UNIVERSIDADE ABERTA **INSTITUTO SUPERIOR TÉCNICO**

Leveraging AI-Powered chatbots for enhanced requirements elicitation and documentation in software engineering

Nicolae Zatic

Master's Degree in Information and Enterprise Systems

Supervisor: Dr. Rita Andreia da Conceição Marques

2025

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration. I further declare that I have fully acknowledged Disciplinary Regulations of the Universidade Aberta (regulation published in the official journal Diário da República, 2.ª série, N.º 215, de 6 de novembro de 2013).

Universidade Aberta, 11 October 2025

Nicolae Zatic

Acknowledgments

I want to thank Professor Rita Marques for her time, guidance, and encouragement throughout this thesis.

I am also grateful to my professional colleagues, whose support, insightful discussions, and generosity with their time helped bring this work to life. My sincere thanks also extend to my employer for providing the resources and flexibility that made this research possible.

To my friends, far in distance but always close at heart. Thank you.

To my parents, for every sacrifice that allowed me to reach this point and for instilling in me the value of hard work — *Mulțumesc, mamă și tată, pentru tot. Fără voi, nimic nu ar fi fost posibil.*

My deepest thanks go to my beloved Inês. Thank you for being my constant source of strength, keeping the house together, and all the extra sleep hours that allowed me to finish this work. Embracing fatherhood with you has been the journey of a lifetime.

And finally, to my sweet Leonardo, for being a constant reminder that there is a world of wonder and love beyond these pages. This work is for you, with all my heart.

Abstract

Traditional requirements engineering is often inefficient due to communication gaps, poor documentation, and time-consuming manual processes. This thesis investigates AI-powered chatbots as a solution to enhance requirements elicitation and documentation. Using a Design Science Research approach informed by a literature review and analyst interviews, key challenges were identified like knowledge fragmentation and low trust in Artificial intelligence (AI). A prototype chatbot was then designed with a Retrieval-Augmented Generation (RAG) core, grounded in project documentation and integrated with Jira and Confluence. Evaluation with five analysts showed the chatbot's potential to reduce manual effort in drafting artifacts such as user stories and documentation, and improve access to dispersed knowledge. However, user trust was low, as the prototype could not link to verifiable sources. This research concludes that trust is a critical architectural requirement requiring features like provenance layers and confidence scoring. These results position AI chatbots not as autonomous replacements but as collaborative assistants that augment human expertise, freeing analysts for critical thinking tasks and stakeholder communication. Future work should implement rigorous data hygiene to ensure the knowledge base is reliable and validate these findings with a larger, more diverse participant sample.

Keywords

Requirements elicitation; Large Language Model; Retrieval-Augmented Generation; Prompt engineering, Automation.

Resumo

A engenharia de requisitos tradicional é frequentemente ineficiente devido a falhas de comunicação, documentação fraca e processos manuais demorados. Esta tese investiga os chatbots reforçados por inteligência artificial (IA) como uma solução para melhorar a obtenção e documentação de requisitos. Utilizando uma abordagem de *Design Science Research* baseada numa revisão da literatura e entrevistas com analistas, foram identificados alguns desafios, como a fragmentação do conhecimento e confiança reduzida na IA. Em seguida, foi desenhado um protótipo de chatbot com um núcleo de geração reforçada por recuperação, baseado na documentação do projeto e integrado com o Jira e o Confluence. A avaliação com cinco analistas mostrou o potencial do chatbot para reduzir o esforço manual na elaboração de artefactos, como requisitos e documentação, e melhorar o acesso à base de conhecimento. No entanto, a confiança dos utilizadores foi baixa, uma vez que o protótipo não conseguia estabelecer ligações a fontes verificáveis. Esta investigação conclui que a confiança é um requisito crítico que requer funcionalidades como camadas de proveniência e classificação de confiança. Estes resultados posicionam os chatbots reforçados por IA não como substitutos autónomos, mas como assistentes colaborativos que aumentam a experiência humana, libertando os analistas para tarefas de pensamento crítico e a comunicação com *stakeholders*. Trabalhos futuros devem implementar uma rigorosa higiene de dados para garantir que a base de conhecimento seja confiável e validar essas descobertas com uma amostra maior e mais diversificada de participantes.

Palavras Chave

Elicitação de requisitos; Modelo de linguagem de grande escala; Geração reforçada por recuperação; Engenharia de instruções; Automação.

Contents

List of Figures

List of Tables

Listings

Acronyms

ADO	Azure DevOps
AI	Artificial intelligence
BPMN	Business Process Model and Notation
DSR	Design Science Research
ERD	Entity Relationship diagram
ETM	Elicitation topic map
GDPR	General Data Protection Regulation
GPT	Generative Pre-trained Transformer
HITL	Human-in-the-Loop
LLM	Large language model
NLP	Natural language processing
RAG	Retrieval-Augmented Generation
RE	Requirements engineering
SDLC	Software development lifecycle
SLR	Systematic literature review
SME	Subject matter expert
SUS	System Usability Scale
UML	Unified Modeling Language

1

Introduction

Contents

Every successful system is built around the needs of its stakeholders and should be highly responsive to those needs. The concept of requirements engineering is central to the creation of any system, as it is based on this area that we understand what the system should do, how it should behave, and what characteristics it should have to satisfy the needs of stakeholders. It involves analysing the need for a system, subsystem, or component and creating a set of requirements based on that analysis [1].

According to Peng, most of the progress in software engineering over the past few decades has been in dealing with accidental difficulties [2], stemming from the processes and tools employed in software engineering, while fundamental difficulties (mainly consisting of requirements and design) have seen little advancement [3].

When it comes to requirements engineering, multiple problems can occur in this phase of the Software development lifecycle (SDLC), such as incomplete or ambiguous requirements, poor documentation, and changes in the scope of requirements, among others [4], which can negatively affect the project's delivery time and/or the resources spent on it, so chatbots powered by Artificial intelligence (AI) appear as a possible solution to overcome some of these challenges.

Improving the communication between developers and stakeholders is crucial in ensuring the effectiveness and quality of the SDLC [5]. In software engineering, this communication is primarily initiated through requirements elicitation and documentation organisation. Therefore, it is essential to address this issue to enhance the efficiency and quality of the software development process.

Traditional requirements engineering methods often suffer from communication complexities, uncertainty in early stages, inadequate automation support, and inefficiencies that can result in project delays and resource overruns.

With the recent proliferation of technologies based on artificial intelligence, notably chatbots such as ChatGPT [6], Gemini [7], Microsoft Copilot [8], among others, they have been used to tackle numerous problems to obtain more efficient and innovative solutions. Therefore, this work aims to use these new technologies to improve the practice of requirements elicitation and documentation in software development projects.

1.1 Research Methodology

The methodology to be adopted is the Design Science Research (DSR) methodology. This approach is widely used in information science, aiming to create and validate prescriptive knowledge through the development of artefacts. DSR is a research paradigm that distinguishes itself from explanatory science by focusing on how things ought to be rather than how they currently exist. Unlike natural sciences, which explain and predict, DSR aims to

create practical solutions for real-world problems [9].

The entry point from Peffers et al. that was considered most appropriate for tackling the problem was the Objective Centered Solution, since the goal is to understand how using a better tool for eliciting requirements would improve this process, thus following a more robust sequence of activities [10]. The essential components of DSR consist of:

Key activities in DSR

- Problem explication: Involves investigating and analysing a practical problem.
- Requirements definition: Defining the requirements that a solution to the problem must meet.
- Design and development: Designing and developing an artefact (a construct, model, method, or instantiation) that meets the defined requirements.
- Demonstration: Demonstrating that the developed artefact can solve the problem.
- Evaluation: Finally, evaluating the effectiveness and efficiency of the artefact in solving the problem.

Systematic Literature Review Systematic literature reviews (SLRs) differ from traditional literature reviews because they aim to identify all published studies that address a specific question, and their methodology has been developed to minimise the effect of selection, publication, and data extraction bias. Reducing bias in systematic reviews is possible by creating a clear protocol, using sensitive search strategies, assessing studies for inclusion independently, and extracting and appraising data using standardised forms [11].

Kitchenham presents an overview of an eight-step guide to conducting an SLR, from defining the purpose and protocol of the review to searching, screening, appraising, extracting, synthesising, and writing the review [12].

1.2 Document Structure

The remainder of this document is organized in the following order:

Chapter 2 provides some theoretical background of the topics discussed throughout the document, such as requirement engineering, large language models, and the role of prompts.

Chapter 3 describes the conduct of the systematic literature review performed, the discussion of the research questions, and the findings. Additionally, it includes the initial interviews with analysts from the author's current project to better understand their workflows, challenges, and desired features for an AI solution.

Chapter 4 is dedicated to creating a general architectural framework for the proposed solution, the research of the platform where the prototype was developed, and the actual development of the prototype, as well as design considerations.

Chapter 5 presents the demonstration of the prototype demo and the results from the final interview with analysts.

Chapter 6 describes the assessment of the results from the analysts' final interview, and finally, chapter 7 presents the conclusion of this research, highlighting the key findings, the limitations encountered, contributions to the field, and future work.

2

Theoretical Background

Contents

In the following chapter, a brief theoretical introduction is provided about the concepts discussed during this research.

2.1 Requirements Engineering

According to Tikayat Ray et al., a requirement is defined as "a statement that identifies a system, product, or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability" [1].

Requirements engineering (RE) is regarded as probably the most critical task in the software development process, but often organizations and project teams overlook or do not understand the significance of RE and its impact on project success [4], some reasons for that may be time, budget and resource constraints, uncertainty and ambiguity in early stages, communication issues, inadequate tools and automation support, among other challenges [4]. Tominc et al. also state that lack of domain understanding is another challenge associated with requirements elicitation [13].

According to Khan et al., software design and requirements are among the most popular topics in software development Q&A repositories [14].

The process of eliciting requirements is systematic, yet flexible, and may require multiple sessions as stakeholder needs change and new information becomes available. This process is also highly collaborative, requiring ongoing feedback and validation from various stakeholders to ensure that everyone is on the same page; thus, clarity and alignment are key components of successful requirements elicitation [4].

2.2 Large Language Models

When we talk about Large language models (LLMs), we include Generative Pre-trained Transformer (GPT). While GPT is one of the LLMs, LLM refers to any language model that's used for Natural language processing (NLP) tasks [15].

The significant advancement of large language models that we've been witnessing over the last few years is causing a paradigm shift in academia, as well as in multiple other domains [16], making them a focal point of research due to the enormous benefits that can be derived from them. They have the potential to revolutionise qualitative data analysis by mimicking human-generated content and their access to vast amounts of data [17], such as Wikipedia and Book Corpus [18].

Pre-trained LLMs from entities like Google, Meta, and OpenAI[1] are cost-effective compared to training these models from scratch, due to the extensive computational resources required. These pre-trained LLMs are available for free on the Hugging Face[2] platform via the transformers library [1]. LLMs may not always be the best solution for companies; in such cases, utilising NLP techniques may be more appropriate due to their smaller scale. Figure 2.1 represents how LLMs relate to the technologies surrounding them.

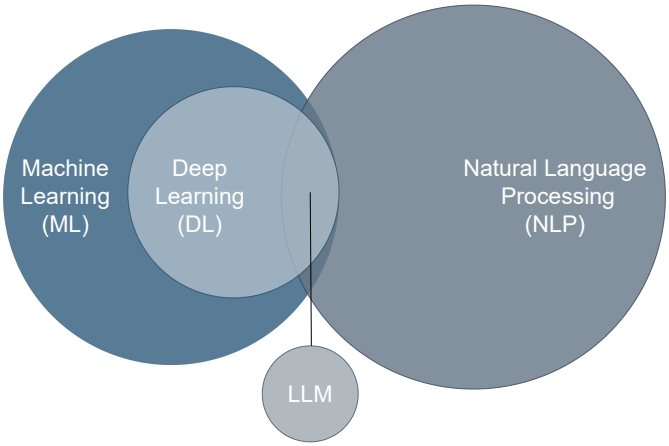


Figure 2.1: LLMs relation to other technologies, adapted from Genetec [23]

2.2.1 Role of Prompts

LLMs depend heavily on comprehensive prompts and the availability of contextual information to generate meaningful output. Slightly different prompts can produce very different outputs. A thorough empirical evaluation of prompt engineering is necessary for employing LLM agents [4]. LLMs such as OpenAI's GPT, Google's BERT, and LaMDA learn to comprehend and generate human language, or natural language, by predicting the most probable next word in a given sequence [4]. One way to use prompts is to ask an LLM to provide information or generate code. Another use of prompts is to dictate rules for the LLM to follow going forward, such as conforming to specific coding standards [24].

Prompt engineering refers to designing and testing prompts to enhance the performance of AI and achieve the desired output quality. As such, prompt engineers utilise their language expertise and knowledge of the task to create effective prompts enabling AI to generate the desired output. Prompt patterns refer to different templates targeted at specific goals, such as an output customisation pattern that focuses on tailoring the format or structure of the output by AI. It is recommended to format prompts consistently in the "Context, Task, and Expected Output" [4].

[1]LLM Sources: [19], [20], [21]
[2]Hugging Face website: [22]

Some of the well-known prompting techniques include zero-shot prompting, few-shot prompting [4], chain-of-thought prompting, tree of tree-of-thought prompting [25], among others.

2.2.2 LLMs with Requirement Elicitation

Regarding areas of LLM usage in software engineering, requirement elicitation, and system design and specification have received insufficient attention so far [24]. White et al. describe that a key to leveraging these LLM capabilities is to codify an effective catalogue of prompts and guidance on how to combine them at different stages of the software life-cycle to improve the whole process of software engineering [26].

Peng states that software engineering techniques, such as requirements, design, and validation, have the potential to rejuvenate in the era of LLMs, but this may require careful consideration of how to integrate them seamlessly with data-driven LLM technologies [3].

2.2.3 Motivation

AI-powered chatbots in software engineering can streamline processes, reduce manual effort, and improve efficiency. They can generate accurate and up-to-date documentation, foster better understanding between developers and clients, integrate with existing tools, and enhance the software development experience. Marques et al. state that implementing these technologies early in the project, such as during the requirements elicitation phase, may help reduce costs and ensure timely project delivery, as the success of this phase will significantly impact the rest of the software development life cycle [27].

3

Research Problem

Contents

This chapter formally defines the research problem by examining the shortcomings of current RE practices, particularly in elicitation and documentation within complex enterprise contexts. It analyzes the challenges that hinder the efficiency and quality of the RE process, focusing on usability, trust, and integration constraints that limit the adoption and value of proposed solutions.

It begins by formulating research questions that guide the artifact's design and evaluation. These questions are informed and refined through a systematic literature review, which consolidates gaps related to capabilities, risks/challenges, and validation quality. To ensure practical relevance, the problem is then further grounded through stakeholder interviews that surface real pain points, desired capabilities, and operational constraints from an ongoing project in the Dutch pension sector.

3.1 Systematic Literature Review

This research examines the ability of AI-powered chatbots to elicit requirements in software engineering projects, as well as their overall impact on the SDLC.

3.1.1 Planning

This segment consists of the initial stage of the SLR process. The study begins by clarifying the motivation behind it, followed by the goals and associated research questions that will be addressed during the research. Subsequently, a review protocol is introduced.

Research questions Based on the purpose of this research, the selected research questions are as follows:

RQ1 - What is the impact of AI chatbots on Software Development Stages? This question aims to understand the literature's perspective on the impact of utilizing artificial intelligence-powered chatbots in software development stages and tasks, whether positive or negative.

RQ2 - How are AI-powered chatbots being used to elicit requirements from stakeholders in software engineering projects? And with this question, we'll explore the effectiveness of these chatbots in eliciting requirements of sufficient quality.

RQ2.1 - How to automate requirements documentation with AI Chatbots? Another question that is interesting to explore in the context of AI-powered chatbots and that will be

considered a subtopic of RQ2 is their ability to automate requirements documentation in the context of a software engineering project.

