FOM University of Applied Sciences for Economics and Management

university location Cologne

Bachelor Thesis

in the study course Business Informatics

to obtain the degree of

Bachelor of Science (B.Sc.)

on the subject

Implementation of an AI-Supported Centralized Procurement Interface to Enhance Procurement Processes in a Large-Scale Enterprise

by

Dominik Fey

Advisor:	Prof. Dr. Bernd Ulmann
Matriculation Number:	601373
Submission:	October 13, 2024

Contents

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

IaC Infrastructure-as-Code

IEEE Institute of Electrical and Electronics Engineers

MVP Minimum Viable Product

NLP Natural Language Processing

NLU Natural Language Understanding

PO Product Owner

RE Requirements Engineering

List of Symbols

Sperrvermerk

Die vorliegende Abschlussarbeit mit dem Titel ‘Implementation of an AI-Supported Centralized Procurement Interface to Enhance Procurement Processes in a Large-Scale Enterprise’ enthält unternehmensinterne Daten der Firma Deutsche Telekom AG. Daher ist sie nur zur Vorlage bei der FOM sowie den Begutachtern der Arbeit bestimmt. Für die Öffentlichkeit und dritte Personen darf sie nicht zugänglich sein.

Cologne, October 13, 2024

(Ort, Datum)

(Eigenhändige Unterschrift)

1 Relevance

Dies soll eine LaTeX-Vorlage für den persönlichen Gebrauch werden. Sie hat weder einen Anspruch auf Richtigkeit, noch auf Vollständigkeit. Die Quellen liegen auf Github zur allgemeinen Verwendung. Verbesserungen sind jederzeit willkommen.

1.1 Objective

Kleiner Reminder für mich in Bezug auf die Dinge, die wir bei der Thesis beachten sollten und LaTeX -Vorlage für die Thesis.

1.2 Structure of the Paper

Kapitel 2 enthält die Inhalte des Thesis-Days und alles, was zum inhaltlichen erstellen der Thesis relevant sein könnte. In Kapitel 3 Methodology findet ihr wichtige Anmerkungen zu LaTeX , wobei die wirklich wichtigen Dinge im Quelltext dieses Dokumentes stehen (siehe auch die Verzeichnisstruktur in Abbildung.

1.3 Technical Stack Relevance

The selection of a robust and integrated technical stack is pivotal for the successful implementation of sophisticated software solutions, particularly when developing a chatbot designed to understand and fulfill complex customer needs. The project presented in this bachelor's thesis involves the creation of a chatbot that leverages Natural Language Processing (NLP) to interpret customer inquiries and match them with relevant items from a supplier catalog. To achieve this, the backend relies on Python and Haystack, while the frontend utilizes Vue, and the deployment is managed through containerization and orchestration technologies like Docker and Kubernetes.

1.3.1 Backend Technologies

Python serves as the backbone of the application due to its extensive ecosystem and its ability to seamlessly integrate various libraries and frameworks that facilitate rapid

prototyping and the development of complex data-driven functionalities.[1,2] Its prominence in the fields of machine learning and data science makes it an ideal choice for implementing a chatbot that requires advanced Natural Language Understanding (NLU) capabilities.[3,4] Specifically, Python's compatibility with NLP frameworks like Haystack and many libraries ensures that the chatbot can parse user inputs and perform context-aware semantic searches.[5] This capability is crucial for accurately interpreting customer needs and mapping them to appropriate products or services in the supplier catalog.

The integration of Haystack within this project serves as a cornerstone for the development of an intelligent, search-driven chatbot, which is designed to address the intricate nature of customer inquiries. Haystack, a robust open-source NLP framework, employs a sophisticated Reader-Retriever architecture that harmonizes the capabilities of both information retrieval and deep semantic understanding. This dual approach capitalizes on advanced NLP methodologies to enhance the chatbot's performance in extracting pertinent information from extensive datasets.[6]

Notably, rather than utilizing the standard BERT model, this implementation leverages OpenAI's GPT-4o model within the Haystack framework. This allows the system to engage in nuanced contextual interpretation, thereby significantly improving the precision of semantic searches. The choice of GPT-4o is particularly advantageous in question-answering scenarios, as it allows the chatbot to comprehend the subtleties of customer queries and generate responses that are not only contextually relevant but also demonstrate a high degree of language understanding.[7]

Moreover, Haystack's modular architecture and extensible Application Programming Interfaces (APIs) offer a high degree of flexibility, facilitating seamless integration within the chatbot's overall architecture. This ensures that the processes of searching and retrieving supplier catalog data are executed with optimal accuracy and efficiency. Consequently, Haystack's inclusion in the technical stack is not merely contributory to the current system's capabilities but also establishes a solid foundation for prospective advancements and refinements in the chatbot's functional repertoire.

The backend system also incorporates PostgreSQL as its database solution. PostgreSQL's support for complex queries and its capability to handle structured data are essential for

1 cf. *Shrivastava*, *S.*, 2024, p. 12.
2 cf. *Christensen*, *S.* et al., 2022, pp. 240–241.
3 cf. *Lortie*, *C. J.*, 2022, p. 1.
4 cf. *Joshi*, *A.*, *Tiwari*, *H.*, 2024, p. 85.
5 cf. *Fareez*, *M. M. M.*, *Thangarajah*, *V.*, *Saabith*, *S.*, 2020, p. 21.
6 cf. *Krishnamoorthy*, *V.*, 2021, p. 236.
7 cf. *Syed*, *Z. H.* et al., 2021, pp. 943–944.

managing and accessing the supplier information and product details stored within the system.[8] The integration of PostgreSQL ensures that the chatbot can quickly and efficiently retrieve the necessary data, thereby reducing latency and enhancing the overall user experience.

For monitoring and observability, the project employs Langfuse and OpenTelemetry, which provide comprehensive tracing and metrics collection across the microservices architecture.[9] This is particularly relevant given the experimental nature of the prototype, where understanding system performance and identifying potential bottlenecks are crucial for iterative development and refinement. By utilizing these tools, the project gains valuable insights into the behavior of the chatbot, allowing for continuous improvement and optimization.

FastAPI serves as the web framework for the backend, offering a high-performance environment that supports asynchronous programming.[10] This choice is particularly relevant for the chatbot, as it enables handling multiple concurrent requests with minimal overhead, ensuring that the application remains responsive even under heavy loads.

The project also leverages the `uv` package for dependency management and deployment configuration. `uv` simplifies the process of configuring Python dependencies and allows for a smoother deployment process by ensuring compatibility and consistency between various package versions.[11]

1.3.2 Frontend Technologie

On the frontend, Vue.js and Vuetify are utilized to create an intuitive and responsive user interface. The decision to use Vue.js stems from its reactive nature and modular architecture, which align with the need for a maintainable and easily extensible codebase.[12,13] It is effective for developing a chatbot interface that needs to present complex data in an accessible manner, while also allowing for dynamic updates based on user interactions.[14]

[8] cf. *Abbasi*, *M.* et al., 2024, pp. 23–24.
[9] cf. *Thakur*, *A.*, *Chandak*, *M. B.*, 2022, p. 15014.
[10] cf. *Chen*, *J.*, 2023, p. 9.
[11] cf. *Cano Rodríguez*, *J. L.*, 2024, URL last accessed on 2024-10-05.
[12] cf. *Kaluža*, *M.*, *Vukelic*, *B.*, 2018, p. 268.
[13] cf. *Li*, *N.*, *Zhang*, *B.*, 2021, pp. 1–2.
[14] cf. *Mokoginta*, *D.*, *Putri*, *D. E.*, *Wattimena*, *F. Y.*, 2024, p. 493.

