively.

The advantages of Databricks Apps are significant. First, it simplifies the development process by providing an easy-to-use interface and eliminating the need for manual infrastructure management. The serverless compute system ensures that the app can scale as needed without worrying about hardware limitations. Security is also a key benefit, as Databricks provides strong measures such as encryption, access control, and user authentication to protect the data and ensure only authorized users can interact with the application. Furthermore, Databricks Apps integrates seamlessly with the Databricks Unity Catalog, enabling efficient data management and governance.

However, there are a few limitations to consider. For one, users need to authenticate through Databricks, which may present a challenge for people unfamiliar with the platform. Additionally, while Databricks Apps offers great flexibility, the documentation can be somewhat limited, which might make it difficult for beginners to understand certain features or troubleshoot issues. There are also some restrictions, such as a limit on the number of apps you can create within a workspace and file size limitations, which could affect larger or more complex projects.

I chose to use Databricks Apps and Streamlit because it was simple and very well-suited for our goal, which was to test and develop the employee scheduling app. The ease of use, integration with Databricks, and serverless compute capabilities made it an ideal choice for the project. Additionally, I documented the process of creating and using Databricks Apps, providing clear instructions on how to set up and deploy apps, ensuring that the team can easily work with it.



Figuur 3: Databricks Apps serving as the user interface for interaction

4.3 Backend System Evolution: From Databricks Genie To A Multi-Agent System

The backend system of the employee scheduling application plays a crucial role in managing and processing the complex data involved in the scheduling process. It is responsible for handling the logic behind employee assignments, ensuring that all relevant factors, such as project timelines, employee availability, skill sets, and wages, are considered when assigning employees to projects. The backend system automates the scheduling tasks, reducing the manual effort and potential for human error, while also improving the overall efficiency and accuracy of resource allocation. This ensures that the company's workforce is optimally utilized and that the scheduling process is streamlined and scalable.

As part of the solution, I initially considered leveraging Databricks Genie, a tool that allows users to interact with data using natural language queries. Genie converts business questions into SQL queries and generates insights or visualizations based on the available data. This feature was particularly appealing because it could potentially simplify the scheduling process by allowing users to ask complex questions about employee availability and project timelines without needing deep technical knowledge. Genie's ability to generate SQL queries automatically seemed like a promising approach for addressing the challenge of automating employee scheduling, particularly when dealing with large datasets and dynamic schedules.

However, despite its potential, I ultimately could not use **Databricks Genie** because it was still in **Private Preview**, which restricted my ability to integrate it into the Databricks Apps interface. Since

the API was not publicly available, I was unable to implement it at scale or fully incorporate it into the scheduling system. As a result, I had to explore alternative solutions.

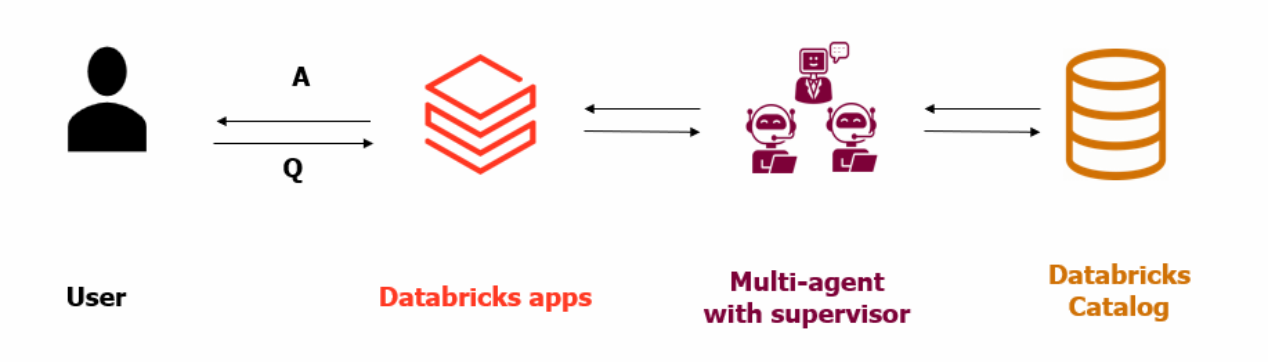
Given the complexity of the scheduling process and the limitations of Genie, I decided to switch to **LangChain**, an agent-based approach that provided more flexibility and customization. LangChain allowed me to design an agent that could intelligently interact with various data sources, apply complex logic, and make decisions based on a broader set of constraints. This solution proved to be a better fit for handling the intricacies of the scheduling process at Element61. LangChain's ability to customize workflows and handle more sophisticated interactions enabled me to create a more robust and scalable backend system, ensuring that the employee scheduling process could be automated effectively and efficiently.