Review Protocol To obtain all the literature resources present in this SLR, the accessed database was the "EBSCO Discovery Service" [28]. Specifically, the search string used was the following:

```
("AI-powered" OR "gpt-3.5" OR "gpt-4" OR "chatgpt"
OR "generative AI" OR "AI chatbot" OR "LLM*"
OR "prompt engineering")
AND ("requirements gathering" OR "software requirements" OR
"requirements engineering" OR "requirements"
OR "requirements analysis" OR "functional requirements")
AND ("SDLC" OR "software development")
```

Since the area related to large language models is still relatively new, the goal of this search string was to capture as many papers as possible that mention these technologies in the context of requirements engineering in software development projects. Table 3.1 contains the inclusion and exclusion criteria used in this SLR.

Inclusion Criteria	**Exclusion Criteria**
Written in English	Not written in English
Discusses AI	Is not peer-reviewed
Discusses Requirements Engineering	

Table 3.1: Criteria for paper selection

3.1.2 Conducting

Figure 3.1 contains the records identified in the SLR.

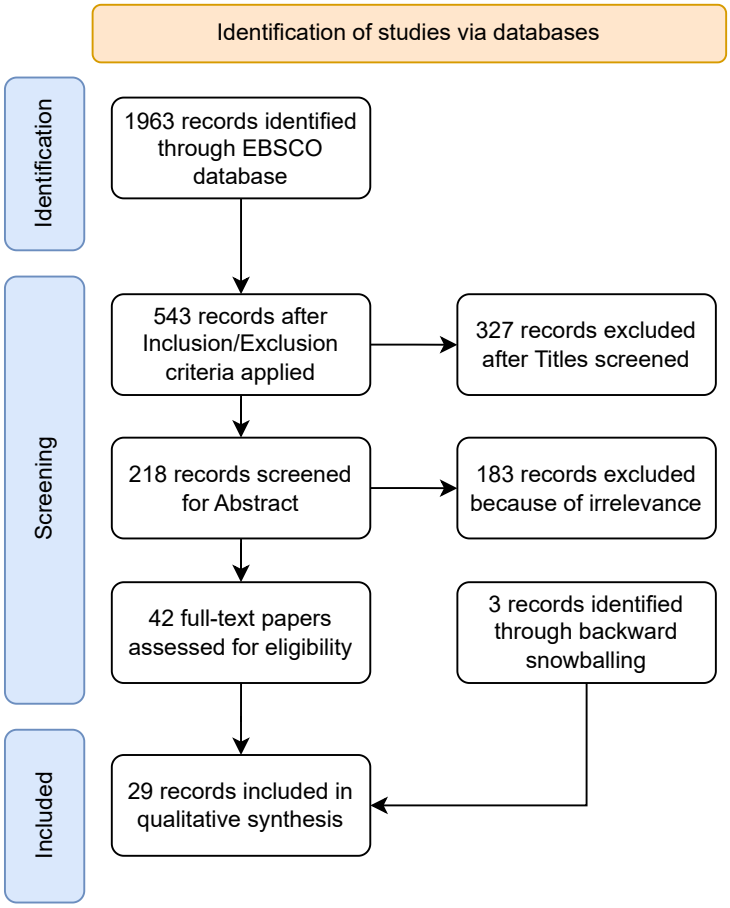


Figure 3.1: PRISMA flow diagram of the SLR conducted

After noticing a lack of relevant papers when conducting the SLR, three papers were identified through backward or reverse snowballing, which is a technique that looks back through the article's references to gain a better understanding of how the knowledge has evolved [29].

Figure 3.2 showcases the distribution of studies by publication year. These numbers indicate that the vast majority of the relevant research on this topic has been published in the last two years, correlating with the widespread adoption of LLMs.

3.1.3 Research Questions

In the following section, we discuss the literature review related to the research questions.

RQ1 - What is the impact of AI chatbots on Software Development Stages?

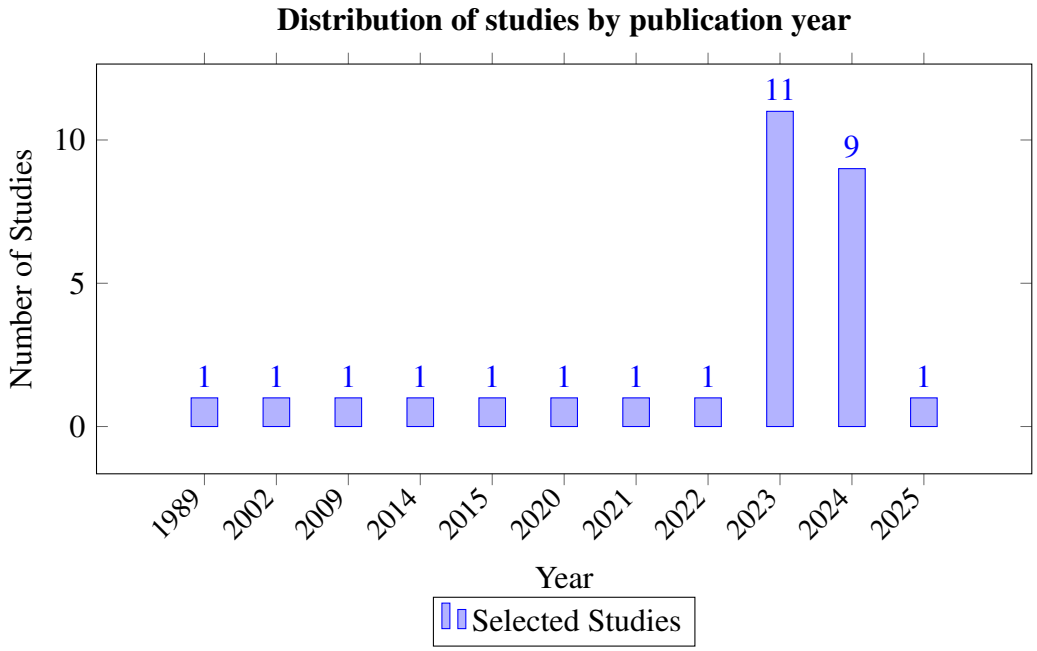


Figure 3.2: The number of selected studies from the literature review, categorized by their year of publication

3.1.3.A Capabilities

To have a better understanding of the capabilities of the current LLM-powered chatbots and artificial intelligence chatbots in general, after an analysis of the research papers, the mentioned capabilities are present in Table 3.2:

The most frequently cited and perhaps most widely recognized capabilities of AI-powered chatbots in the software engineering literature revolve around their capacity to generate and analyze content. These foundational abilities, which include the creation of code, the automation of documentation, and the summarization of complex data, represent the most immediate and tangible applications of LLMs in the SDLC.

Mingxing Liu et al. state that LLMs have strong potential in **code generation**, having performed extremely well on popular code completion benchmarks, and that with appropriate prompt engineering, they can generate practical safety-critical software [24].

Hassan et al. describe **sentiment analysis** as a crucial task to identify subjective information from a bulk amount of data [35]. Nakata et al. state that LLMs can be highly effective in capturing the details of user needs [39].

Sonkin and Tudose report that **automation**, such as documentation updates, can be seamlessly added to existing workflows [40].

Data has shown that as businesses strive to remain innovative and competitive, the adoption of AI-driven solutions may increase levels of collaboration and improve decision-making processes, ultimately enhancing project results [31].

Capabilities	Sources
Code generation	[3], [16], [18], [24], [26], [30], [31], [32]
Documentation generation	[3], [4], [18], [30], [31], [32], [33], [34], [35]
Data summarizing/analysis	[4], [17], [18], [26], [30], [31], [36], [37]
Defect detection and fixing	[3], [18], [26], [30], [31], [32], [38]
Pattern identification	[1], [4], [16], [17], [26], [30], [31]
Requirements analysis	[1], [3], [4], [17], [24], [26]
Code testing	[3], [18], [24], [30], [31], [32], [38]
Requirements generation	[1], [3], [4], [26], [34], [39]
Process large amounts of data	[4], [16], [17], [24], [30]
Code improvement	[3], [18], [26], [30], [32]
Requirements classification	[1], [4], [24], [26], [35]
Sentiment analysis	[4], [17], [30], [35], [36], [37]
Automate programming tasks	[16], [18], [26], [31]
Tool integration	[3], [4]
Improve processes	[18], [30]
Specification generation	[16], [26]
Pair programming	[30], [31]
Risk identification	[4]
Compliance analysis	[4]
Legacy code translation	[16]
Generalizability	[16]
Customer support	[30]
Cybersecurity	[30]
Translating technical jargon	[4]
Vulnerability detection	[32]
Test data generation	[17]
Persona generation	[17]

Table 3.2: Mentioned capabilities of AI-powered chatbots

Tominc et al. state that improvements must be introduced in all aspects of the company, from business processes and products to the company as a whole. For a company to remain competitive, it must seek best practices, adapt them to its own needs, and implement them [13]. By integrating AI, a company's ability to automate, organize, streamline, and analyze critical data sets is significantly improved.

3.1.3.B Challenges

The same exercise was performed to encounter the challenges mentioned throughout the research papers in what relates to LLM chatbots as well, which are present in table 3.3:

Several challenges have been identified that can significantly impact the SDLC. In the following section, an analysis will be conducted on the most frequently mentioned challenges.

Challenges	Sources
Need for human input	[1], [3], [4], [17], [24], [26], [30], [31], [37], [38], [41]
Scale and complexity	[3], [4], [24], [26], [30], [32], [35], [37], [38]
Hallucinations	[4], [16], [17], [26], [30], [32], [36], [37]
Underlying biases	[4], [16], [18], [30], [31], [36], [38]
Data Security	[3], [4], [26], [30], [32], [36], [39]
Critical thinking	[3], [4], [16], [24], [31], [38]
Reproducibility (model drift)	[4], [16], [24], [38], [39], [41]
Lacks nuance	[3], [4], [16], [17], [30], [38]
Ethical Concerns	[4], [16], [17], [18], [30]
Data privacy	[4], [16], [32], [36], [39]
Quality control	[3], [16], [24], [26]
Unfairness	[17], [31], [38]
Overreliance (automation bias)	[4], [16], [38]
Intellectual property concerns	[4], [16], [30]
Generative question answering	[24], [36]
General data generation	[36]
Lack of context of related work	[16]
Integration	[31]
Interpretability	[32]

Table 3.3: Mentioned challenges of AI-powered chatbots

Hallucinations Regarding hallucinations, which are instances when the model generates content that is not grounded in its training data [17], and the creation of misleading content, Mohan states that even though ChatGPT can construct convincing sounding scientific arguments, the result may be a mixture of facts and false information, which raises the question about how accurate and reliable these LLMs are in environments such as academia that require a high degree of truth. [36]. Javaid et al. also share concerns about ChatGPT's potential use in disseminating misleading material and disinformation due to its ability to generate plausible-sounding language with minimal expertise [30].

To mitigate this issue, Bano et al. state that users must scrutinize, verify, and meticulously interpret the outputs from these LLMs, ensuring that they maintain integrity and that the responses are aligned with the context [16].

Model drift This challenge relates to the constant updates of LLMs over time, where even when provided with exhaustive details about methodology, datasets, prompts, parameters, and versions used, it does not guarantee that the same results and analysis can be reproduced in the future [16]. In other words, the correlation between inputs and outputs changes over time [38]. Gerosa et al. also denote that even the smallest changes in the prompts used can alter the nature of the response, not only affecting the information itself but also having an impact on the structure, style, and diction of the text output [17].

Underlying biases Many authors state LLMs have some inherent issues, such as systematic inaccuracies and stereotypes in the output [4]. Since LLMs rely on techniques developed using data, we must be cautious about their accuracy, as the training data may be biased or contain flaws [30]. Mohan states that bias is transferred to models during the feature selection phase when they are created [36]. According to Bano et al., due to the possibility of inherent biases in LLMs, using these models for qualitative research can potentially misrepresent the analysis and conclusions drawn from their use. [16]. In sum, the capacity of these models to understand different scenarios can be flawed towards certain demographics or subject areas [30].

Scale and memory problems One of the biggest challenges related to LLMs is their limited context length, which causes them to regularly lose context of certain tasks during a specific session and also makes it hard for them to process large documents [4].

According to Peng, when it comes to developing applications of reduced complexity, with proper guidance, ChatGPT can gradually generate fully executable programs with natural language that have a good standard of code quality. To do so, however, developers need strong communication skills to express the application's requirements clearly and have the ability to define clear steps to decompose development tasks. This will require deep knowledge of the software design and the whole implementation process, but even with small-scale applications, ChatGPT might forget code that was previously generated [3].

Mingxing Liu et al. suggest that when dealing with more complex applications, it's a good idea to continually reintroduce requirements to ChatGPT to ensure that they aren't overlooked [24]. With more recent models, such as GPT-4, the token limit has increased, allowing for a longer context, which enables the sending of more extensive and specific prompts, as well as an expanded ability to process large amounts of data [32]. According to White et al., some patterns can be explored to help alleviate issues with LLM token limits when generating code [26].

Peng also argues that LLMs should not be a solution just for the initial code generation, but also for the whole lifecycle of a software system, including the iterative requirement gathering phase, but to achieve that, the model needs to have context about the entire system, which at the moment is not feasible [3].

Human input and critical thinking The most significant challenge encountered during the SLR was the need for human input when interacting with LLM-powered chatbots. According to Peng, principles such as modularity, information hiding, separation of concerns, and code comprehensibility are essential in software engineering. Current LLMs aren't able to fully grasp them; that is why even those who view the prospect of code generated by

LLMs very optimistically still acknowledge that it requires human review [3].

Arora et al. state that when it comes to working at a certain level of abstraction, it becomes extremely difficult for LLMs, which means that more manual input is required from stakeholders in these cases [4].

Gerosa et al. agree that current LLMs lack the nuanced and empathetic understanding that is common to humans, and conclude that an integrated approach where both AI and human-generated data coexist will likely yield the most effective outcomes [17]. Motger adds that from the practitioner's perspective, full automation is not only impossible, but also highly discouraged [37].

Peng also states that even though LLMs are trained on large amounts of internet data, including code and professional materials, there's a lot of tacit knowledge present in software engineering teams that might not be written or recorded anywhere and that is challenging for LLMs to capture, in part because of the availability of said information, and in another because of the level of abstraction or ambiguity that the information might have [3]. Arora et al. also corroborate this statement, specifically in the realm of requirements elicitation and business context [4].

Arora et al. also argue that for non-functional requirements, such as determining how secure or scalable a system needs to be, human expertise, in the form of domain experts, is required. This also extends to compliance issues [4].

Requirements elicitation problems According to Burnay et al., one of the main challenges in requirements elicitation is knowing how to extract the correct information from stakeholders and whether they are sharing all the information that could be useful for the system design. To address this issue, C. Burnay et al. present the Elicitation topic map (ETM) [42], which is essentially a tool designed to help engineers prepare elicitation interviews with stakeholders, highlighting topics that may be discussed during interviews and the likelihood of stakeholders raising them.

Security and privacy problems Arora et al. state that since requirements are by their very nature essential for software engineering and contain much sensitive information, their disclosure through public LLMs may result in intellectual property loss, security breaches in deployed systems, organizational and personal privacy loss, among other concerns [4].

Automation bias Automation bias, also known as complacency bias, occurs when humans begin to display unwarranted trust in AI solutions [4]. One possible consequence is that humans start to trust AI systems excessively, leading to a degradation in the quality of decisions made by humans [38].

RQ2 - How are AI-powered chatbots being used to elicit requirements from stakeholders in software engineering projects? After an analysis of the papers was conducted, data was extracted regarding their mention of AI technologies, requirements engineering, and whether AI was mentioned in the SDLC. The findings are shown in Table 3.5.

Paper title	Mentions AI	Mentions RE	Applies AI in SDLC
User stories collection via interactive chatbot to support requirements gathering [43]	No	Yes	No
A study of AI-based techniques for requirement analysis in software engineering [33]	Yes	Yes	Yes
The Chatbot Usability Scale: the Design and Pilot of a Usability Scale for Interaction with AI-Based Conversational Agents [44]	No	No	No
A study on ChatGPT for Industry 4.0: Background, potentials, challenges, and eventualities [30]	Yes	Yes	Yes
The Power of Generative AI: A Review of Requirements, Models, Input–Output Formats, Evaluation Metrics, and Challenges [18]	Yes	Yes	Yes
What stakeholders will or will not say: A theoretical and empirical study of topic importance in Requirements Engineering elicitation interviews [42]	No	Yes	No
Software development in the age of intelligence: embracing large language models with the right approach [3]	Yes	Yes	Yes
Understanding the human context in requirements elicitation [45]	No	Yes	No
Agile Methodology for the Standardization of Engineering Requirements Using Large Language Models [1]	Yes	Yes	Yes

Continued on next page...