1.3.3 Deployment and Infrastructure Management

Deployment is managed through a combination of Docker, Kubernetes, and Terraform. Docker's role in containerizing the application ensures that the entire software stack can be encapsulated and deployed consistently across various environments.[15] This is essential for a project like this, where different iterations of the prototype may need to be tested in different setups. Kubernetes, in turn, provides the orchestration needed to manage these containers, allowing for automated scaling and high availability.[16] The use of Terraform as an Infrastructure-as-Code (IaC) tool ensures that cloud resources can be provisioned and managed efficiently, providing a stable and reproducible deployment environment.[17]

In summary, each component of the technical stack has been carefully selected to meet the unique requirements of the chatbot project. The combination of Python and Haystack provides robust NLP capabilities for understanding and processing user inputs, while FastAPI support real-time interactions. PostgreSQL ensures efficient data management, and Langfuse and OpenTelemetry offer the necessary monitoring tools. On the frontend, Vue.js and Vuetify deliver a responsive and interactive user interface, and the deployment stack, comprised of Docker, Kubernetes, and Terraform, guarantees scalability and reliability. This cohesive selection of technologies forms a solid foundation for the development of a chatbot that not only meets the functional requirements but also adheres to best practices in software engineering.

[15] cf. *Openja*, *M.* et al., 2022, p. 191.
[16] cf. *Carrión*, *C.*, 2022, pp. 2, 7–8.
[17] cf. *N.*, *N.*, *Pub*, *I.*, 2023, p. 24.

2 Fundamentals

3 Methodology

3.1 Requirements Engineering

3.1.1 Introduction to Requirements Engineering According to Sommerville

Requirements Engineering (RE), as outlined by Ian Sommerville, is a multifaceted process aimed at systematically defining a system's specifications.[18] It comprises five interconnected activities: Requirements Documentation, Requirements Elicitation, Requirements Analysis and Negotiation, Requirements Validation, and Requirements Management.[19] Each activity has distinct objectives but collectively ensures the system satisfies both functional and non-functional requirements.[20]

Requirements Documentation serves as the foundation, formalizing needs, expectations, and constraints into a structured format, such as a Software Requirements Specification. This document bridges communication between stakeholders and developers and evolves with the project, supporting traceability and consistency. It is not merely a list of requirements but provides contextual rationale for each, enhancing clarity and stakeholder alignment.[21]

The process begins with Requirements Elicitation, which involves proactively engaging with stakeholders, including end-users and domain experts, to extract comprehensive requirements. Sommerville highlights this phase as an active process of understanding the problem domain and reconciling conflicting needs through techniques like interviews, workshops, and prototyping. This activity is vital for capturing all relevant requirements and establishing a shared understanding of the system's objectives.[22]

Next, Requirements Analysis and Negotiation transforms these inputs into a coherent and conflict-free set of requirements. Techniques such as modeling and prioritization help refine requirements, while negotiation resolves trade-offs between competing stakeholder priorities. This ensures a balanced and agreed-upon set of requirements, addressing concerns like performance versus cost or user flexibility versus security.[23]

Requirements Validation, a critical phase, ensures requirements reflect stakeholder needs and are feasible within constraints. Using techniques such as formal inspections, peer reviews, and prototyping, validation identifies ambiguities and contradictions early,

[18] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.5.
[19] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.11.
[20] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.7–8.
[21] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, pp. 38–40.
[22] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.64–65.
[23] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.112–113.

minimizing costly changes later and ensuring the system aligns with stakeholder expectations.[24]

Requirements Management, the final activity, involves tracking and controlling changes as the project evolves. Requirements are inherently dynamic, shifting with stakeholder understanding, new regulations, or market conditions. Sommerville's approach emphasizes maintaining traceability and documenting changes to ensure all modifications are transparent and approved, reducing risks and maintaining requirement integrity.[25]

3.1.2 Selection of Requirements Engineering Approach

The selection of the RE approach by Sommerville for this project was based on tailoring each RE activity to systematically uncover and refine requirements, thus providing a robust methodological foundation.

For Requirements Documentation, Sommerville's recommendations to "Define a Standard Document Structure" and "Make the Document Easy to Change" were implemented. The standardized structure ensured clarity and traceability, while the flexibility to modify the document supported iterative updates.[26,27] This approach facilitated rigorous traceability, deemed more critical for a prototype-focused project than alternatives like summarizing requirements, which prioritize readability over the detailed documentation needed to manage evolving requirements.[28]

In Requirements Elicitation, the strategies "Use Scenarios to Elicit Requirements," "Define Operational Processes," and "Prototyping Poorly Understood Requirements" were selected due to their effectiveness in refining complex and abstract requirements through iterative development and visualization. "Use Scenarios to Elicit Requirements" enabled precise visualization of user interactions, crucial for capturing the chatbot's nuanced behavior.[29] "Define Operational Processes" provided a structured method for visualizing workflows, ensuring accurate capture of procedural requirements, while "Prototype Poorly U nderstood Requirements" translated ambiguous requirements into tangible artifacts for stakeholder discussions.[30,31] These strategies were preferred over approaches like

[24] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.190–191.
[25] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.216–217.
[26] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.41.
[27] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.60.
[28] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.47.
[29] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.99.
[30] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.102–103.
[31] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.94–96.

"Collect Requirements From Multiple Viewpoints," as the prototype's limited scope did not necessitate broad stakeholder involvement from the outset.[32]

For Requirements Analysis and Negotiation, the project adopted "Plan for Conflicts and Conflict Resolution" due to its structured approach to identifying and resolving conflicting requirements.[33] This systematic method facilitated discussions on competing priorities and consensus-building, making it more suitable than less structured alternatives like "Define System Boundaries" or "Provide Software to Support Negotiations," which do not support the iterative, dialogic nature of ongoing negotiations in prototype development.[34,35]

The Requirements Validation phase applied "Organize Formal Requirements Inspections" and "Use Prototyping to Verify Requirements." The former enabled comprehensive identification of inconsistencies through structured reviews, while the latter facilitated visual validation and iterative feedback.[36,37] The decision to forego "Define Validation Checklists" was based on its less adaptable structure, which is not as effective for validating the evolving visual and interactive components of the project.[38]

Finally, in Requirements Management, "Define Policies for Requirements Management" was implemented to maintain traceability and manage changes systematically. Establishing clear policies ensured that all requirements could be efficiently tracked and modified in response to evolving project needs.[39] The project opted not to use a database for managing requirements, focusing instead on robust change management policies to ensure a controlled and flexible environment for reviewing and integrating changes.[40] This emphasis on policies over tools supported the dynamic nature of the prototype's development while preserving project coherence and stakeholder alignment.

