University of Twente
Faculty of Behavioural, Management and Social Sciences (BMS)
BSc Industrial Engineering and Management

Designing for Reduced Expert Dependency in SME Manufacturing

Design of a Retrieval-Augmented Generation System for Troubleshooting Knowledge

Bachelor Thesis
Module 12 – Bachelor Thesis Project

T.C. Stoppelenburg (Tim)
Student Number: 2600994

In collaboration with
Huima Drive Solutions, Enschede, the Netherlands

First supervisor: Dr. M.R. Machado (Marcos)
Second supervisor: Dr. Hao Chen
Company supervisor: C. Veldhuis (Clemens)

Enschede, July 2, 2026

UNIVERSITY OF TWENTE.

Abstract

Huima Drive Solutions is a small manufacturer with more than ninety years of accumulated production knowledge. Much of the critical troubleshooting knowledge is not documented but held by one senior expert, Willy, who is approaching retirement. When Willy is unavailable, operators either search fragmented handwritten documentation or wait until he returns, which costs production time and makes the company dependent on a single person. The core problem is that there is no structured method for capturing and organizing this knowledge. This thesis addresses that problem by developing and validating a knowledge capture and retrieval system for Huima.

The research follows the Managerial Problem-Solving Method (Heerkens & van Winden, 2021) across four phases: a current-state analysis, a requirement specification, a literature review, and the design and validation of a prototype. Data was collected through semi-structured interviews, workplace observations, document analysis and a questionnaire among operators. The current-state analysis confirmed a strong dependency on Willy and documentation that is hard to search. The literature review did not identify a readily suitable existing system that covers the full capture-structure-retrieve chain at low effort for a manufacturing SME, which motivated the chosen direction.

The resulting prototype is a Retrieval-Augmented Generation (RAG) system. Operators capture knowledge by typing, photographing handwritten notes or speaking, and they retrieve it through a natural-language search interface. Spoken and photographed input is checked by a human before it becomes searchable, so unverified knowledge cannot enter the knowledge base. Retrieval and answer generation run locally inside the company, external cloud services are used only to transcribe OCR/read and structure operator-created input, never to store the knowledge base. Keeping proprietary knowledge local is a hard requirement set by Huima. The system was validated functionally against the must-have requirements using real troubleshooting cases, covering the capture chain, retrieval quality, retrieval time and the system's ability to abstain when no answer is available. The validation showed that captured knowledge is transcribed and retrieved correctly and quickly on the tested cases, and that the system abstains rather than inventing an answer when knowledge is absent. The prototype therefore demonstrated a feasible path towards reduced expert dependency, not an achieved solution, since realizing that reduction depends on populating the knowledge base in daily use.

Keywords: tacit knowledge, knowledge capture, retrieval-augmented generation, manufacturing SME, expert dependency, troubleshooting, knowledge management

Management summary

This study was carried out at Huima Drive Solutions to address a problem that management already recognizes: too much of the company's troubleshooting knowledge depends on one person. When Willy is on the floor, operators get a fast and reliable answer. When he is not, they lose time searching handwritten notes and manuals that are incomplete and hard to navigate, and some problems stay unsolved until he returns. With Willy approaching retirement, this dependency is a real risk for daily production.

What was Investigated

The project looked at how knowledge is currently created, stored and retrieved at Huima, what a good solution would need to do and which solution fits the company best. This was based on interviews, observations on the work floor, a questionnaire among operators and analysis of the existing documentation. The findings confirmed the dependency: most operators ask Willy regularly, the documentation is often not up to date or hard to search, and a single problem can cost from ten minutes up to much longer when he is absent.

The solution

The result is a working prototype, the Huima Kennisapp. Operators and Willy can add knowledge by typing, by taking a photo of handwritten notes, or by speaking, which is the easiest way for an expert like Willy to explain a problem. They find knowledge back by asking a question in normal language, and the system returns the most relevant past cases. Two design choices protect the company: spoken and photographed input is checked by a person before it becomes searchable, so wrong information does not silently enter the system. Second, the company's own knowledge stays on Huima's own network, only transcription and reading of notes use external services, and only for input that the operator created. The system gives advice based on past cases, it does not replace the operator's own judgement. The prototype was tested on representative troubleshooting cases and works as intended. The actual reduction in dependency on Willy depends on filling the system with knowledge, which is the first recommendation below.