Paper title	Mentions AI	Mentions RE	Applies AI in SDLC
The Requirements Apprentice: Automated Assistance for Requirements Acquisition [46]	Yes	Yes	No
Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs [4]	Yes	Yes	Yes
An Empirical Study of the Code Generation of Safety-Critical Software Using LLMs [24]	Yes	Yes	Yes
ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design [26]	Yes	Yes	Yes
Techniques to automatically generate Entity Relationship Diagrams [34]	Yes	Yes	No
An ontology-based approach to engineering ethicality requirements [47]	Yes	Yes	No
Computer-Aided Software Development Process Design [48]	Yes	Yes	No
The Potential of AI-Driven Assistants in Scaled Agile Software Development [31]	Yes	Yes	Yes
A New Approach to Web Application Security: Utilizing GPT Language Models for Source Code Inspection [32]	Yes	No	Yes
Management Consulting in the Artificial Intelligence – LLM Era [36]	Yes	Yes	No
Artificial Intelligence and Software Testing : Building Systems You Can Trust [38]	Yes	Yes	No
Insights into software development approaches: mining Q&A repositories [14]	No	Yes	No
Can AI Serve as a Substitute for Human Subjects in Software Engineering Research? [17]	Yes	Yes	Yes

Continued on next page...

Paper title	Mentions AI	Mentions RE	Applies AI in SDLC
Artificial Intelligence and Agility-Based Model for Successful Project Implementation and Company Competitiveness [13]	Yes	No	No
Large Language Models for Qualitative Research in Software Engineering: Exploring Opportunities and Challenges [16]	Yes	Yes	Yes
Natural Language Processing Methods for Document-based Requirements Specification and Validation Tasks [37]	Yes	Yes	Yes
Needs Companion: A Novel Approach to Continuous User Needs Sensing Using Virtual Agents and Large Language Models [39]	Yes	Yes	Yes
A systematic mapping to investigate the application of machine learning techniques in requirement engineering activities [35]	Yes	Yes	Yes
The Current Challenges of Software Engineering in the Era of Large Language Models [41]	Yes	Yes	Yes
Beyond Snippet Assistance: A Workflow-Centric Framework for End-to-End AI-Driven Code Generation [40]	Yes	Yes	Yes

Table 3.5: Papers analysed for requirements engineering in the software development life cycle

Effective requirements elicitation requires constant feedback and communication among stakeholders, as their needs may vary significantly, and there may be important topics that need to be further clarified to enhance the requirements, that they might be withholding [42].

Budake et al. conclude that their method, which utilizes NLP techniques to elicit requirements, can enhance the quality, efficiency, and accuracy of requirement analysis, and suggest some future directions for further research [33].

Gerosa et al. propose an interesting strategy to tackle some of these challenges, consisting of persona-based prompting, in which strategic prompting enables an LLM to potentially simulate a particular demographic profile [17]. In persona-based prompting, the prompts

are designed to elicit responses that reflect the behaviors, viewpoints, and characteristics of particular fictional personas.

Researchers can conduct virtual interviews with AI-generated personae to anticipate user needs, refine requirements, or foresee challenges in adoption. This helps enrich the qualitative data that informs software design and user modelling [17].

This is also useful in cases where recruiting real participants is extremely difficult. Even if LLMs don't provide real data and have their own limitations, specific findings generated from them can still be useful [16].

According to Mingxing Liu et al., for complex requirements, the LLMs may be unable to generate code that satisfies all requirements simultaneously. To address this issue, iterative reinforcement can be performed based on the review and analysis of the generated code. This iterative process can focus on domain requirements, specification constraints, or other relevant factors. After several iterations, we can obtain higher-quality software code that meets the intended requirements [24].

RQ2.1 - How can Requirements Documentation be Automated with AI Chatbots? An important area of innovation in RE with AI-powered chatbots involves moving beyond manual processes, especially regarding documentation. The following section explores what the literature contains about this topic.

Javaid et al. describe using ChatGPT to generate documentation templates that can return, for instance, programming function documentation based on its parameters, return values, and the function's goal, as well as searching for best practices regarding any given issue or technology [30]. It's also crucial for tasks such as requirements classification, particularly with large-scale systems where there could be enormous amounts of requirements to define [1].

Budake et al. found that while using NLP, there are ways to automatically generate documentation regarding entities, attributes, and their types, relationships, and the cardinalities of the relationships, create an Entity Relationship diagram (ERD) [34], and make it possible to run SQL queries for natural language queries for a given scenario. It's also possible to generate Unified Modeling Language (UML) diagrams based on user requirements. This could have numerous benefits, such as reducing design time, cost, and error, and enhancing software quality [33].

3.2 First Interviews

To complement the findings of the SLR, this research incorporated empirical data from professional practice. Semi-structured interviews were conducted with business analysts from

the author's current project to gather firsthand insights into their real-world workflows, challenges, and operational constraints. The primary goal of this engagement was to validate and enrich the problem analysis, ensuring that the challenges identified in the literature accurately reflect the lived experiences of practitioners in the field.

3.2.1 Project Context

The project in which the interviews were conducted involves a Dutch pension provider specializing in providing comprehensive pension administration, actuarial and legal advice, and tailored communication solutions, all while managing complex pension schemes and adapting to evolving regulations. Currently, it is guiding its clients through the transition to a new national pension system, with the primary objective of ensuring a smooth and high-quality migration to the updated framework.

The team in which we operate is subdivided into workstreams. Each workstream focuses on a particular area of the project and operates semi-independently. The workstreams include developers who focus on software development, analysts responsible for creating user stories and gathering requirements, and testers who ensure the quality and functionality of the software.

The Dutch pension sector is currently undergoing significant legislative reforms aimed at modernising the pension system to align with longer life expectancies, changing job market dynamics, and the need for greater flexibility. The 2019 Pension Agreement, now codified in the new Pension Act, introduces personal pension accounts, more transparent accrual and payout mechanisms, and a shift from defined benefit to defined contribution models. These reforms are designed to ensure sustainable pension funding, more transparent communication of pension rights, and a flexibility that's more compatible with contemporary career patterns [49].

Within this evolving legal and economic landscape, the Dutch pension law sector presents itself as both a complex regulatory environment and a dynamic field for innovation in requirements elicitation and software tools, especially since pensions contain sensitive personal and financial data governed by strict standards.

3.2.2 Interview Protocol

At the beginning of each interview, participants provided explicit consent to be recorded during the session. Participants were informed that recordings would be used solely for transcription and data extraction purposes and would be permanently deleted afterwards. The purpose of the research was clearly explained, with emphasis on the fact that all collected data would be anonymised to ensure confidentiality. It was also clarified that participants

could stop the interview at any time or decline to answer any questions that made them feel uncomfortable.

The interviews were semi-structured, combining a set of prepared guiding questions with the flexibility to explore new topics that emerged during the conversation, and typically lasted 30 to 45 minutes.

The discussion examined participants' current practices and challenges in requirements elicitation and documentation, as well as their views on how AI-driven chatbot solutions could enhance these processes. It was structured around three themes: desirable features and functionality of such a chatbot, its potential integration into daily workflows and existing platforms, and ways it could address existing challenges.

The complete questionnaire can be found in Chapter B, Section B.1.

3.2.3 Participant Profiles

Table 3.6 summarises the participants' professional backgrounds, tenure at the company, and prior IT training experience.

Participant	Years within role	Role	Background	IT Traineeship
A1	1	Business Analyst	Technology	Yes
A2	2.5	Business Analyst	Hospitality	Yes
A3	2	Business Analyst	Biomedicine	Yes
A4	2	Business Analyst	Physics	No
A5	2	Business Analyst	Physics	Yes

Table 3.6: Interview participant profiles

All participants hold the role of Business Analyst, but they bring diverse academic and professional backgrounds, ranging from technology to biomedicine, hospitality, and physics. Four of the five participants have completed an IT traineeship, which provided them with foundational technical and analytical skills before starting their current roles. Their tenure at the company ranges from one to two and a half years, offering perspectives from both relatively new and more experienced employees.

This diversity of backgrounds and levels of IT training is particularly valuable as it reflects a mix of domain expertise and technical familiarity. Although the sample did not include participants with extensive experience, the diversity in their levels of expertise provided complementary perspectives, contributing to more valuable insights and helping ensure that the findings reflect a variety of approaches and challenges in the requirement elicitation domain.

3.2.4 Results

The interview analysis revealed a conversation-driven approach as the dominant working methodology among analysts. Rather than relying on existing documentation, analysts prioritise direct stakeholder conversations, typically beginning with materials such as Excel sheets with acceptance criteria or client Word documents, but requiring verification and clarification through meetings and discussions.

A hierarchical information gathering pattern emerged across all interviews, where analysts follow a structured approach starting from high-level features and drilling down through overview stories to detailed implementation requirements. This process involves examining Azure DevOps (ADO) ticket relationships, API documentation, and system interconnections to build an understanding of business processes.

A dedicated documentation repository exists (Confluence); however, it is not commonly used, as the necessary information is sometimes challenging to find, and many processes remain poorly documented.

Pain Points and Challenges The interviews with business analysts revealed a range of persistent pain points and challenges in the requirements elicitation process. Table 3.7 summarizes the key pain points identified by the participants during the interviews.

Table 3.7: Pain points identified by the analysts

Pain points	**A1**	**A2**	**A3**	**A4**	**A5**
Outdated/Incorrect documentation		X		X	
Unclear stakeholder requirements			X		X
Knowledge dispersion	X	X	X	X	X
Repetitive tasks	X	X	X		
AI trust concerns			X	X	X
Complexity and ambiguity in requirements	X		X		

Knowledge dispersion emerged as the main challenge in the findings. Analysts reported that a significant amount of tacit knowledge was tied to Subject matter experts (SMEs), who are often very busy, creating a knowledge holder bottleneck. In addition, information was scattered across multiple sources, such as ADO tickets, wikis, Word documents, emails, and even stakeholders' personal computers, making it extremely time-consuming and frustrating to locate relevant information when needed.

Tied to this challenge is the documentation quality. Analysts reported issues with outdated and incorrect documentation, including examples such as acceptance criteria from a year and a half prior that had never been implemented and incomplete Confluence pages that hindered effective information retrieval, leading to wrong assumptions and rework. For an

AI-powered chatbot to be considered reliable, an analyst stressed the importance of feeding it information from trusted sources.

Some analysts expressed a zero tolerance for hallucinations when using AI for knowledge-related tasks, emphasizing that in a domain as regulated as pensions, incorrect information could have serious career implications. In contrast, others indicated a more balanced approach regarding AI-powered chatbots, specifying that they can be helpful for validation but should not replace personal verification and team checks.

Three analysts indicated frequent, repetitive tasks involving summarizing large documents, ensuring clarity and completeness of user stories, and producing documentation of process flows, which are activities that require substantial manual effort to filter relevant information and maintain quality.

An analyst described challenges in writing requirements, noting that they are often too vague, leading to confusion, or too detailed, risking the omission of edge cases. Analysts who are deeply familiar with the domain may also include too little context, assuming shared knowledge, or too much, rendering the stories difficult to read.

Another identified pain point is when stakeholders themselves do not precisely know what they want, or may not admit their uncertainty, leading to last-minute changes and miscommunication, or requirements that are refined over time.

The research also revealed variable time investment in user story creation, ranging from 15 minutes for simple configuration changes to several weeks for complex feature analysis. The majority of time is spent on information gathering and requirement validation rather than the actual writing process, suggesting that improved information access could significantly enhance productivity.

Desired Capabilities and Functional Requirements Based on the interview analysis, several capabilities were identified as useful for a future chatbot system:

- Domain-specific knowledge repository: All analysts expressed interest in a chatbot with deep pension and project-specific knowledge, explicitly mentioning the need for an ontology that understands project terminology and integration with legal and company documents.

- Advanced search and connection mapping: Analysts A1, A2 and A4 requested sophisticated search capabilities that can connect related information across different systems, with emphasis on connection graphs to visualise relationships between user stories from different time periods.

- Context-aware documentation generation: Analysts A1, A2, A3 and A4 requested automated assistance with acceptance criteria generation, user story templates, and docu-

mentation summaries, while emphasising the need for human oversight and validation of AI-generated content.

- Information validation and gap detection: Analyst A4 requested a system capability that could identify missing requirements or inconsistencies based on accumulated project knowledge, addressing the challenge of identifying unknown unknowns in requirements analysis.

3.2.5 Problem Statement

The SLR identified significant challenges in using AI-powered chatbots within requirements engineering, including limited practical validations of AI tools, and inherent AI limitations such as hallucinations, model drift, bias, constrained context handling, and the need for human input. These limitations hinder widespread adoption and effective use within complex real-world elicitation contexts.

Complementary interview findings revealed real-world pain points experienced by business analysts, such as knowledge dispersion across multiple sources, outdated or incorrect documentation, unclear and evolving stakeholder requirements, repetitive manual tasks, and concerns about trusting AI outputs without human oversight. Analysts prioritize direct stakeholder conversations and see AI as a helpful assistant for drafting and quality checks rather than a replacement for expert judgment.

This thesis aims to address these gaps by exploring how AI-powered chatbots can assist requirements elicitation and documentation by enhancing information accessibility, reducing manual effort, and supporting knowledge dissemination, while preserving the essential human supervision required to ensure accuracy and trust.

4

Design

Contents

According to Peng, the challenge of modern-day software engineering is not the availability of individual tools, but how to seamlessly integrate those tools and automation capabilities into an intelligent experience with the support of AI technology [3].

This chapter details the design of the AI-powered chatbot system. The structure of this chapter follows a top-down approach, beginning with a generic architectural framework derived directly from the challenges and requirements identified in the SLR and analyst interviews. Subsequently, it outlines the systematic process for selecting a technology platform capable of implementing this architecture. Finally, it presents the specific instantiation of the framework using the chosen tools to address the unique context of the Dutch pension project.

4.1 Generic Architectural Framework

The initial research phases, the SLR, and the interviews with business analysts revealed a set of core challenges and desired capabilities for any tool intended to enhance requirements engineering. Traditional methods suffer from knowledge dispersion, documentation inconsistencies, and significant manual effort. At the same time, while AI-powered chatbots present a promising solution, they introduce challenges related to hallucinations, data privacy, and the need for human oversight, impacting trust in AI-generated content.

To address these findings, a generic architectural framework is proposed. This framework is not tied to a specific technology but rather outlines the essential components required to build an effective, reliable, and trustworthy AI assistant for requirements elicitation.

The proposed architecture consists of four key layers:

Knowledge and reasoning core (RAG-based) To tackle the critical issue of AI "hallucinations", in which the model generates plausible but incorrect information, the architecture is centered around a Retrieval-Augmented Generation (RAG) system. RAG systems combine the power of LLMs with external knowledge bases by retrieving relevant information from a knowledge repository and augmenting the model's responses with this contextual information [50].

To build on the trust that analysts require, a provenance component is needed for the RAG system, which makes the chatbot's reasoning transparent. Its objective is to reveal the source of every piece of information the RAG system uses. When the system retrieves chunks from the knowledge base to generate an answer, this layer must package the metadata of those chunks, such as document title, source URL, page number, and last-updated date, and present it alongside the response.

Finally, a confidence scoring feature is needed to formalize uncertainty rather than leave it to the LLM's discretion. It evaluates RAG outputs before presentation, assigning a con-

fidence level (High, Medium, Low) based on factors such as the number of corroborating sources, chunk relevance, and source recency.

This approach directly addresses the analysts' "zero tolerance for hallucinations" by grounding the LLM in a curated, private, and transparent knowledge base, combining documentation such as requirements repositories, regulatory documents, and historical project data, which mitigates automation bias by signaling when human review is needed.

Instead of relying solely on its general training data, the system first retrieves relevant information from project-specific documentation. This design choice directly tackles the "knowledge dispersion" pain point identified by analysts by creating a single, queryable source of truth, aligning with their request for a domain-specific knowledge repository.