3.1.3 Application of Requirements Engineering in the Project

The application of RE in this project adhered to an iterative and collaborative methodology, incorporating structured documentation, continuous stakeholder engagement, and interactive design tools to refine and validate requirements throughout the development

[32] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.90.
[33] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.125–127.
[34] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.114.
[35] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.121.
[36] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.195–196.
[37] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.203–204.
[38] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.200.
[39] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.221–222.
[40] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.236.

lifecycle. This approach, guided by Sommerville's principles, enabled effective communication, alignment, and feedback across all parties.

The foundation, the Requirements Documentation, began with a structured Word document containing a requirements table, which served as the central repository for capturing, organizing, and updating requirements. This document ensured systematic traceability and provided a consistent foundation for ongoing modifications as new insights emerged. The example from Institute of Electrical and Electronics Engineers (IEEE), mentioned in Sommerville's book, is not used, because for a Prototype it is to much.[41] A simpler version was created.

During Requirements Elicitation, workshops with domain experts were conducted to deepen understanding of the procurement process and to capture core requirements. These sessions were complemented by ad-hoc meetings throughout development to address emerging needs and clarify ambiguities. The elicitation phase leveraged iterative prototyping, progressing from initial mockups in Miro to refined designs in Figma, and ultimately to functional screens in Vue. This incremental approach allowed stakeholders to engage with evolving system representations, offering targeted feedback and further refining requirements at each stage.

The Requirements Analysis and Negotiation phase synthesized feedback gathered during workshops and prototyping sessions. Weekly meetings provided a forum to discuss and resolve conflicting priorities among stakeholders, while bi-weekly management meetings ensured strategic alignment and secured high-level approval for changes. This continuous engagement facilitated a dynamic negotiation process, allowing for re-prioritization and refinement of requirements based on stakeholder feedback and evolving project constraints.

For Requirements Validation, a combination of systematic reviews and interactive prototyping was used. The prototypes, evolving from initial sketches in Miro to detailed designs in Figma and implemented screens in Vue, enabled stakeholders to visualize how their requirements would be realized, aiding in early identification of inconsistencies or misalignments. Regular reviews ensured alignment between the documented requirements and the evolving system, minimizing the risk of discrepancies.

Throughout Requirements Management, change control policies were strictly followed, ensuring that all requirement modifications were systematically documented, reviewed, and approved. The structured requirements table was continuously updated to reflect these changes, maintaining traceability and transparency. This disciplined approach to requirements management preserved control over evolving requirements and ensured that stakeholders remained informed of any adjustments.

[41] cf. *Sommerville*, *I.*, *Sawyer*, *P.*, 1997, p.42–43.

3.2 Prototyping

3.2.1 Introduction to the Prototyping Methodology According to Floyd

The prototyping approach developed by Christiane Floyd represents a structured methodology used primarily in software development to improve communication between developers and users, reduce misunderstandings, and ultimately enhance the quality of the final product.[42] This methodology provides an alternative to the traditional linear, phase-oriented development process by introducing a dynamic element of iteration and feedback.[43] As a result, it facilitates a more interactive and user-centered development process.[44]

Floyd's prototyping process is structured as a cyclical sequence of four distinct steps: functional selection, construction, evaluation, and further use.
The first step, Functional Selection, involves identifying the specific functions that the prototype should demonstrate. The selected functionalities are derived from the relevant work tasks to ensure meaningful demonstrations, while acknowledging that the prototype does not need to represent the final product comprehensively. This allows for a degree of flexibility in the selection and prioritization of features to be included in the prototype.[45]

The second step, Construction, entails building the prototype using techniques and tools that enable rapid development and easy adjustments. At this stage, the focus is not on developing a fully functional system, but rather on demonstrating and assessing specific aspects of the final product. This approach enables the prototype to act as a tool for exploring different solutions and gathering feedback.[46]

Evaluation, the third step in the process, serves as the cornerstone of the prototyping methodology. In this step, feedback from all relevant stakeholders—including potential users—is collected and analyzed to refine and guide subsequent development stages. This iterative process ensures that the prototype evolves in alignment with user expectations and needs.[47]

The final step, Further Use, determines the prototype's role after the evaluation phase. Depending on its effectiveness and the degree to which it meets requirements, the prototype can be discarded, modified for continued use, or serve as a foundation for the final product.

[42] cf. *Floyd, C.*, 1984, p.2–3.
[43] cf. *Floyd, C.*, 1984, p.3.
[44] cf. *Floyd, C.*, 1984, p.3–4.
[45] cf. *Floyd, C.*, 1984, p.4.
[46] cf. *Floyd, C.*, 1984, p.4.
[47] cf. *Floyd, C.*, 1984, p.4–5.

This flexibility is critical in accommodating evolving requirements and objectives, making the prototyping approach particularly valuable in contexts where specifications are expected to change frequently.[48]

Floyd's methodology also categorizes prototyping into three primary approaches, each based on the goals and context of development: exploratory prototyping, experimental prototyping, and evolutionary prototyping.[49]
Exploratory Prototyping is primarily used to clarify requirements and foster creative cooperation between developers and users during the early stages of development. It is particularly useful when there is a lack of clarity on the system's final objectives, as it allows for broad experimentation and refinement of ideas before committing to a specific solution.[50]

In contrast, Experimental Prototyping focuses on testing proposed solutions to validate specific hypotheses, such as user interface design, system performance, or algorithm feasibility. This approach might involve techniques such as full functional simulation or human interface simulation to verify that the proposed design meets the intended objectives.[51]

Finally, Evolutionary Prototyping treats the prototype as a system that continuously evolves to adapt to changing requirements over time. Each version of the prototype serves as a basis for the next iteration, incorporating new insights and user feedback. This approach is especially beneficial in scenarios where requirements are expected to change frequently, rendering a static set of requirements impractical.[52]

To support these prototyping processes, various techniques and tools can be utilized.[53]
Modular Design, for instance, encourages the use of small, independent modules that can be replaced or refined as needed, thereby facilitating iterative development and easing integration into the final product.[54]
Additionally, Dialogue Design plays a crucial role in ensuring that the user interface is adaptable and transparent, which enables effective evaluation and modification of the user experience.[55]
Furthermore, Simulation techniques allow for simulating aspects of the final system that

[48] cf. *Floyd*, *C.*, 1984, p.5.
[49] cf. *Floyd*, *C.*, 1984, p.6.
[50] cf. *Floyd*, *C.*, 1984, p.6–7.
[51] cf. *Floyd*, *C.*, 1984, p.8–10.
[52] cf. *Floyd*, *C.*, 1984, p.10–12.
[53] cf. *Floyd*, *C.*, 1984, p.12.
[54] cf. *Floyd*, *C.*, 1984, p.12.
[55] cf. *Floyd*, *C.*, 1984, p.12.

are not yet fully implemented, enabling the assessment of system performance and user interaction without the need for a complete implementation.[56]

3.2.2 Selection of Prototyping Approach

Given the context and objectives of this project, the decision was made to adopt the experimental prototyping approach. This choice is rooted in the fact that the project requirements have already been well-defined through a comprehensive requirements engineering process, thus eliminating the need for exploratory prototyping. The clear specification of functionalities and user expectations ensures that the focus can shift from understanding requirements to validating and testing specific design choices.