Initially, I implemented a **single-agent** system using LangChain, which **performed well under simpler scheduling conditions**. The agent was capable of processing employee data, considering availability constraints, and generating schedules efficiently. However, as the internship progressed and additional criteria were introduced such as employee preferences, wage constraints, and project priority levels the complexity of the scheduling task increased significantly. This resulted in **longer and more intricate prompts** for the agent, leading to **performance bottlenecks**. The agent struggled to process the growing number of constraints within a reasonable timeframe, ultimately reducing its effectiveness.

To address this challenge, I explored a multi-agent approach using **LangGraph**, a framework designed to manage and coordinate **multiple agents** within a structured workflow. By distributing the workload among specialized agents, I was able to optimize performance and ensure better scalability. Each agent in the system was assigned a specific role, such as handling availability constraints, prioritizing project deadlines, or optimizing wage allocations. This modular design allowed for parallel processing and more efficient handling of complex scheduling logic, leading to faster response times and improved accuracy.

The transition to a multi-agent system provided several key advantages. First, it enhanced scalability by enabling the system to handle an increasing number of constraints without overloading a single agent. Second, it improved efficiency by allowing specialized agents to focus on specific tasks, reducing the complexity of individual prompt executions. Finally, the manager agent within the LangGraph framework acted as a coordinator, ensuring that all agents worked together seamlessly, integrating their outputs into a coherent and optimized schedule. This structured approach significantly improved the overall performance and reliability of the backend system.

By using LangGraph, I successfully addressed the limitations of the single-agent system. This transition not only improved scheduling efficiency but also ensured that the system remained adaptable to future requirements. The multi-agent architecture provided a solid foundation for handling complex scheduling scenarios, reinforcing the importance of a dynamic and scalable backend system in optimizing workforce allocation.

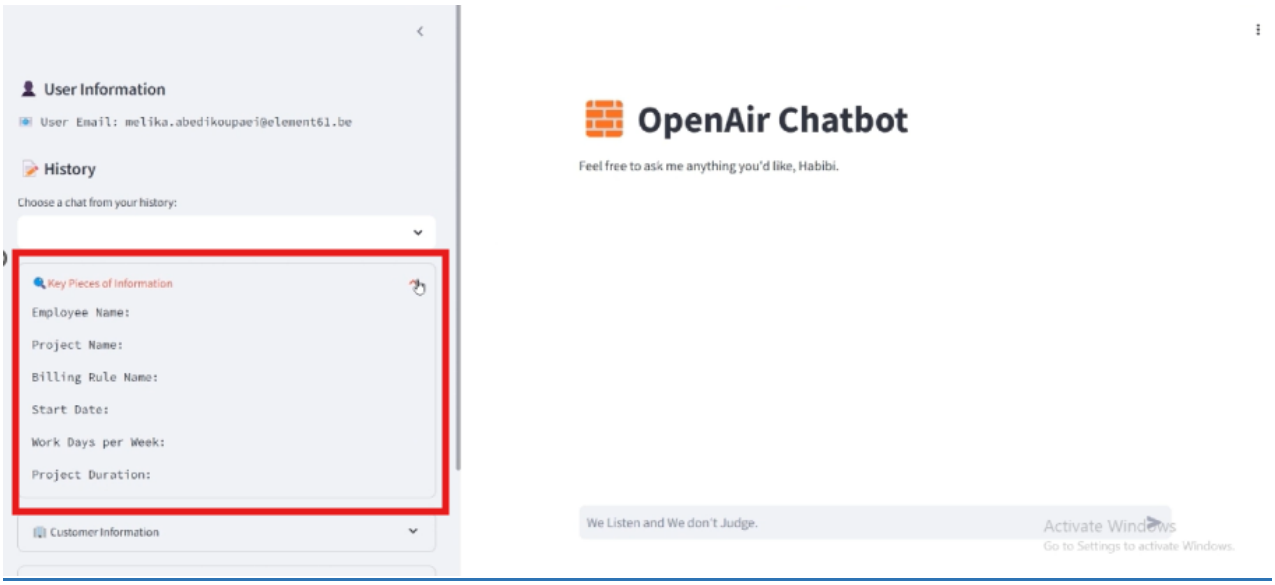


Figuur 4: Multi-agent backend system with a supervisory agent powered by LangGraph