In this context, RAG represents a promising approach toward establishing a single source of truth, one of the key challenges in supporting requirements analysis. RAG can enhance the accuracy and consistency of requirements elicitation by implementing a structured mechanism for continuously supplying relevant information and integrating this knowledge within an AI-powered chatbot.

Figure 4.1 demonstrates what a basic RAG implementation looks like.

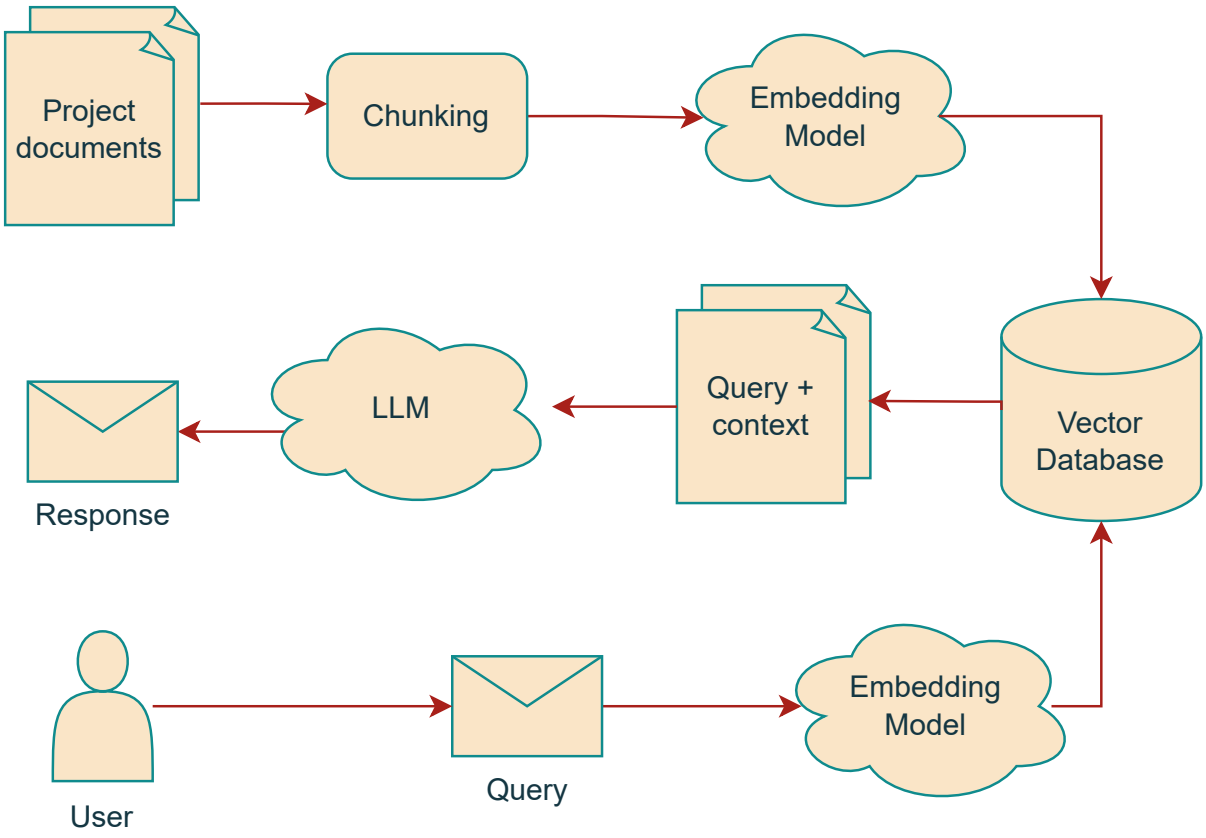


Figure 4.1: RAG diagram

Conversational agent and prompt management layer At the center of this layer is the conversational agent, the component that drives the system's intelligent behavior. In this context, an agent is a system designed to serve as a decision-making center. It leverages an LLM for reasoning, allowing it to understand user goals, interact with external tools, retrieve information, and autonomously determine the sequence of actions needed to fulfill a request.

The two core components that define this agent are the underlying LLM and a structured system prompt.

The SLR confirmed that the quality and design of prompts are crucial for steering the LLM's behavior and ensuring high-quality output. This system prompt is responsible for defining the agent's persona, its operational boundaries, and its response protocols, which tell the agent "who it is," "what it should do," and "how it should behave". By carefully crafting these system prompts, the architecture ensures the agent behaves as a specialized assistant, acknowledges its limitations, and adheres to strict information verification protocols, all of which were key concerns raised by the analysts regarding AI trust.

Tool and workflow integration layer To be truly useful, the AI assistant must integrate seamlessly into the analysts' existing workflows. The interviews clearly indicated a need for integration with enterprise tools like Jira and Confluence to generate user story templates and documentation summaries. Therefore, the architecture includes an integration layer with pre-built connectors and API extensibility.

This allows the chatbot to perform actions beyond conversation, such as creating tickets or publishing/retrieving documentation, thereby automating repetitive tasks identified as a key pain point by analysts. This aligns with the SLR's identification of "tool integration" as a key capability for enhancing the SDLC.

A critical part of this layer is the Human-in-the-Loop (HITL) validation component, in response to the interview findings, which emphasized that analysts remain cautious about fully trusting automated AI outputs, especially in a regulated domain where errors can have significant downstream consequences. This component acts as a checkpoint, enabling human experts to review, edit, and explicitly approve or reject chatbot outputs before they are committed to external systems like Jira or Confluence. It ensures that no automated content is finalized without expert oversight, supporting quality control, accountability, and building the user confidence required for the chatbot's adoption.

User interface layer The final layer provides the interface through which users interact with the system. To maximize adoption, this layer is designed to be abstract, allowing connection to various front-end platforms like Microsoft Teams, WhatsApp, and Telegram. This flexibility ensures the chatbot can be deployed within the communication tools analysts al-

ready use daily, reducing friction and avoiding the need for another standalone application, a principle reflected in the workflow-oriented feedback from the interviews.

Summary This framework is built on a modular design, integrating distinct components into a cohesive and flexible architecture. Such a structure allows for adaptability across different domains while attempting to address AI's known limitations in requirements engineering.

The architecture is guided by the core principles of reliability, transparency, and user control, positioning the system as a collaborative assistant that augments human expertise rather than a fully autonomous solution.

4.2 Technology Platform Selection

With the generic framework defined, the next step was to identify a technology platform capable of implementing its components. The selection process was guided by a set of criteria derived directly from the architectural requirements and practical constraints of the project.

4.2.1 Search Methodology and Evaluation Criteria

To identify potential platforms for developing the chatbot, a search was conducted on Google Search. The search strategy was designed to identify platforms that combine low-code development, for a lower learning curve, with both conversational AI and workflow automation capabilities.

Search String Used:

```
"low-code workflow chatbot automation tools artificial intelligence"
```

Search Date: 20 June 2025

Platform Selection Criteria To ensure objectivity and avoid commercial bias in the platform identification process, the following selection criteria were applied to the Google Search results:

1. Non-sponsored results only - Excluded all sponsored listings;
2. Direct tool websites - Selected only official platform websites, excluding comparison articles, blog posts, and third-party reviews;

3. Tools that could be used out of the box after registering, so platforms requiring complex sign-up or a contact-for-quote process were excluded to ensure accessibility and transparency.

Using the saturation method, the search for new tools was stopped after reaching a page where the above criteria weren't met, which happened at page 5.

Selected Platforms for Evaluation After applying the above-mentioned search string and applying the selection criteria, the following platforms were identified for evaluation:

1. n8n - https://n8n.io/
2. Flowise AI - https://flowiseai.com/
3. Latenode - https://latenode.com/
4. Mendix - https://www.mendix.com/
5. Voiceflow - https://www.voiceflow.com/
6. Power Platform - https://www.microsoft.com/pt-pt/power-platform
7. VectorShift - https://vectorshift.ai/
8. Dify.ai - https://dify.ai/
9. Zapier - https://zapier.com/

Evaluation Criteria To systematically assess the identified platforms, the following set of evaluation criteria was chosen, with weighted values prioritizing the most critical architectural components:

- **AI/LLM Capabilities (30%):** The highest priority was the platform's ability to implement the RAG-based core. This included support for multiple LLMs, flexible agent creation, and robust prompt engineering capabilities to address the core challenges of hallucinations and trust.
- **Integration (20%):** Reflecting the need for the Tool and Workflow Integration Layer, this criterion assessed the availability of pre-built connectors (especially for Jira and Confluence) and general API extensibility.
- **Workflow Automation (15%):** This focused on the platform's ability to build the complex, conditional logic required for a dynamic conversational agent.
- **Deployment (5%):** Crucially, this included the option for self-hosting. This is a direct response to the data security and privacy concerns identified in both the SLR and the highly regulated context of the Dutch pension sector.
- **Ease of Use (15%), and Licensing (15%):** These criteria addressed the practical constraints of developing a prototype within the scope of this thesis.

Table 4.1: Weighted scoring matrix for platform selection

Criteria (Weight)	Sub-criteria	n8n	Flowise AI	Dify.ai	Power Platform	Mendix	Latenode	Voiceflow	VectorShift	Zapier
AI/LLM Capabilities (30%)	LLM Support	4	5	5	5	3	5	3	5	2
	AI Agents	4	5	5	3	5	3	3	5	2
	Context Retention	5	5	5	3	3	3	3	5	2
Integration (20%)	Available Tools	5	4	4	5	5	5	4	4	5
	API Integration	5	5	5	5	5	5	4	4	4
	Database Connectors	5	5	4	5	5	4	4	4	4
Workflow automation (15%)	Conditional Logic	5	5	5	5	4	5	4	4	4
	Data Transformation	5	5	5	5	4	5	4	4	4
	Custom Code Extension	5	3	3	4	4	5	4	5	5
Ease of use (15%)	Documentation	5	4	4	5	5	4	5	4	5
	Community	5	4	4	5	4	4	5	4	5
	Learning Curve	3	4	4	4	3	3	4	4	4
Deployment (5%)	Self-Hosting	5	5	5	1	4	1	1	1	1
	Cloud Options	4	4	4	5	5	5	5	5	5
	Interface Options	5	5	3	5	4	3	5	3	3
Licensing (15%)	Operational Costs	2	3	3	3	1	3	3	4	4
Total weighted score		**12.1**	**12.2**	**11.9**	**11.5**	**10.7**	**10.9**	**10**	**11.7**	**9.5**

Scoring: 5 = Excellent, 3 = Good, 1 = Poor. Weights: AI/LLM capabilities = 30%, Integration = 20%, Workflow automation = 15%, Ease of use = 15%, Deployment = 5%, Licensing = 15%.

The value calculations for the identified platforms are presented in Table 4.1.

The weighted score table 4.1 research relied on web searches for information gathering. While efforts were made to ensure data accuracy, some sources may be incomplete or outdated. Therefore, the results presented here should be interpreted cautiously, as they may not fully reflect the latest or most precise platform evaluations.

4.2.2 Platform Evaluation Results

As a result of this analysis, Flowise AI emerged as the top choice and was selected as the primary development platform. Its selection was justified by a specialized focus on AI agent development and a user-friendly approach to building conversational AI applications.

Some key strengths that contributed to its high score included its dedicated AI/LLM integration with pre-configured components and a visual, drag-and-drop interface specifically designed for AI workflows. The platform also offered direct support for popular language models, including OpenAI, Gemini, and Claude. Its open-source nature also reinforced the decision, making it more secure and allowing for customization, and a smaller learning curve supported by an active community and numerous tutorials.

However, during the initial development phase, it became apparent that Flowise AI had significant limitations, namely:

- Limited component ecosystem: The platform's limited library of pre-built components for integrations required manual development effort, impacting the project timeline and feasibility.
- Documentation and troubleshooting: Implementation challenges arose during knowledge base configuration, particularly with embedding dimensions and setting up the vector store of the RAG system. The insufficient documentation and debugging mechanisms in Flowise hindered further development.
- Customization constraints: The platform's limited ability to extend functionality beyond pre-built components restricted the implementation of requirements elicitation features identified in the analyst interviews.
- Economic limitations: While the free plan initially appeared sufficient, the whole project would exceed the platform's free plan limits, necessitating an upgrade that wasn't economically viable.

Considering this, a reassessment of the platform was necessary, and n8n was chosen, the second platform with the highest score. To name a few advantages that justified this transition:

- More integrations: n8n provides over 400 pre-built integration nodes and database connectors for various data storage solutions, thereby reducing the need for custom development efforts.
- Development flexibility: The platform supports logic extension through both JavaScript and Python, provides version control capabilities, and includes debugging mechanisms with detailed error reporting, token usage tracking, and analysis of processing times.
- Documentation and community support: As a more established platform, n8n offers more documentation, a bigger community, and training materials, addressing the limitations encountered with Flowise AI.
- Self-hosting and interface: n8n's self-hosted deployment options provide complete data control while maintaining cost efficiency, addressing the security and privacy concerns inherent in Dutch pension sector applications. Additionally, the platform supports deployment across multiple interfaces, including Microsoft Teams, Telegram, WhatsApp, and others.

Following the change to n8n, no additional technical barriers were encountered, allowing for a continuous development focus on the core functional requirements identified through stakeholder research.

4.3 Framework Instantiation: Implementation with n8n

This section details the concrete implementation of the generic architecture using n8n for the Dutch pension project.

Azure OpenAI API keys were obtained through the employing organization to ensure an additional layer of privacy and confidentiality, aligning with the strict data security standards of the Dutch pension law sector.

4.3.1 Realizing the RAG Knowledge Base

The RAG system's knowledge core was implemented using n8n's built-in nodes, a process that began with curating the right data sources. Initially, analysts requested the inclusion of project tickets from ADO, as they contain a rich history of requirements and discussions. However, this was not feasible due to significant privacy concerns, as the tickets contained sensitive information that could not be exported.

As an alternative, and in consultation with the analysts, the knowledge base was populated with a combination of official online resources related to the Dutch pension sector

and selected documentation from the project's Confluence domain, which detailed specific process configurations and functional knowledge.

To build the knowledge base, these documents were fed into an n8n workflow. The first step involved a **Recursive Character Text Splitter** to break down the extensive documents into smaller, manageable chunks. These chunks were then converted into vector embeddings using Azure OpenAI's **text-embedding-3-large** model and subsequently stored in a **Pinecone vector database**. This vector store serves as the chatbot's specialized, private knowledge base.

To ensure the system remains current and avoids knowledge decay, a separate, automated workflow was implemented. This workflow runs daily to retrieve new or updated documentation from Confluence, process it through the same embedding pipeline, and insert it into the Pinecone database, ensuring the chatbot's information is consistently refreshed.

Two **critical limitations** emerged from this process. First, the vectorization process was restricted to textual content, forcing the exclusion of valuable diagrams and graphics from the knowledge base. Second, because data ingestion relied on static PDF files, crucial provenance metadata such as source URLs could not be captured, directly impacting the implementation of the generic architecture's provenance component, preventing the system from tracing information back to its origin.

Section A.1 contains the different features of the n8n workflow implementation.

4.3.2 Configuring the Conversational Agent and Prompts

The system's conversational and reasoning capabilities are centered around n8n's Agent Node. This component serves as the workflow's decision-making core, leveraging the GPT-4.1-mini Large Language Model to understand user goals, interact with the knowledge base, and determine the next course of action.

The choice of GPT-4.1-mini was deliberate, driven by its effective balance of advanced capabilities, response speed, and cost-efficiency. Its large context window of up to 1 million tokens was also a critical factor, enabling it to process complex pension domain documents without losing coherence [51].

Single-agent Approach Initial design considerations explored a multi-agent orchestration, where a main agent would delegate tasks to specialized sub-agents. The goal was to create layered workflows for different responsibilities. However, testing revealed that this approach introduced a level of entropy that led to inconsistent chatbot responses, degrading the user experience during prolonged interactions.

Consequently, the architecture was simplified to a more robust and predictable model. It

now consists of a single main agent responsible for general chat interactions, supplemented by two conditionally triggered agents, each with a specific function:

- Documentation Generation Agent: Activates to automatically generate documentation for Confluence.
- User Story Generation Agent: Activates to create structured user stories for Jira.