Furthermore, the experimental approach is particularly well-suited for scenarios where a prototype is intended to serve as a preliminary proof of concept rather than a foundation for incremental development. This aligns perfectly with the anticipated lifecycle of the prototype in this project, which is expected to be discarded once the Minimum Viable Product (MVP) phase begins. As the project moves towards the MVP stage, a fresh start will be made, incorporating only validated concepts and findings from the experimental prototype. Therefore, evolutionary prototyping is not applicable, as it is primarily designed for projects that involve iterative refinement and continuous evolution of the same prototype.

The experimental prototyping approach allows for focused experimentation with various design elements, interface interactions, and technical implementations, all within a controlled environment that does not necessitate long-term integration into the final product.

3.2.3 Application of Experimental Prototyping in the Project

The experimental prototyping approach will be implemented in this project with a focus on validating user interface designs, interactions, and core functionalities against predefined requirements. The development of the prototype will leverage a set of carefully selected techniques that facilitate rapid iteration and feedback.

A key technique employed is simulation, which is used to mimic certain system behaviors without integrating the prototype into live production environments. This decision is motivated by the limited scope of the prototype and the intention to avoid disruptions to existing systems. By relying on test data instead of actual production data, the prototype

[56] cf. *Floyd*, *C.*, 1984, p.13.

can simulate real-world scenarios and provide valuable insights into its performance and user experience.

Moreover, the prototype will not attempt to implement every function in its final depth and breadth. Instead, certain features will be simulated to convey the look and feel of the system, providing a realistic representation of how the final product would function. This is where Modular Design plays a crucial role. By leveraging modular design principles , the prototype can separate the user interface from the underlying logic and backend functionalities. This allows specific modules to be developed exclusively for the UI, simulating the presence of certain features without the need for fully developed backend logic. For instance, UI elements such as buttons, forms, and interactive components can be displayed and interacted with as if they were functional, even though the backend processing is either simulated or entirely absent. This approach enables the developer to receive early feedback on key aspects of the design and functionality without committing extensive resources to full-scale implementation.

By focusing on these aspects, the prototype will serve as a learning tool, guiding the refinement of requirements and design choices before moving into the more resource-intensive MVP phase.

4 Requirements Engineering

4.1 Requirement Document

The document is structured to facilitate the collection of requirements from stakeholders in a clear and systematic manner. At its core, the document consists of a table where stakeholders are required to fill out specific fields related to the system requirements. The table is designed to capture both functional and non-functional requirements, providing the foundation for further analysis and prioritization by the Product Owner (PO). The empty template that was circulated to stakeholders for completion can also be found in Appendix 1 of the paper, providing a reference for how the requirements were initially collected.

Table 1: Requirement Collection Table [57]

Requirement ID (to be filled by PO)	Description (to be filled by Stakeholder)	Functional/Non-Functional (to be filled by Stakeholder)	Stakeholder (to be filled by Stakeholder)	Priority (to be filled by PO)	Status (to be filled by PO)	Comments/Notes (to be filled by PO)	Last Updated (to be filled by PO)

Each row in the table corresponds to a unique requirement. The first field, the "Requirement ID," is reserved for the PO to assign a unique identifier to each requirement once it has been submitted. This ensures proper traceability and management of requirements across various phases of the project. Following this, the "Description" field is intended for stakeholders to provide a detailed explanation of what they expect the system to do. It emphasizes the need for precision, encouraging stakeholders to specify whether the requirement pertains to a particular system behavior or performance characteristic.

The table also includes a "Functional/Non-Functional" column, where stakeholders are asked to categorize their requirement. If the requirement pertains to what the system should do, it is considered functional. Non-functional requirements, on the other hand, describe how the system performs in terms of attributes such as speed, reliability, or security. To accommodate any uncertainties, the stakeholders are informed that they can leave this classification to be determined during subsequent discussions with the PO, should they find it difficult to distinguish between the two.

Another key component is the "Stakeholder" field, where individuals or teams responsible for submitting the requirement are identified. This enables clear accountability and provides a way to trace the origins of each requirement back to its respective department or stakeholder.

[57] Own illustration

To ensure that stakeholders understand how to complete their portion of the table, a comprehensive guidance section follows the table. This guidance elaborates on each field that needs to be filled, providing examples and clarifications. It emphasizes the importance of clarity in the description, guiding stakeholders to avoid ambiguity, and includes explanations of both functional and non-functional requirements. Additionally, the guidance outlines how to identify themselves as stakeholders, ensuring that their input is properly documented.

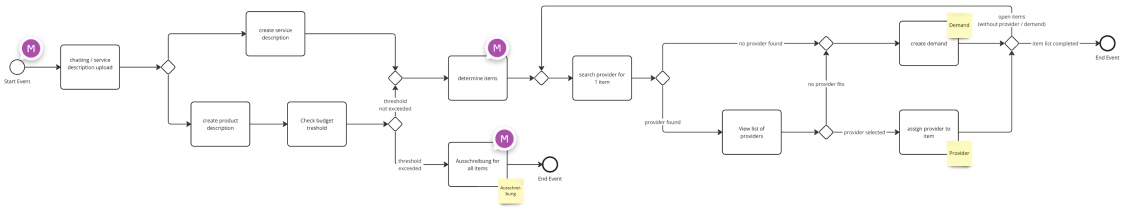
The remainder of the table, including fields such as "Priority," "Status," "Comments/Notes," and "Last Updated," is reserved for the PO to complete after the requirements have been collected. These fields will be used to prioritize, track the progress, and document any changes or updates to each requirement, ensuring that the project remains aligned with both stakeholder expectations and project constraints as it evolves.

This structure ensures a well-organized process for gathering, documenting, and managing system requirements, fostering collaboration between the stakeholders and the PO.