5 Final Solution

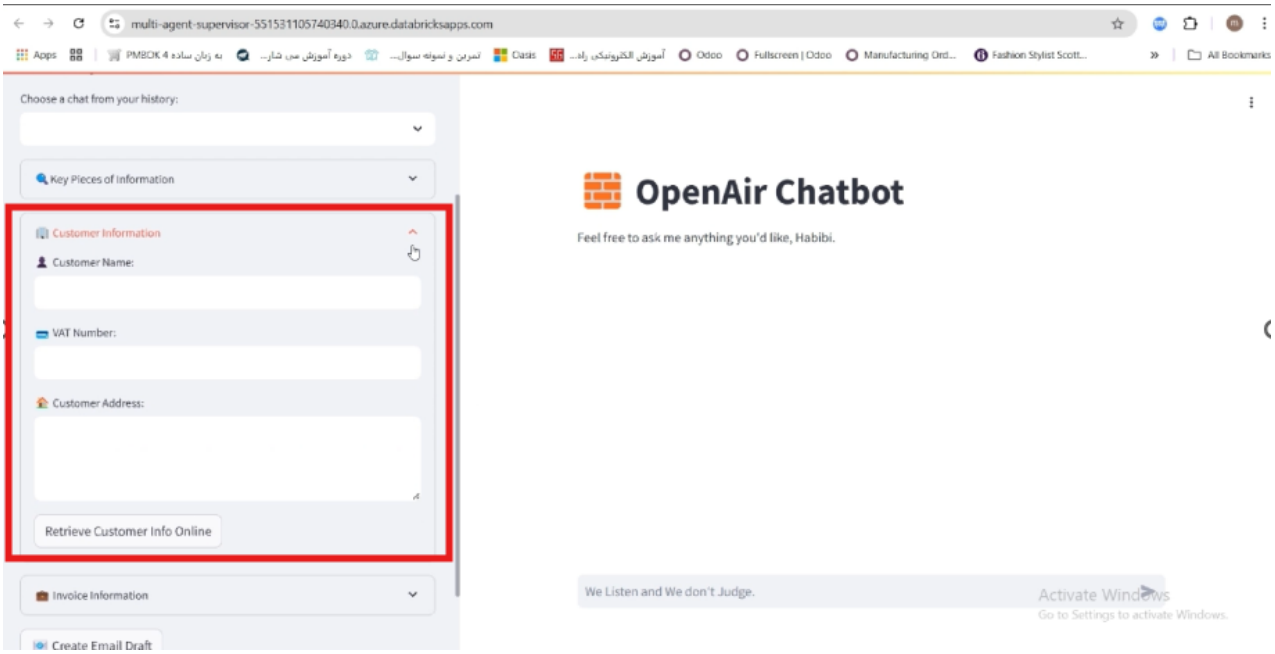
In the final implementation of our backend system, I developed a fully integrated multi-agent solution that not only streamlined employee scheduling but also assisted users in extracting and managing customer data efficiently. This solution was designed to enhance user experience by automating key processes, reducing manual effort, and ensuring all necessary information was structured and readily available. The core of this system relied on multiple agents working together seamlessly, with an interactive chat interface and a structured data-saving mechanism integrated into the application.

As shown in the image below, the application interface features a chat function where users can interact with the scheduling agent. This agent retrieves relevant information from databases, processes employee availability, and ensures that all scheduling constraints are met. Additionally, the agent saves important extracted details to the sidebar in Streamlit, providing users with an organized overview of key information. This sidebar functionality allows users to quickly reference extracted data without needing to scroll through previous chat interactions, improving accessibility and efficiency.