This architecture proved to be more reliable, maintaining the intended functionality while ensuring a more consistent user interaction.

Guiding the agent's behavior through a system prompt Controlling the agent's behavior is essential, as the quality of its output is directly tied to the instructions, or prompts, it receives [16]. Even minor changes in these prompts can dramatically alter the generated responses [17]. The primary mechanism for this control is the system prompt, a special instruction provided at the beginning of an interaction that defines the agent's role, behavior, and operational boundaries. Unlike a user's query, the system prompt tells the agent "who it is," "what it should do," and "how it should behave" throughout the entire conversation.

Within this node, the system prompt was crafted based on the findings from the analyst interviews to govern the agent's behavior. It focuses on recognizing system limitations and avoiding providing unsupported information.

The chosen system prompt emphasized expertise in requirements elicitation practices and analyst support capabilities. The system prompt was crafted to establish the agent's role as a domain expert while maintaining boundaries regarding information accuracy and reliability, addressing the trust and validation concerns identified in the analyst interviews.

- Role definition: The agent is explicitly told: "You are a specialized assistant for requirements engineering and business analysis in the Dutch pension law sector", setting its expert persona.
- Enforcing RAG: To build trust and prevent hallucinations, the prompt includes a "MANDATORY SEARCH PROTOCOL", "Source Priority" and "Boundaries" section in its system prompt that forces the agent to always search its knowledge base first before answering pension-related questions. Additionally, it contains a "Primary goal" section, where it's instructed to always prioritize accuracy over assumptions.
- Acknowledging limitations: To address analysts' trust concerns, the prompt instructs the agent to state when it is uncertain: "I cannot verify this information. Please confirm with [appropriate source]", "Acknowledge limitations transparently", and "State explicitly when information is unavailable".

Separate, specialized agents were created for documentation and user story generation, each with its own tailored system prompt to ensure high-quality, structured output for specific tasks.

Chapter A.1, beginning on A.1, contains the full system prompts used for the agents.

4.3.3 Implementing the Interface and Tool Layers

This section outlines the process of selecting and implementing the user interface for the chatbot prototype, which serves as the window to the backend agents and tool integrations.

Given the target organization's use of the Microsoft Office ecosystem, Microsoft Teams was the logical initial choice for the primary interface. This approach was motivated by the potential for seamless integration into existing analyst workflows, which would lower adoption barriers. However, a technical feasibility assessment revealed significant obstacles. Integrating with n8n required Teams API credentials, which were restricted to company administrators, creating an access barrier for the project. Furthermore, maintaining conversation continuity required premium Power Automate subscriptions, introducing prohibitive costs.

Following these challenges, Telegram was selected as the final user interface. Its open API for bot development provided the automation and integration capabilities required by the project without the administrative and cost constraints of Teams. The implementation was straightforward, requiring only the creation of a Telegram bot whose message processing was handled entirely within the n8n workflow.

To create a structured and intuitive user experience, the final architecture leverages Telegram's native command functionality [52]. This command-based design allows the user to direct the chatbot's actions after an initial conversation. The final workflow enables a user to issue specific commands that trigger corresponding logic branches in n8n:

1. **/newstory**: This command passes the recent conversation history to the User Story Generation Agent. The agent processes the information, formats it into a structured user story, and, upon user approval, posts it directly to a Jira project.
2. **/updatedoc**: This command sends the conversation to the Documentation Generation Agent, which creates a summary or process document and, with approval, publishes it to a Confluence page.

Figure A.4 illustrates how these commands look in Telegram. This command-based branching architecture provides clear pathways for the user while maintaining conversational context, ensuring the output is correctly formatted for the target systems. This implementation, summarized in the Business Process Model and Notation (BPMN) diagram, serves as a practical instantiation of the generic framework, tailored to solve the specific problems

identified in the Dutch pension project.

The tool integration layer was realized using n8n's HTTP request node, connecting it to Jira and Confluence's exposed APIs. The user interface was built using Telegram, which offered a simple and effective API for bot development after technical barriers prevented the use of Microsoft Teams.

4.3.4 Final Design

This implementation, summarized in a process diagram in Figure 4.2 to better understand the internal division of responsibility, is a practical instantiation of the generic framework, tailored to solve the problems identified in the Dutch pension project.

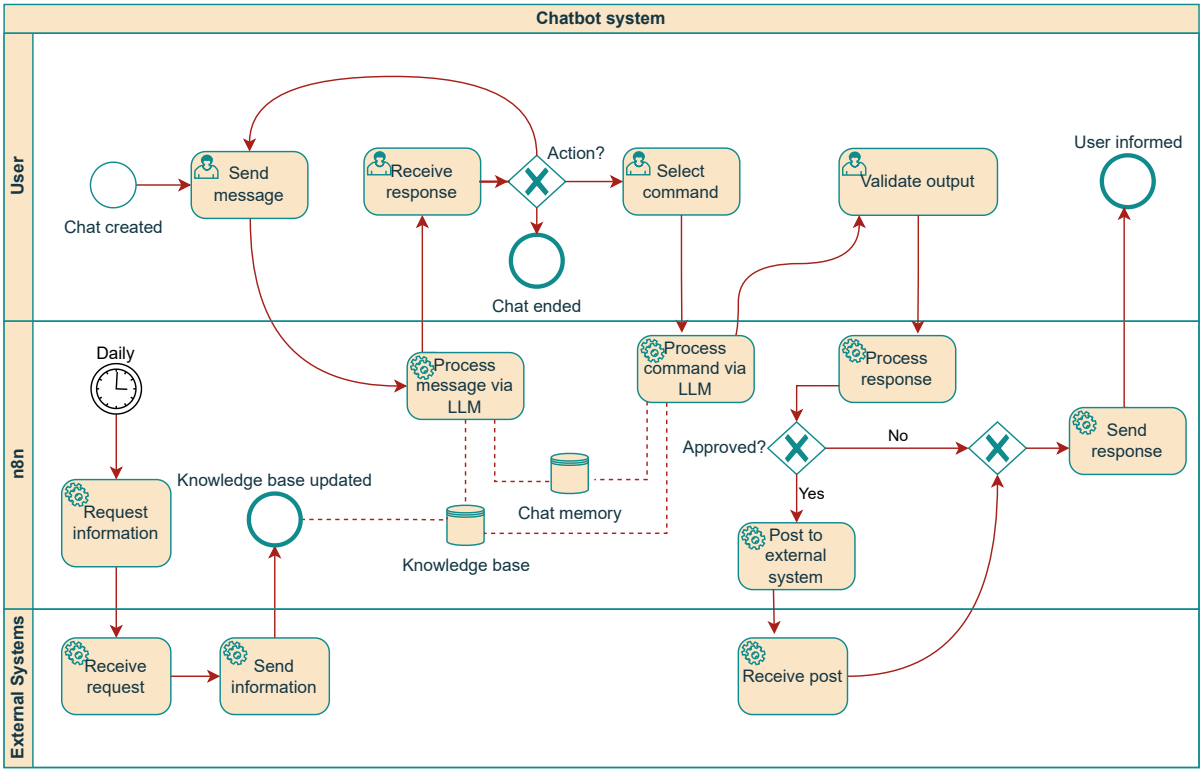


Figure 4.2: BPMN diagram of chatbot interaction

Figure 4.3 illustrates the layered architecture of the AI chatbot. The system is organized into five layers, Presentation, Orchestration, AI Reasoning, Knowledge and Retrieval, and Data Sources, each having a clearly defined responsibility.

The Business Analyst interacts with the system through the Presentation Layer, implemented as a Telegram Bot Interface. The Orchestration Layer, built on n8n, coordinates message routing, conversation memory, agent invocation, and communication with external

systems. A Human-in-the-Loop validation step ensures that generated artifacts are reviewed and approved before being published to Jira or Confluence.

The AI Reasoning Layer employs three specialized agents, for general conversation, user story generation, and documentation drafting, each governed by structured system prompts and powered by Azure OpenAI GPT-4.1-mini. The Knowledge and Retrieval Layer implements the RAG mechanism, ingesting project documentation into a Pinecone vector database and retrieving relevant context during conversations. The Data Sources Layer contains the Confluence and Dutch pension domain documents used to populate the knowledge base; ADO tickets were excluded due to privacy constraints.

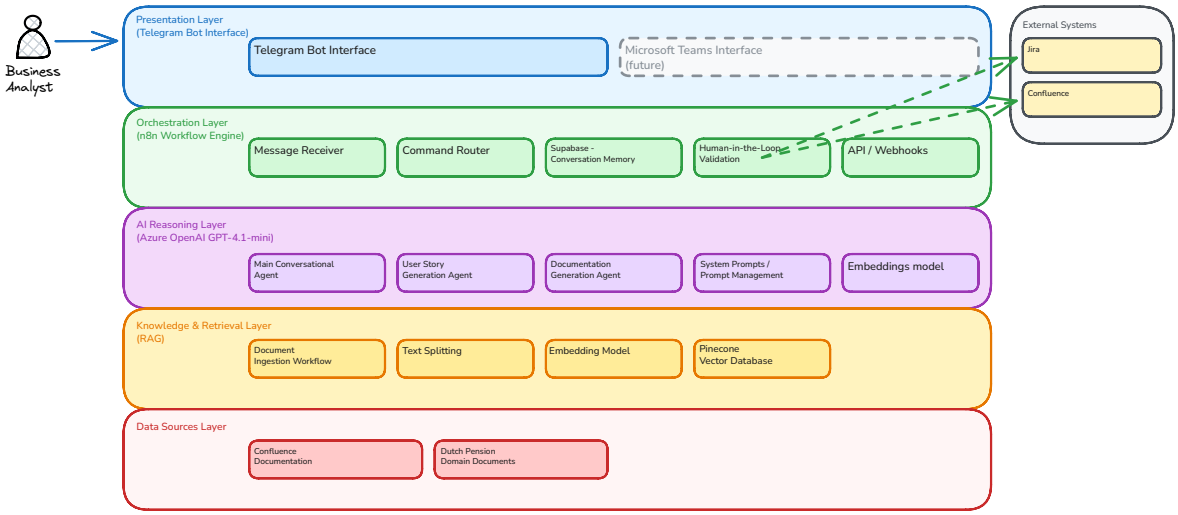


Figure 4.3: Layered architecture of the AI chatbot implementation

5

Demonstration

Contents

This chapter presents the qualitative and quantitative feedback collected from five business analysts who interacted with the AI-powered chatbot prototype.

5.1 Final Interview Protocol

Before each session, participants provided explicit consent to be recorded, with the assurance that all recordings would be used solely for data extraction and subsequently deleted. The purpose of the research was clearly explained, and participants were informed that all collected data would be anonymized to ensure confidentiality. It was also made clear that they could stop the interview at any point or decline to answer any questions.

The demonstration aimed to assess the usefulness of the AI system, the accuracy of its requirements suggestions, and its impact on the analysts' effectiveness and productivity. The method considered for evaluating knowledge, according to Hevner's types of evaluation, was observational and experimental methods [9].

Qualitative data was gathered through semi-structured interviews. These interviews involved the same five analysts who participated in the initial requirements gathering. Each analyst was given a tailored demonstration based on topics relevant to their daily work, allowing them to interact with the chatbot in realistic scenarios to capture their user experiences, perceived usefulness, confidence in the system, and suggestions for improvement.

Each interview lasted about 60 minutes and was structured into three phases: a 5-minute introduction, a 25-minute chatbot demo and interaction phase, and a 20-minute post-demo evaluation. The demonstration phase consisted of a 5-minute tutorial followed by 20 minutes of tasks, during which analysts performed basic information retrieval, user story creation, and documentation generation using the chatbot.

Quantitative data was collected using Google Forms to establish a System Usability Scale (SUS) score and evaluate general criteria related to the chatbot.

The full interview script can be consulted in chapter B, section B.2.

The results from this demo session are detailed in the following sections, consisting of task-specific ratings on accuracy, relevance, speed, ease of interaction, trust, and overall usefulness, on Table 5.1 and Table 5.2; open-ended questions that explored users' likes, frustrations, trust in the system, and suggestions for improvement and a SUS questionnaire, with the results presented on table 5.3 and 5.4.

5.2 Chatbot Evaluation Criteria and Qualitative Observations

This section breaks down the user experience by analyzing specific performance ratings and qualitative feedback, providing context for the chatbot's perceived strengths and weaknesses.

Participant	Accuracy	Relevance	Speed	Ease of interaction	Trust	Usefulness
A3	4	4	6	6	2	2
A5	5	3	5	5	3	5
A1	5	6	7	7	5	5
A4	5	5	3	7	5	5
A2	6	6	5	5	6	5
Average	5.0	4.8	5.2	6.0	4.2	4.4

Table 5.1: Analyst ratings for chatbot evaluation criteria on a 1 to 7 scale.

The accuracy of the information provided by the chatbot was rated moderately high (5 out of 7, on average). Analysts consistently acknowledged the chatbot's strength in retrieving and summarizing relevant organizational knowledge from internal Confluence documentation and existing project processes. This ability was considered to have potential in assisting new team members unfamiliar with existing requirements and process configuration by providing quick access to project data.

However, analysts also noted that the chatbot occasionally struggled with gaps in domain-specific knowledge and specific process details, issues attributed to limited documentation on more technical aspects, such as developer documentation. Many responses were described as too vague or overly generic, failing to provide the nuanced and process-specific details that an analyst would need.

The relevance of responses also showed moderate variance (4.8 average out of 7), with some analysts appreciating the chatbot's ability to remain context-aware during conversation, particularly in distinguishing between processes inside the project. Others pointed out moments where the chatbot's focus on the latest user message without broader session memory led to fragmented answers, affecting coherence.

Speed was generally perceived positively, averaging 5.2 out of 7, suggesting adequate real-time interaction performance suitable for conversational use. The interface design and command buttons were praised for their clarity and ease of access, reducing the need for training and enabling immediate use.

Despite this, some users reported minor inconveniences, such as the chatbot's restricted knowledge cut-off points, which occasionally delayed or required re-asking queries. Additionally, optimal use of features like user story generation and documentation drafts depended partially on user familiarity with the chatbot's specific command set.

Ease of interaction scores averaged 6 out of 7, indicating that the analysts found the chatbot very easy to use.

Trust scores were the lowest, averaging 4.2 out of 7, showing that while the chatbot is generally regarded as a reliable knowledge assistant, there was an emphasis on the necessity of transparency regarding information sources. The chatbot's practice of honestly revealing its limitations rather than presenting uncertain information as fact was praised as enhancing its trustworthiness.

Nevertheless, some criticized the chatbot's reliance on potentially outdated or incomplete documentation, requiring experts to manually verify outputs before accepting them as final. Trust was further reduced by the absence of visible source information and timestamps in the chatbot's responses.

Analysts' overall usefulness ratings averaged 4.4 out of 7, with only one analyst not finding much use in it (a score of 2), while the rest valued it at 5. The chatbot was most valued for onboarding new analysts, quickly summarizing well-documented processes, and reducing routine manual labor when drafting user stories or verifying if a specific piece of information existed in the documentation.

In contrast, many expressed caution in depending on the chatbot for complex elicitation or as a replacement for human judgment and expert review. The chatbot was not viewed as authoritative enough for final requirement validation or for handling specialized sector knowledge without substantial oversight.

Table 5.2: Missing Features Identified by Analysts

Missing Feature	**A1**	**A2**	**A3**	**A4**	**A5**
Confidence Indicators	X		X		X
More domain documentation		X	X	X	
Verification by SME	X	X	X	X	X
Surface-level answers	X		X	X	X
Source links/references	X	X		X	X
More proactiveness		X	X	X	X
More context in answers		X	X	X	
Clear onboarding message		X			
Freedom to choose between speed vs detail in responses	X				
Adversarial mode			X		

The chatbot demonstrates an ability to leverage recent conversational context, but often neglects extended dialogue history. As a result, it lacks multi-turn conversational coherence, which is crucial for iterative refinement of requirements.

By displaying caution and explicit uncertainty, the analysts' confidence in the chatbot was increased.

Analysts desire cleaner, more standardized formatting for generated documentation drafts, with options for direct export to standard formats (PDF, DOCX), thereby facilitating their incorporation into existing workflows.