4.2 Requirement Elicitation

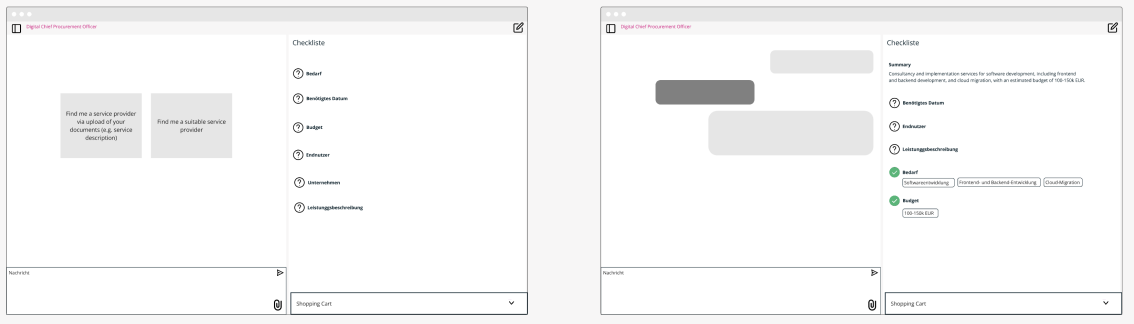
During the Requirement Elicitation phase, a combination of structured workshops and ad-hoc meetings via Microsoft Teams played a central role in defining both the system requirements and the associated business processes. These sessions provided a platform for stakeholders, domain experts, and the PO to collaborate closely, ensuring that the chatbot's functionalities would align with the operational needs of the procurement process. Throughout these meetings, key requirements were identified and discussed in detail, laying the groundwork for the chatbot's behavior, such as understanding user needs, searching through supplier catalogs, and facilitating procurement requests. A critical outcome of these discussions was the definition of core business processes, which were visualized in Miro to provide a clear understanding of how the chatbot would integrate into existing workflows. One example business process can be found in figure 1. By mapping out these processes collaboratively, the stakeholder and PO was able to capture the procedural steps the chatbot would follow, ensuring that both functional and non-functional requirements were well understood and aligned with the organizational objectives.

Figure 1: Defined Business Process in Miro [58]



As these discussions progressed, certain requirements emerged as more complex or abstract, making them difficult to fully articulate through conversation alone. In these cases, prototyping became an essential tool for refining and simplifying these requirements. The Miro mockups, 2 examples shown in figure 2, provided an early-stage visualization of how users would engage with the system, focusing on key interaction points, such as the user inputting a request, the chatbot interpreting the need, and the subsequent search of supplier catalogs for relevant offerings. All Miro Screen Mock-Ups can be found in Appendix 2.1. These early designs enabled stakeholders to visualize the flow of operations and provided a concrete foundation for discussions, allowing for immediate feedback and refinement. The outcome of these discussions clarified the scope of the chatbot's tasks and helped prioritize functionalities, ensuring that the system would meet the diverse requirements of both end-users and the internal procurement processes.

Figure 2: Screen Mock-Ups in Miro [59]



Building on these initial mockups, the project transitioned to more detailed prototyping in Figma. The initial concepts developed in Miro were translated into high-fidelity interactive designs, offering a more precise depiction of the user interface and interaction flows. The Figma prototypes, examples shown in figure 3 and figure 4, allowed stakeholders to experience a more realistic representation of the chatbot, including how it would handle user queries, present search results from supplier catalogs, and guide the user through the

[58] Own illustration
[59] Own illustration

selection and checkout process. All Figma Screen Prototypes can be found in Appendix 2.2. This phase proved critical in refining the system's requirements, as the visual and interactive nature of the Figma screens enabled stakeholders to provide targeted feedback on specific elements of the interface, including the chatbot's responses, the organization of data, and the overall user experience. As the designs became more detailed, the stakeholder was able to refine aspects such as response times, data handling, and catalog integration, ensuring that the chatbot's functionalities aligned with both the technical capabilities of the system and the users' expectations.

Figure 3: Screen Prototype in Figma (Lightmode) [60]

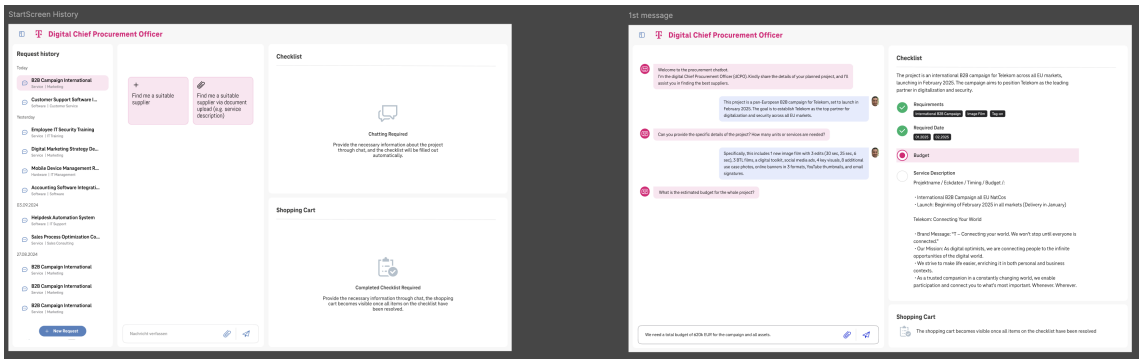
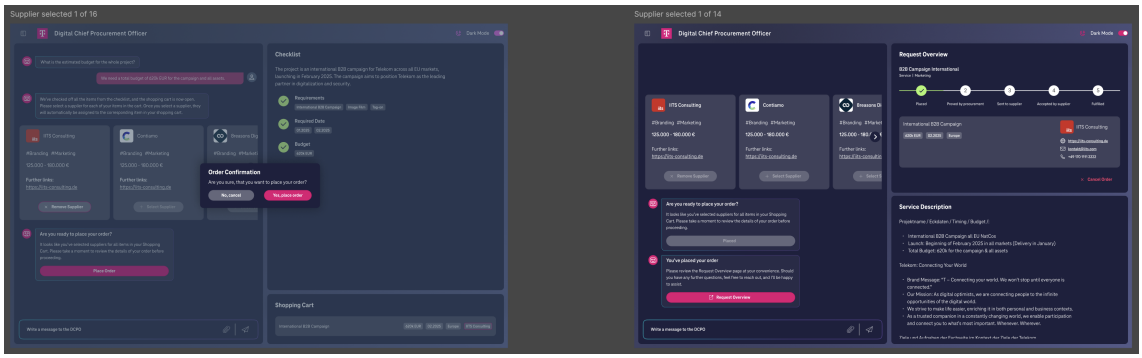


Figure 4: Screen Prototype in Figma (Darkmode) [61]



Once the Figma designs were validated, the project moved to the development of functional builds using Vue. These builds marked the first instance of the system in a live environment, where the chatbot could be tested in real-time against actual user inputs. The Vue builds, examples shown in figure provided the opportunity to verify that the system met the documented requirements in practice, with users engaging directly with the chatbot interface to simulate procurement scenarios. During these tests, it became evident whether the chatbot's decision-making processes were functioning as expected and whether its

60 Own illustration
61 Own illustration

performance, particularly in searching supplier catalogs and handling user requests, met the defined non-functional requirements. The feedback gathered during this phase led to further adjustments, with particular focus on optimizing the chatbot's responsiveness and ensuring the smooth integration of supplier data into the system's workflow.

The culmination of the Requirement Elicitation phase was the consolidation of all gathered requirements into a structured document that served as the definitive guide for the project's development. This document captured the full scope of the system's functional and non-functional needs, including the precise behavior of the chatbot when interacting with users, the expected performance standards, and the overall user experience considerations. This requirements document evolved alongside the project, incorporating feedback from stakeholders at each stage—from the Miro mockups to the Figma prototypes and finally the Vue builds—ensuring that the requirements were fully traceable throughout the entire development process. The result of this phase was a clear, comprehensive set of requirements that not only reflected the needs of the users but also aligned with the technical constraints and objectives of the system, providing a solid foundation for the subsequent stages of development.