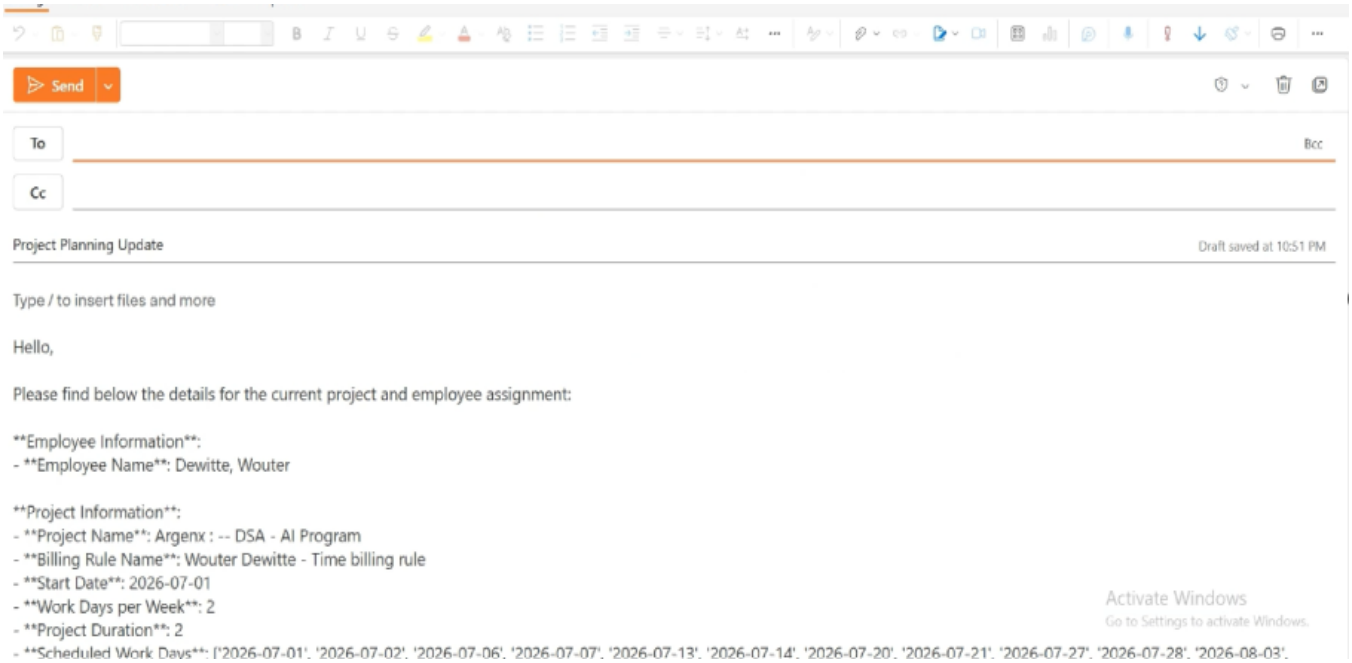
Figuur 5: Efficient scheduling and data management with an interactive chat and organized sidebar

Beyond scheduling, I created an additional agent within the sidebar to assist users in extracting customer information. By the name of the customer, this agent connects to a tool for searching the internet to retrieve the VAT number and address of the customer online. By integrating this automation, I eliminated the need for manual searches, making the process faster and reducing the risk of errors. This feature proved especially useful for administrative and financial workflows, ensuring that all necessary client details were readily available for documentation and communication purposes.



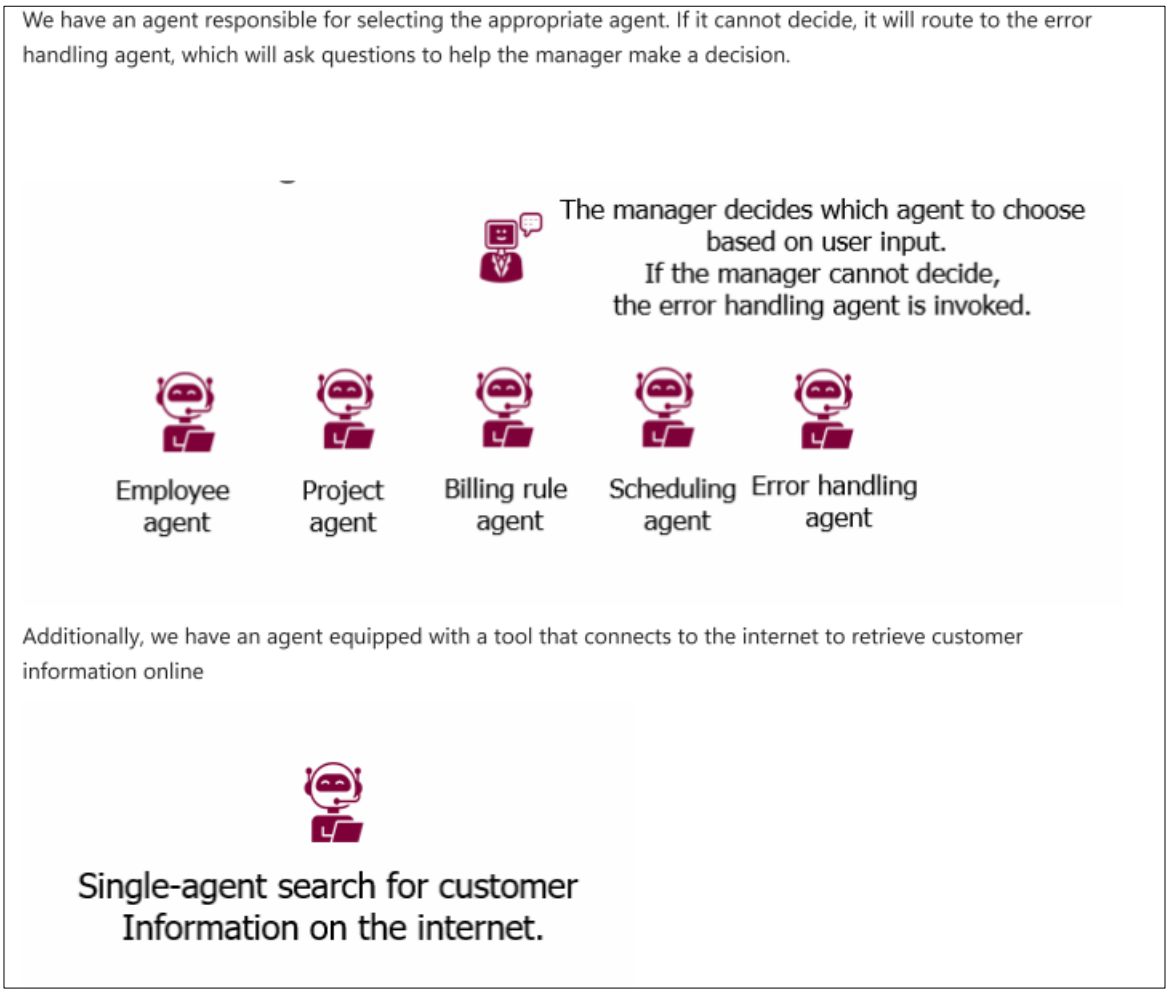
Figuur 6: Auto-fetch customer details: Quick VAT and address lookup