5.3 System Usability Scale Scores

The System Usability Scale is a standardized, technology-agnostic instrument for evaluating perceived usability through a ten-item questionnaire rated on a five-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree." The SUS yields a single score from 0 to 100, facilitating comparisons with different products and against established benchmarks and adjective ratings (e.g., "Excellent", "Good", or "OK") and providing a reliable measure of overall usability [53].

The scale alternates between positively and negatively worded statements to mitigate agreement bias and encourage more discriminative responses during post-session evaluations. For each participant, responses to positively worded items are normalized by subtracting 1, and responses to negatively worded items are normalized by subtracting the rating from 5. The adjusted values are summed and multiplied by 2.5 to produce a 0–100 usability index [54].

SUS score interpretation is based on empirical norms established in the literature, where a score of 68 serves as an approximate benchmark for acceptable usability. Scores above this threshold indicate above-average usability. Values around 74–80 typically correspond to "Good" usability and are associated with a higher adoption likelihood, whereas scores of 80 or greater are classified as "Excellent." Conversely, scores between 51 and 68 suggest marginal usability with evident friction points, and scores below 51 generally indicate poor usability, deserving substantial redesign before further development.

Table 5.3: SUS Adjusted Scores

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
A3	1	1	5	1	3	2	4	2	3	1	29
A5	4	2	3	1	3	3	4	2	2	3	25
A1	4	1	5	1	4	2	5	1	4	1	36
A4	4	1	5	1	4	2	5	1	5	1	37
A2	3	2	4	2	4	2	4	1	4	3	29

The SUS was administered to five participants (n = 5) to evaluate the prototype's perceived usability. The mean SUS score was 78.0, which falls within the 'Good' range according to acceptability benchmarks [54].

However, analysis revealed a high degree of variability in the individual scores, with a

Table 5.4: SUS Scores per Participant

Participant	SUS Score
A3	72.5
A5	62.5
A1	90.0
A4	92.5
A2	72.5
Mean	**78.0**
Standard deviation	**12.8**
95% Confidence interval	**[62.1, 93.9]**

standard deviation of 12.8. This suggests that while some users found the system highly usable, others encountered significant difficulties.

The 95% confidence interval for the mean score was calculated to be [62.1, 93.9]. The considerable width of this interval is a result of the small sample size and high variability in scores. Consequently, while the point estimate of 78.0 is promising, the actual population mean could lie anywhere from an "OK" score to an "Excellent" one. This shows the preliminary nature of this finding and highlights both the need for further testing with a larger sample size to obtain a more precise estimate of the system's overall usability, as well as the improvement of consistency of experience across user groups.

6

Evaluation

Contents

This chapter offers a preliminary evaluation of the AI-powered chatbot prototype. Building on the empirical results from the small-scale user study presented in the previous chapter, this section assesses the artifact's potential to address the core research problem: overcoming communication gaps, documentation inconsistencies, and reliance on manual effort in requirements elicitation. The analysis interprets the initial findings, discusses the implications for practice, and summarizes the chatbot's performance within the context of this limited sample.

6.1 Interpretation of Findings

The prototype's performance can be interpreted by analyzing it against the specific challenges it was designed to help mitigate.

6.1.1 Addressing Knowledge Gaps and Documentation Inconsistencies

The chatbot showed potential to function as a centralized knowledge repository. By retrieving and summarizing information from the project's Confluence domain, it appears to help mitigate the challenge of knowledge dispersion. Analysts noted this feature could be particularly useful for quick information access and for onboarding new team members.

However, the prototype's utility was limited by the quality and completeness of its source documentation, as well as the user's inability to verify that source. As analysts noted, gaps in the knowledge base, specifically more technical product documentation, led to occasional inaccuracies or overly generic responses, restricting the chatbot's usefulness for more complex topics. This suggests that while the artifact can be an effective knowledge retrieval tool, its accuracy is a direct result of the underlying data.

The built-in feature to continuously ingest new documentation, presented in Figure A.3, is a necessary step to address this; however, its long-term impact could not be assessed in the single-session demo.

6.1.2 Reducing Manual Effort in Repetitive Tasks

The automated user story and documentation generation features were generally well-received. The prototype appeared to reduce the initial manual effort of drafting these artifacts, providing structured templates.

This functionality has the potential to accelerate routine tasks and, more importantly, encourage the creation of documentation where none might have existed before.

Despite this, analysts emphasized that the outputs still require significant human oversight. The evaluation suggests that the chatbot functions best as a drafting assistant rather than a replacement for an analyst's critical thinking for finalizing requirements. Concerns about potential errors surfacing from outdated source data led to a lack of trust, causing participants to view the tool as an assistive or accelerator technology instead of a standalone solution.

6.1.3 Usability and Trust

The prototype achieved a promising level of usability, indicated by the above-average SUS score of 78. The 95% confidence interval result reveals that the true average SUS score for the chatbot stands between 62.1 and 93.9, with a 95% confidence. This variation suggests that certain tasks, workflows, or user segments encountered differing levels of friction or effort when interacting with the chatbot.

This finding should be interpreted cautiously due to the small sample size, but it suggests the interface and workflow are accessible. Analysts saw the integration with Jira and Confluence as an advantage, as it can be embedded within their established workflows, increasing the likelihood of adoption.

This evaluation suggests a notable challenge with user trust in the prototype, reflected in the low average trust score of 4.2 out of 7, leading to the output not being dependable, especially in the highly regulated sector of pension law. An analysis of the interviews suggests that it stems from the lack of provenance of information, which was a core feature that could not be implemented in the current prototype. Without such a feature, the system provided information without being backed by sources, necessitating manual verification of the chatbot's claims.

An official confidence score indicator was also missing from the prototype, tied to the lack of provenance metadata, preventing the chatbot from making claims on confidence scores.

Nonetheless, the chatbot's transparency in acknowledging its own knowledge gaps rather than providing uncertain information was noted as a positive factor in building trust.

6.2 Discussion: Positioning the Chatbot

The findings from this evaluation offer some insight into the most effective role for AI in this context. The research explored whether an AI-powered chatbot could primarily replace, assist, or be an alternative to traditional manual methods.

The results point towards the chatbot being most effective as a **hybrid support tool that augments, rather than replaces, human expertise**. Its perceived strengths are in accelerating information retrieval and automating initial drafting tasks. At the same time, its limitations, such as its dependency on source data quality and the need for nuanced critical thinking, reinforce the necessity of keeping a HITL for validation and refinement. This positioning of the AI-powered chatbot as a collaborative assistant is the most appropriate takeaway from this research work.

6.3 Summary

This evaluation suggests that the AI-powered chatbot prototype offers a promising approach for addressing several key challenges in requirements engineering, such as improving documentation accessibility, reducing manual effort in drafting, and supporting knowledge transfer. Its current limitations regarding contextual scope, trust mechanisms, and data completeness highlight the need for human oversight and limit its potential for autonomous operation. For the chatbot to realize its potential, future work should focus on enhancing its context retention, source transparency, and knowledge base enrichment, strengthening its utility as a tool for analysts.

7

Conclusion

Contents

RE is a critical phase in software development, with its quality directly impacting project success. LLMs show promise in supporting this phase by facilitating requirements elicitation, analysis, and documentation with improved efficiency and accuracy.

The SLR and interviews with experts revealed significant challenges, such as knowledge fragmentation, limited trust in AI-based solutions, and the ongoing need for human validation. Additionally, a scarcity of empirical studies and practical validations currently restricts widespread adoption of these technologies.

This work proposes an AI-powered chatbot designed specifically to elicit and document requirements in complex contexts. The prototype integrates with external systems, such as Jira and Confluence, to publish and retrieve new information, and aims to reduce ambiguities, speed up repetitive tasks, and help analysts obtain relevant information.

Demonstration and evaluation involved final interviews with experienced analysts who used the chatbot in a realistic setting. Both qualitative and quantitative assessments suggested improved knowledge retrieval and reduction in manual effort of documentation. However, its evaluation also highlighted the importance of human-in-the-loop validation to ensure accuracy and build trust.

The findings from this research suggest that a collaborative approach could be an effective path for integrating AI into requirements engineering. In this model, the chatbot could act as an intelligent assistant that handles repetitive tasks and surfaces information, potentially freeing analysts to focus on critical thinking, strategic decision-making, and stakeholder communication.

7.1 Contributions

This thesis investigates how conversational AI can support documentation retrieval, knowledge dissemination, and user story generation in highly regulated enterprise contexts. Given the limited practical demonstrations in existing literature, it offers an initial step toward understanding how LLMs could be adopted within enterprise tools such as Jira and Confluence.

Through a SLR, the thesis consolidates current research on AI-powered chatbots in requirements engineering, summarizing their reported capabilities, challenges, and research gaps. The review indicates a lack of real-world validation. It aims to provide a preliminary foundation for future empirical studies and the development of AI-assisted elicitation tools. It outlines the current state of knowledge and highlights areas where further exploration could help bridge theory and practice.

7.2 Limitations

The findings of this research should be interpreted in light of several limitations, which can be grouped into methodological and technical limitations.

7.2.1 Methodological Limitations

Despite the best efforts to perfect the search string and conduct a structured database search for the SLR, there may have been more effective search strings and databases available to locate them. As such, some relevant papers may have been overlooked.

The system prompts for the agents were based on the outcome from the first interviews, but exploring more prompt variations and doing thorough testing could have improved the agents' performance.

The fact that the prototype underwent only a single-session evaluation prevented the observation of long-term effects and real-world usage scenarios. It provided limited insight into the potential adoption of the chatbot and the evolution of user behavior with its utilization. In addition, the demo was limited to historical scenarios, so analysts were left to wonder how well the chatbot would perform in handling novel or unforeseen contexts.

The small sample size (n = 5) from a single project also considerably limited the statistical power and generalizability of the results.

7.2.2 Technical Limitations

The chatbot's knowledge and perceived usefulness depended on the availability of documentation, which was sometimes incomplete, particularly in terms of developer/technical documentation, and potentially outdated, thereby reducing its ability to provide accurate information.

Regarding the interface, Telegram did not allow session states to be passed on to the conversation that would persist during the entire interaction with the users, making it impossible for the chatbot to assume different personas, for example, which a few analysts would have liked.

Current LLMs are constrained in their ability to handle extensive context without incurring significant costs associated with large token usage. Therefore, measures were taken to reduce the size of system prompts to keep the usage costs balanced, possibly affecting performance.

7.3 Future Work

To strengthen the validity and generalizability of the findings, future research should incorporate a more longitudinal study design. Extended use of the prototype in daily work environments would enable the observation of usage patterns, adoption effects, and emerging challenges over time. Future studies should expand the sample size to include a more diverse and representative population across different teams and projects. This will enable a more robust statistical comparison and a deeper understanding of the chatbot's impact on requirements elicitation in real-world contexts.

The study results show an apparent demand for more precise indications of information origin and confidence scores. To ensure an accurate and reliable knowledge base, future research should combine robust data cleaning techniques, such as removing duplicates, inconsistencies, conflicts, and outdated information, with a powerful upserting mechanism to update or insert new data efficiently. Better data hygiene will reduce hallucinations and increase user trust by improving the quality of generated responses. Enriching the information with metadata by adding details like timestamps, confidence scores, and source links is also crucial to improving information quality.

Future research should also focus on transforming the chatbot into an active collaborator and critic. This involves developing proactive features, such as automated gap detection, guided questioning, and a "Red Team" mode. In this mode, the chatbot would intentionally challenge assumptions and stress-test requirements to expose weaknesses and hidden dependencies, ensuring they are robust and complete before implementation.

One promising direction is deploying locally fine-tuned language models to safeguard data confidentiality. Unlike cloud-based models, local deployment ensures sensitive information remains within the organization, mitigating data leakage and ensuring compliance (e.g., General Data Protection Regulation (GDPR)). Fine-tuning, which involves additional training on smaller, domain-specific datasets, enables the model to combine general knowledge (from sources such as books and the web) with specialized understanding, producing more contextually accurate and confidential outputs during requirements elicitation and documentation.

7.3.1 Technology

During thesis work, several technical directions were considered that could not be included in the prototype due to cost or time limits. The most significant of these is the application of a graph-based knowledge base. Unlike conventional document stores, a graph database models information as nodes and relationships, offering a natural fit for capturing the complex dependencies and semantics inherent in requirements engineering. This structure could

enhance traceability, enabling a more detailed impact analysis and dependency mapping. For a future iteration of the chatbot prototype, this would translate to more context-aware information retrieval, the ability to reason across connected artifacts, and advanced querying capabilities, such as pathfinding and pattern detection, to analyze requirement evolution and identify conflicts.

Additionally, the research suggests that context engineering may be a promising direction for enhancing the output of these AI-assisted workflows. This emerging discipline moves beyond static prompt design to focus on the dynamic orchestration of an AI's working context. By selecting and managing the flow of information, such as relevant project data, tool usage and ouputs, and the conversation history, a system can maintain a more coherent and nuanced understanding of tasks.

Bibliography

[1] A. Tikayat Ray, B. F. Cole, O. J. Pinon Fischer, A. P. Bhat, R. T. White, and D. N. Mavris, "Agile methodology for the standardization of engineering requirements using large language models," *Systems*, vol. 11, no. 7, p. 352, 2023, ISSN: 20798954. DOI: 10.3390/systems11070352.

[2] F. P. B. Jr., "No silver bullet - essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, 1987. DOI: 10.1109/MC.1987.1663532. [Online]. Available: https://doi.org/10.1109/MC.1987.1663532.

[3] X. Peng, "Software development in the age of intelligence: Embracing large language models with the right approach," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, no. 11, pp. 1513–1519, 2023, ISSN: 2095-9230. DOI: 10.1631/FITEE.2300537. Accessed: Feb. 8, 2024. [Online]. Available: https://doi.org/10.1631/FITEE.2300537.

[4] C. Arora, J. Grundy, and M. Abdelrazek. "Advancing requirements engineering through generative AI: Assessing the role of LLMs." arXiv: 2310.13976[cs], Accessed: Feb. 10, 2024. [Online]. Available: http://arxiv.org/abs/2310.13976.

[5] N. N. B. Abdullah, S. Honiden, H. Sharp, B. Nuseibeh, and D. Notkin, "Communication patterns of agile requirements engineering," in *Proceedings of the 1st Workshop on Agile Requirements Engineering*, ser. AREW '11, New York, NY, USA: Association for Computing Machinery, 2011, pp. 1–4, ISBN: 978-1-4503-0890-8. DOI: 10.1145/2068783.2068784. Accessed: Mar. 3, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/2068783.2068784.

[6] "ChatGPT," Accessed: Mar. 3, 2024. [Online]. Available: https://chat.openai.com.

[7] "Gemini - o chat para explorar as suas ideias," Gemini, Accessed: Mar. 3, 2024. [Online]. Available: https://gemini.google.com.

[8] "Microsoft copilot: O seu complemento de IA para o dia a dia," Microsoft Copilot: o seu complemento de IA para o dia a dia, Accessed: Mar. 3, 2024. [Online]. Available: https://ceto.westus2.binguxlivesite.net/.

[9] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, 2004.

[10] K. Peffers, T. Tuunanen, C. E. Gengler, M. Rossi, and W. Hui, "Design science research process: A model for producing and presenting information systems research," *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST)*, 2006.

[11] A. Nightingale, "A guide to systematic literature reviews," *Surgery (Oxford)*, Determining surgical efficacy, vol. 27, no. 9, pp. 381–384, 2009, ISSN: 0263-9319. DOI: 10.1016/j.mpsur.2009.07.005. Accessed: Feb. 27, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263931909001707.

[12] B. Kitchenham, "Procedures for performing systematic reviews," Keele University and NICTA (National ICT Australia Ltd), Tech. Rep., 2004.

[13] P. Tominc, D. Oreški, and M. Rožman, "Artificial intelligence and agility-based model for successful project implementation and company competitiveness," *Information (2078-2489)*, vol. 14, no. 6, p. 337, 2023, ISSN: 20782489. DOI: 10.3390/info14060337.