4.3 Requirement Analysis and Negotiation

The Requirement Analysis and Negotiation phase focused on refining the gathered requirements and resolving any conflicts or ambiguities that arose during the elicitation phase. This stage was essential for ensuring that the system would meet both the technical constraints and the diverse needs of the various stakeholders involved in the project. The process of analysis and negotiation was conducted primarily through regular weekly meetings with stakeholders and bi-weekly sessions with senior management, ensuring that the project stayed aligned with both operational expectations and strategic objectives.

The weekly meetings with stakeholders formed the backbone of the Requirement Analysis process. These sessions, which brought together representatives from the procurement teams, technical staff, and other key user groups, provided a structured environment for ongoing discussions about the system's requirements. During these meetings, the feedback gathered from the prototyping phases was reviewed in detail, with particular attention given to how the system's functionalities were evolving in response to the requirements initially defined in the workshops.

A key aspect of these weekly sessions was the continuous prioritization of requirements. The stakeholders provided input on which features needed to be refined or developed

first, helping the PO maintain focus on core functionalities, such as the chatbot's ability to interpret and act on user requests. These discussions also enabled the PO to identify any gaps in the requirements or areas where further clarification was needed, ensuring that the chatbot's functionality would align with the intended procurement processes.

In addition to prioritization, the weekly meetings were also used to resolve conflicts between competing requirements. As different stakeholders had varying expectations of how the chatbot should function, these sessions served as a forum for negotiation. For example, there were often trade-offs between modern design and user accessibility, or between performance and cost. By bringing these issues to the forefront in a collaborative setting, the PO was able to reach consensus on the best approach to take. These discussions were particularly valuable in clarifying the chatbot's core functionalities, such as determining the appropriate level of detail the chatbot should provide when responding to user queries and how it should handle requests that were not well-defined.

Through these weekly discussions, the PO was able to continuously refine the system's requirements, ensuring that they remained both feasible and aligned with the overall project goals. As new insights emerged, they were integrated into the requirements document, ensuring that the evolving needs of the stakeholders were captured and could be tracked throughout the development process.

In parallel with the weekly stakeholder sessions, bi-weekly meetings with senior management were held to ensure that the project remained aligned with the broader strategic objectives of the organization. These meetings provided a higher-level perspective on the system's development, focusing on ensuring that the chatbot's functionalities would not only meet the immediate operational needs but also support the company's long-term goals for procurement efficiency and digital transformation.

The bi-weekly management meetings were critical in securing approvals for key decisions made during the Requirement Analysis phase. For instance, as the system's development progressed and the trade-offs between performance and cost became more apparent, it was essential to have senior management's input to determine the acceptable levels of investment in system performance. These meetings also provided an opportunity to discuss the prioritization of high-impact features, ensuring that the project remained focused on delivering value in line with the company's strategic initiatives.

One important aspect of the bi-weekly management meetings was the focus on risk mitigation. As potential risks were identified during the stakeholder meetings, such as delays in catalog integration or challenges related to system scalability, these issues were brought to the attention of senior management. This allowed the PO to make informed

decisions about how to allocate resources and manage risks effectively, ensuring that the system development remained on track and that any potential bottlenecks could be addressed early.

The management meetings also served as a platform to reflect the progress made in the weekly stakeholder sessions. By reviewing the decisions made and the requirements refined during those discussions, senior management was able to provide feedback and ensure that the system's development remained consistent with the organization's vision. This continuous feedback loop between the weekly stakeholder sessions and the bi-weekly management reviews ensured that the system's development remained agile and responsive to both operational and strategic needs.

The iterative nature of the Requirement Analysis and Negotiation phase allowed for continuous refinement of the chatbot's requirements. The weekly stakeholder meetings provided detailed input on the system's functionality and performance, while the bi-weekly management meetings ensured that these decisions aligned with the organization's broader goals. Throughout this phase, the requirements document was continuously updated, reflecting the outcomes of each discussion and ensuring that every decision was documented and traceable.

4.4 Requirement Validation

The Requirement Validation phase was critical in ensuring that the requirements gathered and refined in previous phases were accurate, complete, and feasible. The goal was to verify that the functional and non-functional needs of the system were well understood and aligned with stakeholder expectations. This phase relied on interactive tools such as Figma prototypes to visualize and simulate the chatbot's core functionalities. In addition, initial backend conversations were tested during this phase to verify key technical aspects of the chatbot's processing logic.

The Figma prototypes played a crucial role in validating the system's design and behavior. These prototypes allowed stakeholders to interact with a detailed and high-fidelity representation of how the chatbot would operate in practice. They offered a clear view of the user interface, navigation, and interaction flows, simulating the chatbot's procurement process without requiring full backend integration or live data.

Stakeholders were invited to explore these prototypes, providing feedback on whether the user experience and functionality met the documented requirements. The prototypes simulated key features such as the chatbot's response to user inputs, search functionality

within supplier catalogs, and the overall flow from request initiation to the presentation of recommendations. This validation helped identify any areas where the chatbot's design needed to be adjusted to better meet business objectives. Based on stakeholder feedback, improvements were made to enhance usability and ensure the chatbot's interaction with users was intuitive and aligned with the intended procurement processes.

In addition to validating the user interface through Figma prototypes, early backend conversations were also validated in this phase. This validation focused on ensuring that the backend could interpret user input and return appropriate responses, even though the system was not yet fully integrated or using live data. By simulating conversations between the chatbot and users, the PO was able to validate that the backend logic was functioning correctly and that the chatbot could handle simple dialogues.

These initial backend tests provided valuable insights into how the chatbot's NLP would work in practice, allowing the PO to refine the backend's ability to interpret requests and provide relevant responses. Testing these conversations during the validation phase ensured that both the frontend prototypes and backend logic were progressing in alignment with the documented requirements.

Throughout the Requirement Validation phase, regular review sessions were conducted with stakeholders to gather ongoing feedback. These sessions were essential for ensuring that the evolving prototypes and backend functionalities were meeting expectations. The interactive nature of these reviews allowed stakeholders to provide detailed feedback on the chatbot's behavior, leading to continuous refinements of both the design and backend logic.

Key areas of focus during these reviews included validating the accuracy of the chatbot's responses, ensuring smooth navigation, and confirming that the user interface was intuitive. The feedback provided by stakeholders was incorporated into subsequent iterations of the prototypes and backend logic, ensuring that the system continued to evolve in line with the project's goals.

After successfully validating the requirements through both the Figma prototypes and early backend tests, the PO was ready to move into the actual development phase. The transition marked the end of the Requirement Validation phase, signaling that the requirements had been thoroughly vetted and were ready for implementation. With the core functionalities validated and feedback incorporated, the development of the full prototype, including both the backend and frontend systems, began using test data.

4.5 Requirement Management

The Requirement Management phase focused on maintaining control over the evolving requirements throughout the project. As new insights emerged from stakeholder feedback and validation sessions, it was essential to manage changes systematically to ensure that all modifications were properly documented and traceable. A structured approach was adopted to track every change in the requirements, ensuring that the PO and stakeholders remained aligned on any updates or shifts in priorities. This process helped avoid scope creep and ensured that only approved changes were implemented, safeguarding the integrity of the original project objectives.