Once all required information is collected, users can finalize the process with a single action. By clicking on the "Send Email" button, all the extracted details are formatted into a structured email template and automatically drafted in Outlook. This integration simplifies communication by ensuring that emails contain accurate and complete information, reducing the likelihood of miscommunication or missing details. Users can then review the email before sending it to the appropriate recipient, making the entire workflow seamless and efficient.



Figuur 7: One-click email generation: Automatically draft emails with all extracted details

In the architecture diagram below, I illustrate how the multi-agent system was structured and deployed in the final solution. Each agent was designed to handle specific tasks, ensuring that the system remained modular and scalable. The interaction between these agents—managed through LangGraph—allowed for efficient data retrieval, processing, and automation, ultimately enhancing the overall functionality of the backend system. This final implementation successfully addressed the challenges I encountered throughout the project, demonstrating the power and flexibility of multi-agent systems in complex business applications.



Figuur 8: Multi-agent architectures

6 CONCLUSION

During this internship, I delivered:

- **A chatbot that is functionally working**: The chatbot is operational and performs as expected. However, it requires a period of thorough testing. Due to time limitations, we could not conduct extensive testing.
- **Documentation of code for single-agent and multi-agent systems**: This includes a detailed breakdown of the implementation, structure, and logic behind both approaches, making it easier for future developers to build upon or modify the system.
- **Documentation of my learnings on AI/BI Genie space in Databricks**: Insights gained from working with Genie, including its capabilities, limitations, and how it compares to other AI-driven tools.
- **Documentation on Databricks Apps**: Covering the architecture, usage, and integration of Databricks Apps within the project.

Element61, of course, can improve the app by implementing the following enhancements:

- **Error Handling:** Enhancing error-handling mechanisms will provide a smoother user experience by ensuring the system can gracefully recover from unexpected issues. Better exception management and debugging tools will improve system reliability.
- **Log Storage Optimization:** Currently, logs are stored in Delta tables on Databricks, which is not ideal for performance and scalability. Exploring more optimized storage solutions, such as a dedicated logging system or cloud-based alternatives, can improve retrieval efficiency and reduce storage costs.
- **Chat History Navigation:** Users should be able to click on a chat ID and seamlessly navigate past conversations. Implementing a structured chat history with better indexing will enhance user accessibility and allow for a more intuitive experience.
- **Streamlit Limitations:** While Streamlit is excellent for prototyping and testing, it is not the best choice for a production-ready application. Exploring more scalable and robust frameworks, such as a dedicated web framework or a custom front-end solution, will enhance performance and user experience.

By addressing these areas, Element61 can further refine the application, ensuring it meets the highest standards of usability, efficiency, and scalability.