Recommendations

The most important next step is to fill the system with knowledge, starting with the most common recurring problems rather than trying to capture everything at once. A practical start is to record the questions operators actually ask Willy and to capture those first, ideally while Willy is still present and can act as the person who approves new entries. Huima should also appoint an owner for the knowledge base, keep the rule that proprietary data stays local and address a small number of technical points (such as stronger login security and a database cleanup) before the system is relied on in everyday production.

Acknowledgements

I would like to thank my first supervisor, Marcos Machado, for the guidance and critical feedback throughout this research. The questions raised during our meetings pushed me to sharpen the work and to be honest about what the prototype does and does not show. I would also like to thank my second supervisor Hao Chen for his critical feedback.

I am grateful to Huima Drive Solutions for the opportunity to carry out this thesis within the company and for the open access to the work floor and its documentation.

A special thanks goes to Willy, whose experience and willingness to share his knowledge and stories form the heart of this research. Without his time and openness, building this research would not have been possible.

Finally, I would like to thank Clemens and Jan-Erik for their cooperation throughout the project and for the trust and decisions that made it possible to develop and test the system within Huima.

AI statement

Throughout this research, large language models were used to support the formulation, structuring and critical reviews of the written text, and AI-based visualization tools were used to produce several of the figures. The tools used include Claude, ChatGPT and Gemini.

The prototype codebase was developed with extensive AI assistance: the majority of the implementation code – frontend, backend, database interaction, prompt logic and integration logic – was AI-generated. The author defined the problem, requirements, architecture, data model, validation logic and all design decisions, and specified, reviewed, adapted, tested and integrated all AI-generated code into the final prototype.

AI was used as a supporting instrument instead of a substitute for the author's own responsibility. For the written text, all AI-supported passages were verified, rewritten and integrated by the author. For the prototype, all AI-generated code was tested against the functional requirements and validation cases reported in the thesis. The author takes full responsibility for the content, argumentation, implementation choices, validation and correctness of the final thesis and prototype.

Contents

Abstract 1
Management summary 2
What was Investigated 2
The solution 2
Recommendations 2
Acknowledgements 3
AI statement 3
1. Introduction 8
1.1 Problem Identification 8
1.1.1 Problem Description and Action Problem 8
1.1.2 Problem Cluster and Core Problem 9
1.1.3 Norm and Reality 9
1.2 Problem-Solving Approach 10
1.3 Research Questions 10
1.4 Scope of the Research 11
1.5 Validity and Reliability 12
1.5.1 Ethical and data-handling considerations 12
1.6 Deliverables 12
2. Theoretical Framework 14
2.1 Tacit Knowledge: Why Expert Knowledge Is Hard to Transfer 14
2.2 Knowledge Conversion: Why Informal Transfer is Not Enough 15
2.3 Conditions for Successful Knowledge Elicitation 15
2.4 Retrieval Augmented Generation as a Technical Response 16
2.5 Key Concepts and Variables 16
2.6 Synthesis 17
3. Current-State Analysis 19
3.1 How Knowledge Is Created, Stored and Retrieved 19
3.2 Scope of the Dependency 22
3.3 Expert Dependency: Questionnaire Findings 22
3.4 Summary: Current-State Assessment 24
4. Literature Review 26
4.1 Capturing Tacit Knowledge 26
4.2 Structuring Captured Knowledge 27
4.3 Retrieving Tacit Knowledge 28

4.4 Synthesis and Gaps 28
5. Solution Design 31
5.1 Requirement Specification 31
5.1.1 How the Requirements Were Gathered 31
5.1.2 Must have 31
5.1.3 Should Have 33
5.1.4 Could Have 33
5.1.5 Won't Have 33
5.1.6 Summary: MoSCoW Prioritization 34
5.2 Solution Exploration 36
5.3 Solution Selection 38
5.4 Prototype Design 40
5.4.1 Design Drivers 40
5.4.2 System Architecture 41
5.4.3 Data Model 42
5.4.4 Knowledge Ingestion and Validation 43
5.4.5 Embedding and Retrieval Design 45
5.4.6 Answer Generation and Safeguards 46
5.4.7 Operational Robustness 47
5.4.8 Limitations of the Prototype Design 47
5.4.9 Summary of Design Choices 47
6. Validation 49
6.1 What This Validation Does and Does Not Prove 49
6.2 Validation Method 50
6.3 Capture Validation 51
6.3.1 OCR Validation 51
6.4 Retrieval Validation 52
6.4.1 Test Set 54
6.4.2 Results 55
6.5 Voice Validation 57
6.5.1 Long Recording 58
6.5.2 What the boundary case shows 58
6.6 Threats To Validation 59
6.7 Overall Validation Conclusion 59
6.7.1 Generalizability 59