Regular reviews of the requirements document were conducted to reflect the evolving nature of the project, particularly as feedback from the validation phase led to refinements in the system's functionality. By maintaining up-to-date records of all changes, the PO was able to ensure consistency between the original requirements and the final deliverables. This approach ensured that any changes were justified, reviewed, and incorporated without disrupting the overall project flow, providing a robust framework for managing the dynamic aspects of the chatbot's development. The complete set of these finalized documents can be found in Appendix 2.3 for further reference.

5 Prototyping

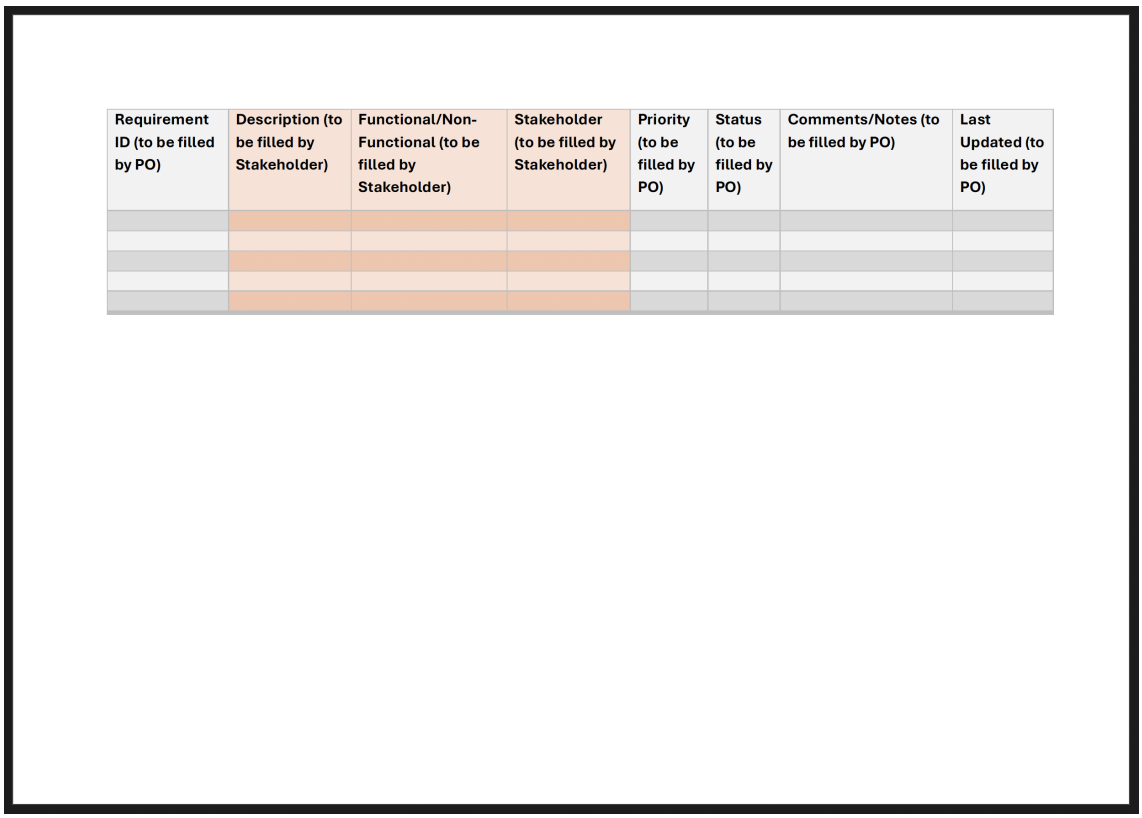
6 Fazit

Wünsche Euch allen viel Erfolg für das 7. Semester und bei der Erstellung der Thesis. Über Anregungen und Verbesserung an dieser Vorlage würde ich mich sehr freuen.

Appendix

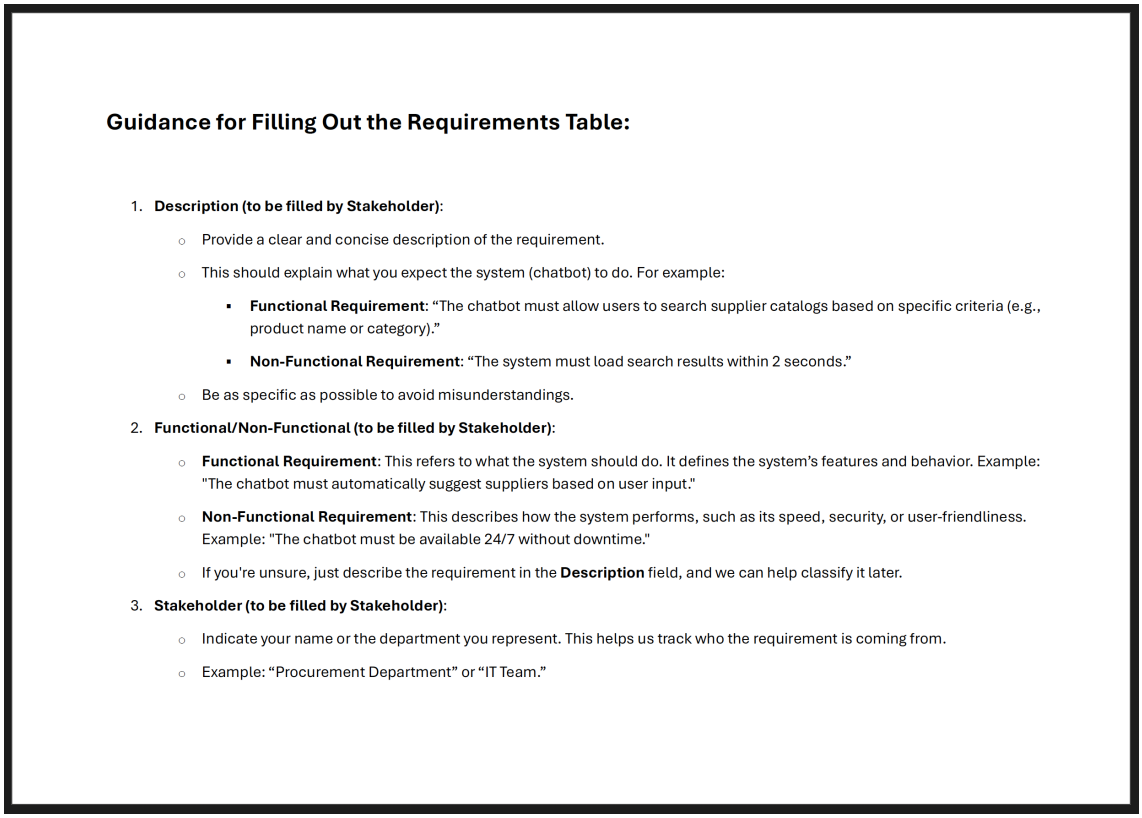
Appendix 1: Requirement Document

Figure 5: Requirement Document (Page 1)