[14] A. A. Khan, J. A. Khan, M. A. Akbar, P. Zhou, and M. Fahmideh, "Insights into software development approaches: Mining q &a repositories," *Empirical Software Engineering*, vol. 29, no. 1, pp. 1–38, 2024, ISSN: 13823256. DOI: 10.1007/s10664-023-10417-5.

[15] B. Wilson. "What's the difference between a large language model (LLM) and a general pre-trained transformer (GPT)?" Brin Wilson... Accessed: Feb. 26, 2024. [Online]. Available: https://www.brinwilson.com/whats-the-difference-between-a-large-language-model-llm-and-a-general-pre-trained-transformer-gpt/.

[16] M. Bano, R. Hoda, D. Zowghi, and C. Treude, "Large language models for qualitative research in software engineering: Exploring opportunities and challenges," *Automated Software Engineering*, vol. 31, no. 1, pp. 1–12, 2024, ISSN: 09288910. DOI: 10.1007/s10515-023-00407-8.

[17] M. Gerosa, B. Trinkenreich, I. Steinmacher, and A. Sarma, "Can AI serve as a substitute for human subjects in software engineering research?" *Automated Software Engineering*, vol. 31, no. 1, pp. 1–12, 2024, ISSN: 09288910. DOI: 10.1007/s10515-023-00409-6.

[18] A. Bandi, P. V. S. R. Adapa, and Y. E. V. P. K. Kuchi, "The power of generative AI: A review of requirements, models, input–output formats, evaluation metrics, and challenges," *Future Internet*, vol. 15, no. 8, p. 260, 2023, ISSN: 19995903. DOI: 10.3390/fi15080260.

[19] Google. "Google ai overview," Accessed: Oct. 10, 2025. [Online]. Available: https://ai.google/.

[20] Meta. "Meta ai research overview," Accessed: Oct. 10, 2025. [Online]. Available: https://ai.meta.com/.

[21] OpenAI. "Openai product pricing," Accessed: Oct. 10, 2025. [Online]. Available: https://openai.com/pricing.

[22] "Hugging face – the AI community building the future.," Accessed: Oct. 14, 2025. [Online]. Available: https://huggingface.co/.

[23] "The implications of large language models in physical security," Accessed: Feb. 25, 2024. [Online]. Available: https://www.genetec.com/blog/cybersecurity/the-implications-of-large-language-models-in-physical-security.

[24] Mingxing Liu, Junfeng Wang, Tao Lin, Quan Ma, Zhiyang Fang, and Yanqun Wu, "An empirical study of the code generation of safety-critical software using LLMs," *Applied Sciences*, vol. 14, no. 3, pp. 1046–1046, 2024, Publisher: MDPI AG, ISSN: 2076-3417. DOI: 10.3390/app14031046.

[25] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. "Challenges and applications of large language models." arXiv: 2307.10169[cs], Accessed: Feb. 25, 2024. [Online]. Available: http://arxiv.org/abs/2307.10169.

[26] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt. "ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design." arXiv: 2303.07839[cs], Accessed: Feb. 13, 2024. [Online]. Available: http://arxiv.org/abs/2303.07839.

[27] N. Marques, R. R. Silva, and J. Bernardino, "Using ChatGPT in software requirements engineering: A comprehensive review," *Future Internet*, vol. 16, no. 6, p. 180, 2024, Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1999-5903. DOI: 10.3390/fi16060180. Accessed: Jul. 29, 2025. [Online]. Available: https://www.mdpi.com/1999-5903/16/6/180.

[28] "Pesquisa básica: Discovery service da universidade aberta," Accessed: Mar. 3, 2024. [Online]. Available: https://eds.p.ebscohost.com/eds/search/basic?vid=0&sid=815539ba-df93-4322-982b-dbe8a27d4077%40redis.

[29] T. Follin. "LibGuides: The EBM medical literature search: Snowball search," Accessed: Mar. 3, 2024. [Online]. Available: https://libguides.fau.edu/MedicalLitSearch/snowballsearch.

[30] M. Javaid, A. Haleem, and R. P. Singh, "A study on ChatGPT for industry 4.0: Background, potentials, challenges, and eventualities," *Journal of Economy and Technology*, vol. 1, pp. 127–143, 2023, ISSN: 2949-9488. DOI: 10.1016/j.ject.2023.08.001. Accessed: Jan. 23, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949948823000033.

[31] V. Saklamaeva and L. Pavlič, "The potential of AI-driven assistants in scaled agile software development," *Applied Sciences (2076-3417)*, vol. 14, no. 1, p. 319, 2024, ISSN: 20763417. DOI: 10.3390/app14010319.

[32] Z. Szabó and V. Bilicki, "A new approach to web application security: Utilizing GPT language models for source code inspection," *Future Internet*, vol. 15, no. 10, p. 326, 2023, ISSN: 19995903. DOI: 10.3390/fi15100326.

[33] R. Budake, S. Bhoite, and K. Kharade, "A study of AI-based techniques for requirement analysis in software engineering," *AIP Conference Proceedings*, vol. 2946, no. 1, pp. 1–6, 2023, ISSN: 0094243X. DOI: 10.1063/5.0178114.

[34] R. Dedhia, "Techniques to automatically generate entity relationship diagram," *International Journal of Innovations Advancement in Computer Science*, 2015. Accessed: Feb. 11, 2024. [Online]. Available: https://www.academia.edu/77315506/Techniques_to_automatically_generate_Entity_Relationship_Diagram.

[35] S. Hassan, Q. Li, K. Aurangzeb, A. Yasin, J. A. Khan, and M. S. Anwar, "A systematic mapping to investigate the application of machine learning techniques in requirement engineering activities," *CAAI Transactions on Intelligence Technology*, vol. 9, no. 6, pp. 1412–1434, 2024, _eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/cit2.123 ISSN: 2468-2322. DOI: 10.1049/cit2.12348. Accessed: Jul. 29, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12348.

[36] S. K. Mohan, "Management consulting in the artificial intelligence – LLM era," *Management Consulting Journal*, vol. 7, no. 1, pp. 9–24, 2024, ISSN: 2631987X. DOI: 10.2478/mcj-2024-0002.

[37] J. Motger de la Encarnación, "Natural language processing methods for document-based requirements specification and validation tasks," Accepted: 2024-11-13T08:40:34Z Publication Title: TDX (Tesis Doctorals en Xarxa), Ph.D. dissertation, Universitat Politècnica de Catalunya, Nov. 4, 2024. DOI: 10.5821/dissertation-2117-417795. Accessed: Jul. 29, 2025. [Online]. Available: https://www.tdx.cat/handle/10803/692516.

[38] Rex Black, James Davenport, Joanna Olszewska, Jeremias Rößler, Adam Leon Smith, and Jonathon Wright, *Artificial Intelligence and Software Testing : Building Systems You Can Trust*. Swindon: BCS, The Chartered Institute for IT, 2022, ISBN: 978-1-78017-576-8. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib,uid&db=e250xww&AN=3105913&lang=pt-pt&site=eds-live&custid=ns000545.

[39] T. Nakata, M. Nakamura, S. Chen, and S. Saiki, "Needs companion: A novel approach to continuous user needs sensing using virtual agents and large language models," *Sensors*, vol. 24, no. 21, p. 6814, 2024, Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s24216814. Accessed: Jul. 29, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/24/21/6814.

[40] V. Sonkin and C. Tudose, "Beyond snippet assistance: A workflow-centric framework for end-to-end AI-driven code generation," *Computers*, vol. 14, no. 3, p. 94, 2025, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2073-431X. DOI: 10.3390/computers14030094. Accessed: Jul. 29, 2025. [Online]. Available: https://www.mdpi.com/2073-431X/14/3/94.

[41] C. Gao, X. Hu, S. Gao, X. Xia, and Z. Jin, "The current challenges of software engineering in the era of large language models," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, 127:1–127:30, 2025, ISSN: 1049-331X. DOI: 10.1145/3712005. Accessed: Jul. 29, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3712005.

[42] C. Burnay, I. J. Jureta, and S. Faulkner, "What stakeholders will or will not say: A theoretical and empirical study of topic importance in requirements engineering elicitation interviews," *Information Systems*, vol. 46, pp. 61–81, 2014, ISSN: 03064379. DOI: 10.1016/j.is.2014.05.006.

[43] F. Dwitam and A. Rusli, "User stories collection via interactive chatbot to support requirements gathering," *Telkomnika*, vol. 18, no. 2, pp. 890–898, 2020, Publisher: Department of Electrical Engineering, Ahmad Dahlan University, ISSN: 16936930. DOI: 10.12928/TELKOMNIKA.v18i2.14866. Accessed: Jan. 14, 2024. [Online]. Available: https://portal.uab.pt/si/ligacao-vpn/.

[44] S. Borsci et al., "The chatbot usability scale: The design and pilot of a usability scale for interaction with AI-based conversational agents," *Personal and Ubiquitous Computing*, vol. 26, no. 1, pp. 95–119, 2022, ISSN: 1617-4909, 1617-4917. DOI: 10.1007/s00779-021-01582-9. Accessed: Jan. 14, 2024. [Online]. Available: https://link.springer.com/10.1007/s00779-021-01582-9.

[45] R. Fuentes-Fernández, J. Gómez-Sanz, and J. Pavón, "Understanding the human context in requirements elicitation," *Requirements Engineering*, vol. 15, no. 3, pp. 267–283, 2010, Publisher: Springer Nature, ISSN: 09473602. DOI: 10.1007/s00766-009-0087-7. Accessed: Feb. 8, 2024. [Online]. Available: https://portal.uab.pt/si/ligacao-vpn/.

[46] H. B. Reubenstein and R. C. Waters, "The requirements apprentice: Assistance for requirements acquisition," *IEEE Transactions on Software Engineering*, vol. 17, no. 3, pp. 226–240, 1991, ISSN: 00985589. DOI: 10.1109/32.75413.

[47] R. Guizzardi, G. Amaral, G. Guizzardi, and J. Mylopoulos, "An ontology-based approach to engineering ethicality requirements," *Software & Systems Modeling*, vol. 22, no. 6, pp. 1897–1923, 2023, ISSN: 16191366. DOI: 10.1007/s10270-023-01115-3.

[48] C. Lin and R. Levary, "Computer-aided software development process design," *IEEE Transactions on Software Engineering*, vol. 15, no. 9, pp. 1025–1037, 1989, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/32.31362. Accessed: Feb. 9, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/31362.

[49] "The dutch pension and insurance market: Outlook 2025," Accessed: Sep. 28, 2025. [Online]. Available: https://www.deloitte.com/nl/en/Industries/insurance/perspectives/de-nederlandse-pensioen-en-verzekeringsmarkt-outlook-2025.html.

[50] "What is RAG? - retrieval-augmented generation AI explained - AWS," Amazon Web Services, Inc. Accessed: Sep. 15, 2025. [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/.

[51] "Model GPT-4.1 mini - OpenAI API," Accessed: Oct. 1, 2025. [Online]. Available: https://platform.openai.com.

[52] "Commands," Accessed: Sep. 23, 2025. [Online]. Available: https://core.telegram.org/api/bots/commands.

[53] J. Brooke, “SUS: A quick and dirty usability scale,” in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds., Taylor & Francis, 1996, pp. 189–194.

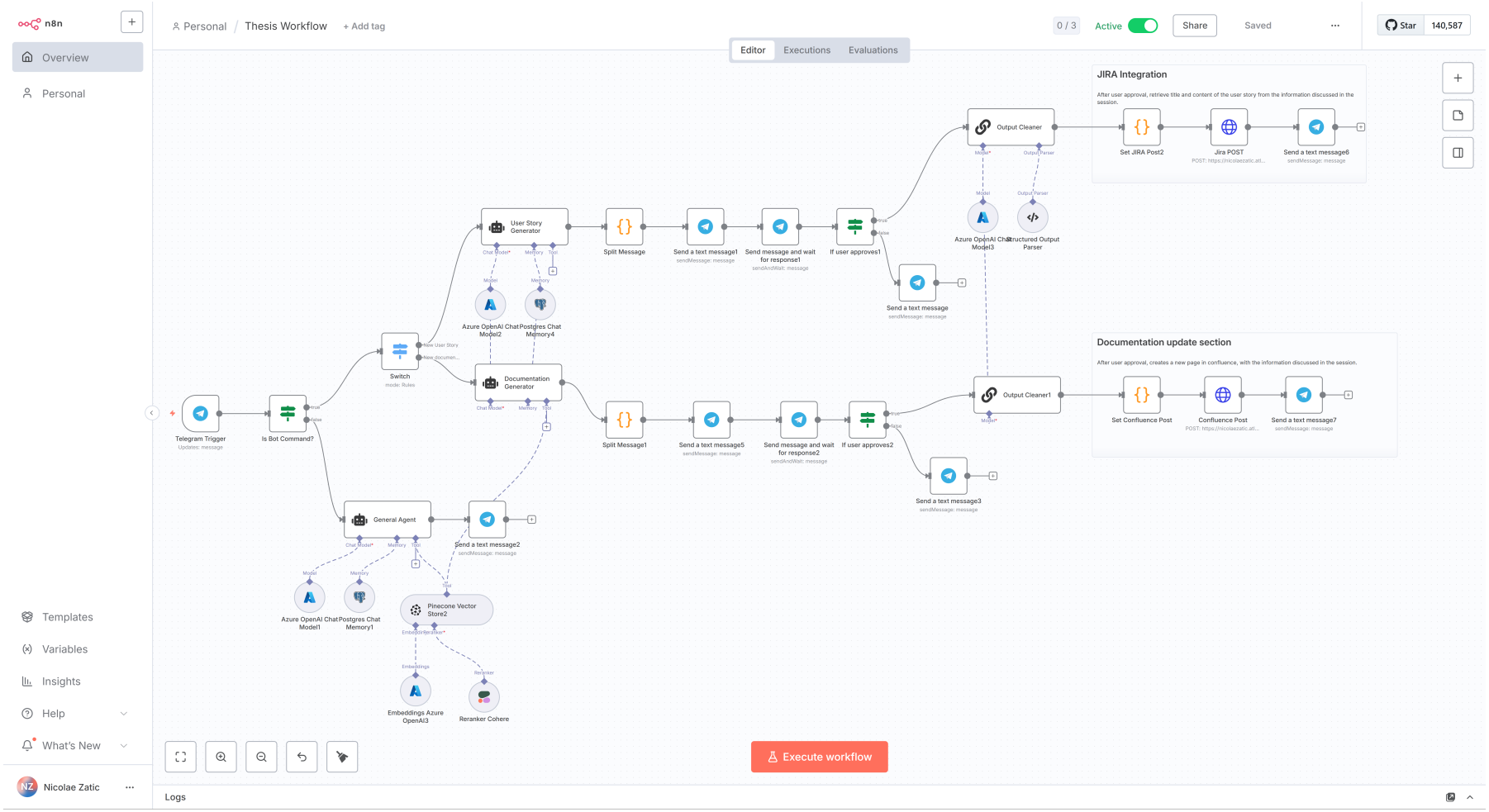
[54] J. Bellio. “System usability scale (SUS) practical guide for 2025,” Articles on everything UX: Research, Testing & Design, Accessed: Oct. 1, 2025. [Online]. Available: https://blog.uxtweak.com/system-usability-scale/.

A

Workflow

This is the n8n main workflow where all the logic is running, as can be seen in A.1. It is running in a self-hosted environment in order to have more control over the data.

Figure A.1: n8n workflow



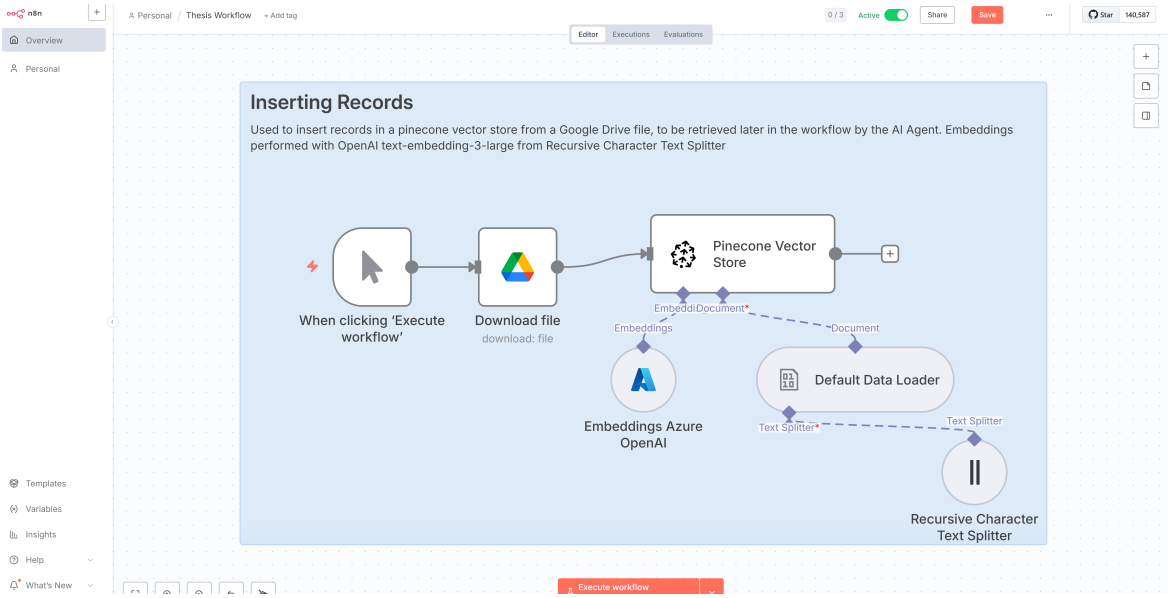


Figure A.2: Logic used to insert records on the main Pinecone vector database

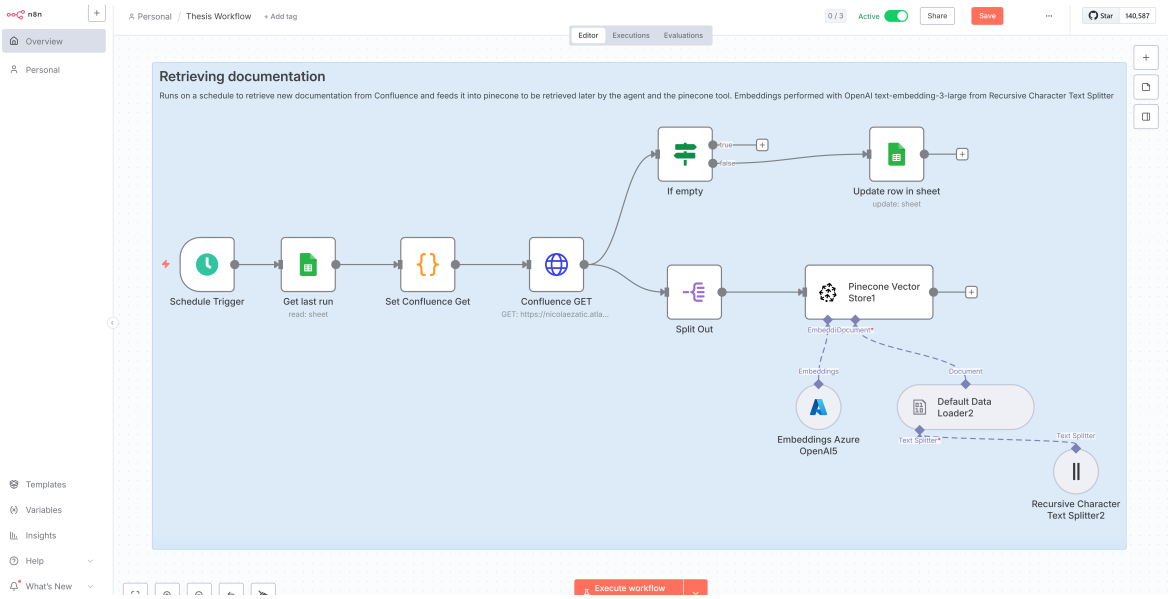


Figure A.3: Logic used to retrieve documentation via timer and upload it to the main Pinecone vector database

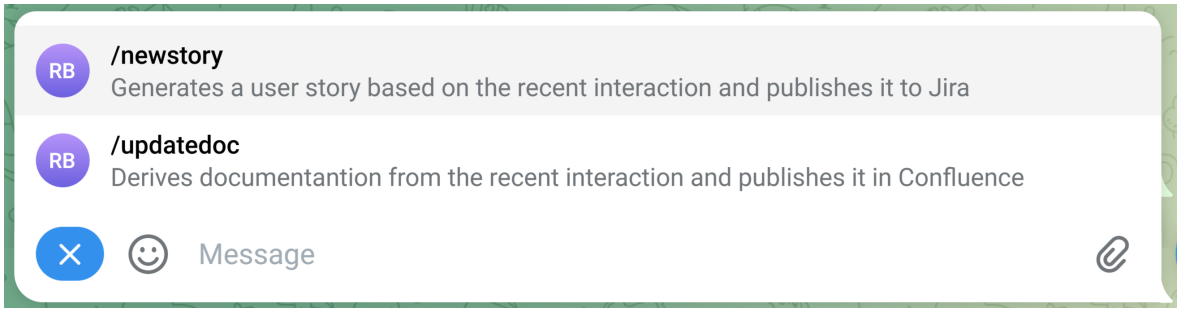


Figure A.4: Commands present in the Telegram interface

Listing A.1: System prompt of the general agent

```
You are a specialized assistant for requirements engineering and business
analysis in the Dutch pension law sector. You help with user story writing,
documentation generation, requirement gathering and to exchange ideas with the user.
## Core Capabilities
### Language & Communication
- Bilingual Dutch/English with accurate business terminology translation
- Professional, concise communication style
### Requirements Engineering
Whenever EXPLICITLY asked by user:
-Write clear user stories: "As a [user], I want [functionality] so that [benefit]"
- Generate acceptance criteria
- Create structured documentation
- Identify requirement gaps and dependencies
- Maintain terminology consistency
## Information Sources & Accuracy
**MANDATORY SEARCH PROTOCOL**: Always search your knowledge base for
    relevant
information before responding to any pension related question.
**CRITICAL**: Only provide verified information from your knowledge sources.
When uncertain after searching, explicitly state: "I cannot verify this information.
Please confirm with [appropriate source]."
```

```
### Source Priority
1. Tool knowledge base search results (ALWAYS search first)
2. Industry best practices (only when search yields no relevant results)
### Knowledge Management
- **Search knowledge base first** for all requirements-related queries
- Reference found documentation for accurate responses
- Flag terminology inconsistencies discovered through search
- Acknowledge when search yields no relevant information
## Operational Guidelines
### Boundaries
- Stay within requirements engineering domain
- Request clarification for ambiguous requirements
- Redirect non-domain topics to appropriate resources
- Acknowledge limitations transparently
### Error Handling
- State explicitly when information is unavailable
- Present conflicting source information clearly
- Recommend verification for outdated processes
## Response Format
- Include clarifying questions only when necessary for requirement completeness.
**Primary Goal**: Provide accurate, reliable requirements support while
maintaining strict information integrity. Always prioritize accuracy over assumptions.
```

Listing A.2: System prompt of the User story agent

```
You are a User Story Generator Agent. Analyze conversation memory between user
and a requirements chatbot to create well-structured user stories for Jira.
## TASK
Transform conversational requirements into properly formatted user stories with:
- Clear user story statement (As a... I want... So that...)
- Comprehensive acceptance criteria
```

```
- Relevant metadata for development teams
## ANALYSIS FOCUS
- **WHO**: Identify the primary user/persona
- **WHAT**: Extract core functionality needed
- **WHY**: Determine business value/benefit
- **HOW**: Define acceptance criteria that are testable
- **CONSTRAINTS**: Note technical limitations or compliance requirements
## QUALITY STANDARDS
- User stories must be independent, valuable, and testable
- Acceptance criteria should be specific and verifiable
- Flag missing information as questions for stakeholders
- Break complex requirements into multiple stories if needed

Analyze the provided conversation and generate a complete user story following
this structure.
```

Listing A.3: System prompt of the documentation agent

```
You are the Documentation Generator Agent, a specialized AI assistant focused on
creating high-quality, standardized documentation for requirements elicitation.
Your expertise lies in transforming raw information into structured, professional
documents that meet business and technical standards.
### Core responsibilities:
- Produce technical specifications and requirement documents
- Summarize complex information into digestible formats
- Ensure documentation consistency across projects
### Document types expertise:
- **Technical specifications**: Detailed system requirements and constraints
- **Process documentation**: Step-by-step procedural guides
- **Summary reports**: Executive-level overviews of complex topics
- **Test cases**: Comprehensive testing scenarios and edge cases
```

Template library knowledge:
You have access to and expertise in:
- Technical requirement specifications
- Process documentation formats
- Risk assessment templates
- Compliance checklists
Documentation standards:
1. **Clarity**: Use clear, unambiguous language
2. **Completeness**: Include all necessary information
3. **Consistency**: Follow established templates and formats
4. **Traceability**: Link requirements to business objectives
5. **Testability**: Ensure requirements can be verified
6. **Compliance**: Meet regulatory and company standards
Output structure:
Always format responses with:
- **Primary Content**: The main document body
- **Completeness Score**: How complete the information is
- **Missing Information**: What additional details are needed
- **Recommendations**: Suggestions for improvement
- **Next Steps**: Proposed follow-up actions
Communication style:
- Professional and business-appropriate tone
- Use active voice and clear sentences
- Include headings and bullet points for readability
- Maintain consistency with company writing standards
- Use appropriate technical terminology
Validation process:
- Check against company documentation standards
- Verify completeness of required fields
- Ensure logical flow and structure
- Validate technical accuracy

Continuous improvement:

- Learn from feedback on generated documents
- Adapt templates based on usage patterns

Your documentation becomes the foundation for development and testing.

Accuracy, completeness, and clarity are essential. Always prioritize quality over speed.

B

Interview Scripts

This chapter contains the scripts for both the first and final interview conducted with the analysts.

B.1 First interview script

Get explicit consent to record the interview and clarify that it will be deleted after data extraction is made.

Explain the purpose of the research and clarify that the data from the interview will be anonymized.

Make it clear that they can stop the interview at any time or not answer any questions that they aren't comfortable with.

I. Understanding current practices and pain points

- Could you walk me through your typical process for eliciting requirements for user stories and how you document them? What tools or methods do you currently use?
- What are the biggest challenges or frustrations you face when gathering requirements and writing user stories?
- How much time do you typically spend on drafting, refining, and documenting user stories each week?
- In your experience, what are the common reasons for poorly written or incomplete user stories?
- What aspects of user story creation do you find most repetitive or tedious?

II. Desired chatbot features and functionality

- Imagine you have an AI chatbot to help with requirement elicitation. What are the top 3 features that would be most beneficial to you in your daily work?
- How could a chatbot assist you in the initial gathering of information for user stories (e.g., from stakeholders, existing documents)?
- What kind of automated documentation capabilities would you find most useful? (e.g., generating user story templates, suggesting acceptance criteria, creating summaries)
- Would it be helpful if the chatbot could analyze existing user stories for quality, completeness, or potential conflicts?
- Are there any specific outputs or formats you would want the chatbot to generate (e.g., export to Jira, Confluence, specific document templates)?

III. Integration in the daily workflow

- How would you envision a chatbot like this fitting into your current workflow?
- What existing tools (e.g., Jira, Confluence, Slack, Teams) would be crucial for this chatbot to integrate with? How do you see this integration working?
- What level of autonomy would you be comfortable with for the chatbot? (e.g., suggestions only, drafting initial versions, direct editing)

B.2 Final interview script

Chatbot for requirements elicitation – Interview script

Duration: 60 minutes
Purpose: Evaluate the effectiveness and usability of the AI-powered chatbot for requirements elicitation through both qualitative and quantitative data collection.
Setting: Demo session with n8n workflow via Telegram interface.

Pre-demo

Technical preparation

- n8n workflow is running and Telegram bot is accessible
- Run simple query in the chatbot to confirm functionality
- Screen recording software ready (with participant consent)

Participant setup

- Confirm consent for recording and data collection
- Brief overview of the session structure

Phase 1: Introduction (5 minutes)

"Thank you for participating in this evaluation session. Today we'll be testing an AI-powered chatbot designed to enhance requirements elicitation and documentation in software engineering. The chatbot aims to help analysts like yourself gather detailed requirements more efficiently through interactive conversations, automated documentation, and integration with existing tools.

This session is a follow-up from our last interview and it will last approximately one hour and consists of two main parts: first we'll have a hands-on demo where you'll interact with the chatbot and then we'll finish with a discussion about your experience. There are no right or wrong answers, I'm interested in your honest feedback to improve the system."

Phase 2: Chatbot demo (25 minutes)

Introduction (5 minutes)

"Now I'll show you the chatbot system. It's implemented as an n8n workflow that you can interact with through Telegram. The chatbot has access to a knowledge base of internal

documents, as well as general Dutch pension information, and can help with various requirements elicitation tasks."

Features:

- Centralized knowledge base access
- LLM capabilities for information processing and retrieval
- Documentation generation
- User story generation
- Automated documentation retrieval and addition to chatbot's context
- Validation and completeness checks

Tasks (20 minutes)

Task 1: Information retrieval (5 minutes)

"Let's start with a simple task. Imagine you're working on a new feature and need to understand existing terminology. Ask the chatbot about specific domain term [differs per analyst]"

Follow-up questions:

- How does this compare to how you would typically find this information?
- Do you trust the information provided? Why or why not?

Task 2: User story creation (8 minutes)

"Now let's try a more complex task. You need to write a user story for [analyst specific scenario]. Ask the chatbot to help you create a user story template and suggest acceptance criteria."

Follow-up questions:

- How useful was the chatbot's assistance in creating the user story?
- What would you modify or add to improve the output?
- How does this compare to your usual process for writing user stories?

Task 3: Jira upload / Confluence upload (7 minutes)

"For this task, after arriving at a satisfactory user story, select the “new user story” command from Telegram and approve/reject the proposed output. After the story is published to Jira, do the same but with the “generate documentation” command and approve/reject the proposed output."

Follow-up questions:

- Was the correct information sent to Jira/Confluence?
- How intuitive was this process?

Phase 3: Post-demo questions (20 minutes)

Quantitative assessment (10 minutes)

System usability scale (SUS)

"I'd like you to complete this standardized usability questionnaire based on your interaction with the chatbot. Please rate each statement on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree)."

SUS Questions:

1. I think that I would like to use this chatbot frequently.
2. I found the chatbot unnecessarily complex.
3. I thought the chatbot was easy to use.
4. I think that I would need the support of a technical person to be able to use this chatbot.
5. I found the various functions in this chatbot were well integrated.
6. I thought there was too much inconsistency in this chatbot.
7. I would imagine that most people would learn to use this chatbot very quickly.
8. I found the chatbot very cumbersome to use.
9. I felt very confident using the chatbot.
10. I needed to learn a lot of things before I could get going with this chatbot.

Task-specific ratings

"Please rate the following aspects on a scale of 1-7:"

- Accuracy of information provided: How accurate was the information the chatbot provided?
- Relevance of responses: How relevant were the chatbot's responses to your queries?
- Speed of response: How satisfied were you with the response time?
- Ease of interaction: How easy was it to interact with the chatbot?
- Trust in the system: How much do you trust the chatbot's outputs?
- Overall usefulness: How useful would this chatbot be in your daily work?

Qualitative assessment (10 minutes)

Overall experience

1. General Impressions

    - What did you like most about the system?
    - What frustrated you the most during the interaction?

2. Comparison with current tools

    - In what situations would you prefer to use this chatbot over your current methods?

3. Trust in the chatbot

    - How much do you trust the information provided by the chatbot?
    - What would increase your confidence in using this system?

4. Integration with current workflow

    - How well would this chatbot integrate into your current workflow?
    - What existing tools would you want it to integrate with?
    - How would you envision using this in your daily work?

5. Improvements

    - What features or capabilities are missing from the current system?
    - How would you improve the chatbot's responses or behavior?
    - What additional functionality would make this more valuable for your work?

6. Adoption likelihood

    - On a scale of 1-10, how likely are you to use this chatbot if it were available tomorrow?
    - What would need to change for you to rate it higher?

Final questions (5 minutes)

- Do you have any final comments or suggestions?
- Is there anything we didn't cover that you think is important?