Requirement ID (to be filled by PO)	Description (to be filled by Stakeholder)	Functional/Non-Functional (to be filled by Stakeholder)	Stakeholder (to be filled by Stakeholder)	Priority (to be filled by PO)	Status (to be filled by PO)	Comments/Notes (to be filled by PO)	Last Updated (to be filled by PO)

Figure 6: Requirement Document (Page 2)



Guidance for Filling Out the Requirements Table:

1. **Description (to be filled by Stakeholder)**:
   - Provide a clear and concise description of the requirement.
   - This should explain what you expect the system (chatbot) to do. For example:
     - **Functional Requirement**: "The chatbot must allow users to search supplier catalogs based on specific criteria (e.g., product name or category)."
     - **Non-Functional Requirement**: "The system must load search results within 2 seconds."
   - Be as specific as possible to avoid misunderstandings.
2. **Functional/Non-Functional (to be filled by Stakeholder)**:
   - **Functional Requirement**: This refers to what the system should do. It defines the system's features and behavior. Example: "The chatbot must automatically suggest suppliers based on user input."
   - **Non-Functional Requirement**: This describes how the system performs, such as its speed, security, or user-friendliness. Example: "The chatbot must be available 24/7 without downtime."
   - If you're unsure, just describe the requirement in the **Description** field, and we can help classify it later.
3. **Stakeholder (to be filled by Stakeholder)**:
   - Indicate your name or the department you represent. This helps us track who the requirement is coming from.
   - Example: "Procurement Department" or "IT Team."

Appendix 2: Requirement Elicitation

Appendix 2.1: Screen Mock-Ups in Miro

Appendix 2.2: Screen Prototypes in Figma

Appendix 2.3: Filled-out Requirement Documents

Bibliography

Abbasi, *Maryam*, *Bernardo*, *Marco V.*, *Váz*, *Paulo*, *Silva*, *José*, *Martins*, *Pedro* (2024): Adaptive and Scalable Database Management with Machine Learning Integration: A PostgreSQL Case Study, in: Information, 15 (2024), Nr. 9, p. 574

Carrión, *Carmen* (2022): Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges, in: ACM Comput. Surv. 55 (2022), Nr. 7, 138:1–138:37

Chen, *Junqiao* (2023): Model Algorithm Research Based on Python Fast API, in: Frontiers in Science and Engineering, 3 (2023), Nr. 9, pp. 7–10

Christensen, *Scott*, *Brown*, *Marvin*, *Haehnel*, *Robert*, *Church*, *Joshua*, *Catlett*, *Amanda*, *Schofield*, *Dallon*, *Brannon*, *Quyen*, *Smith*, *Stacy* (2022): A Python Pipeline for Rapid Application Development (RAD), in: s.l., 2022-01-01, pp. 240–243

Fareez, *MMM*, *Thangarajah*, *Vinothraj*, *Saabith*, *Sayeth* (2020): POPULAR PYTHON LIBRARIES AND THEIR APPLICATION DOMAINS, in (2020)

Floyd, *Christiane* (1984): A Systematic Look at Prototyping, in: *Budde*, *Reinhard*, *Kuhlenkamp*, *Karin*, *Mathiassen*, *Lars*, *Züllighoven*, *Heinz* (eds.), Approaches to Prototyping, Berlin, Heidelberg: Springer, 1984, pp. 1–18

Joshi, *Ankush*, *Tiwari*, *Haripriya* (2024): An Overview of Python Libraries for Data Science | Journal of Engineering Technology and Applied Physics, in ()

Kaluža, *Marin*, *Vukelic*, *Bernard* (2018): Comparison of Front-End Frameworks for Web Applications Development, in: Zbornik Veleučilišta u Rijeci, 6 (2018), pp. 261–282

Krishnamoorthy, *Venkatesh* (2021): Evolution of Reading Comprehension and Question Answering Systems, in: Procedia Computer Science, Big Data, IoT, and AI for a Smarter Future, 185 (2021), pp. 231–238

Li, *Nian*, *Zhang*, *Bo* (2021): The Research on Single Page Application Front-end Development Based on Vue, in: Journal of Physics: Conference Series, 1883 (2021), Nr. 1, p. 012030

Lortie, *Christopher J.* (2022): Python and R for the Modern Data Scientist, in: Journal of Statistical Software, 103 (2022), pp. 1–4

Mokoginta, *Deyidi*, *Putri*, *Desfita Eka*, *Wattimena*, *Fegie Yoanti* (2024): Developing Modern JavaScript Frameworks for Building Interactive Single-Page Applications, in: International Journal Software Engineering and Computer Science (IJSECS), 4 (2024), Nr. 2, pp. 484–496

N., *Nivedhaa*, *Pub*, *Iaeme* (2023): Evaluating Devops Tools and Technologies for Effective Cloud Management, in: 1 (2023), pp. 20–32

Openja, *Moses*, *Majidi*, *Forough*, *Khomh*, *Foutse*, *Chembakottu*, *Bhagya*, *Li*, *Heng* (2022): Studying the Practices of Deploying Machine Learning Projects on Docker, in: Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering, EASE '22, New York, NY, USA: Association for Computing Machinery, 2022-06-13, pp. 190–200

Shrivastava, *Samiksha* (2024): Design and Implementation of Chatbot Using Python, in: IJFMR - International Journal For Multidisciplinary Research, 5 (), Nr. 6

Sommerville, *Ian*, *Sawyer*, *Pete* (1997): Requirements Engineering: A Good Practice Guide, 1st ed., USA: John Wiley & Sons, Inc., 1997-03, 404 pp.

Syed, *Zeeshan Haque*, *Trabelsi*, *Asma*, *Helbert*, *Emmanuel*, *Bailleau*, *Vincent*, *Muths*, *Christian* (2021): Question Answering Chatbot for Troubleshooting Queries Based on Transfer Learning, in: Procedia Computer Science, 192 (2021), pp. 941–950

Thakur, *Aadi*, *Chandak*, *M. B.* (2022): A Review on Opentelemetry and HTTP Implementation, in: International journal of health sciences, 6 (2022), Nr. S2, pp. 15013–15023

Internet sources

Cano Rodríguez, *Juan Luis* (2024): Python Packaging Is Great Now: ‘uv‘ Is All You Need, <https://dev.to/astrojuanlu/python-packaging-is-great-now-uv-is-all-you-need-4i2d> (2024-08-10) [Access: 2024-10-05]

Declaration in lieu of oath

I hereby declare that I produced the submitted paper with no assistance from any other party and without the use of any unauthorized aids and, in particular, that I have marked as quotations all passages which are reproduced verbatim or near-verbatim from publications. Also, I declare that the submitted print version of this thesis is identical with its digital version. Further, I declare that this thesis has never been submitted before to any examination board in either its present form or in any other similar version. I herewith disagree that this thesis may be published. I herewith consent that this thesis may be uploaded to the server of external contractors for the purpose of submitting it to the contractors' plagiarism detection systems. Uploading this thesis for the purpose of submitting it to plagiarism detection systems is not a form of publication.

Cologne, 13.10.2024

(Location, Date)

(handwritten signature)