6.7.2 Requirements Against Validation 60
7. Conclusion 62
7.1 Limitations 63
7.2 Implementation Advice 63
7.2.1 Populating the Knowledge Base 63
7.2.2 Maintenance and Governance 64
7.2.3 Technical 64
7.2.4 Future Capture Methods and Model Choices 64
Appendix A: Empirical Data Collection 66
A.1 Document Analysis 66
A.2 Observation Examples 66
A.2.1 Troubleshooting During a Machine Error 66
A.2.2 Troubleshooting Error Code 220 67
A.3 Willy's Absence Period 67
A.4 Questionnaire Findings 67
A.4.1 Questionnaire Items 67
A.4.2 Open-Ended Responses 68
A.4.3: Questionnaire Findings (Dutch) 69
A.4.4: Questionnaire Findings (English) 70
Appendix B: Database Design and Relationships 72
B.1 Main Tables and Fields 72
B.2 How the Tables are Linked 73
B.3 Status Model for Voice Notes 73
B.4 Lifecycle of a Combined Report 73
B.5 Media Storage 73
B.6 Normalization Choices and Future Work 74
Appendix C: Architecture and API Overview 75
C.1 API Overview 75
C.2 Trust Boundary and Secrets 76
C.3 Error Handling and Robustness 77
Appendix D: Prompt Design and Validation Logic 77
D.1 OCR Prompt 77
D.2 Structuring Prompt 78
D.3 Transcription Prompt 79
D.4 Answer-Generation Prompt 79

D.5 Clarification Prompt 80
D.6 Human Validation Logic 80
Appendix E: Embedding, Retrieval and Threshold Design 81
E.1 Composite Embedding Text 81
E.2 Embedding Model Selection 81
E.3 Retrieval Query 81
E.4 Thresholds 82
E.5 Error-Code Route 82
E.6 Separation of Retrieval and Generation 83
Appendix F: Transcription Design and Fallback Logic 83
F.1 Pipeline Overview 83
F.2 Configuration of the Local Engine 83
F.3 Engine Recording and Comparison 84
F.4 Crash Recovery and Idempotency 84
F.5 Voice Search 84
Appendix G: Technical Glossary 85
Reference List 86

1. Introduction

This chapter introduces Huima Drive Solutions and the problem that forms the basis of this thesis. Section 1.1 identifies the problem through a problem cluster, which leads to the identification of the core problem. Section 1.2 describes the problem-solving approach adopted in this research. Section 1.3 presents the central research question and the sub-questions that guide the study. The scope and research are discussed in Section 1.4, followed by validity and reliability in section 1.5. This chapter concludes with an overview of the main deliverables in Section 1.6.

1.1 Problem Identification

1.1.1 Problem Description and Action Problem

Huima Drive Solutions[1] is a manufacturer of drive systems with a long operational history. Over the years, a large amount of production troubleshooting knowledge has been built up within the company. This knowledge is valuable because it supports daily production, problem solving and machine-specific maintenance decision making. However, a significant part of this knowledge is not formally documented in a structured way. Instead, it exists in handwritten notes, old manuals, personal experience and tacit knowledge of experienced employees.

This creates a knowledge management problem. In practice, employees often rely on the experience of one senior employee, Willy[2], when they face production or troubleshooting questions. These questions can range from identifying the right tool or setting to be used in a specific machine to diagnosing more complex operational issues. When Willy is available, employees often obtain an answer quickly. When he is not available, employees have to search through fragmented sources such as handwritten books, manuals, logs or notes. These sources are difficult to navigate and are not consistently structured.

The theoretical basis for this problem lies in the concept of tacit knowledge. Polanyi (1966) argues that people know more than they can explicitly tell. This is visible in Huima's context: experienced employees often recognize machine behavior, sounds, errors or irregularities based on years of experience, but this knowledge is difficult to write down in a conventional manual. Suhonen & Kankaanpää (2025) found that across 163 Finnish manufacturing companies, most knowledge transfer efforts remain informal and unstructured, suggesting that simply asking experts to document what they know is unlikely to produce reliable results. A more structured approach is therefore needed to capture practical troubleshooting knowledge in a way that is simple enough for the production floor and useful for later retrieval.

Heerkens & van Winden (2021) define an action problem as the discrepancy between norm and reality as perceived by the problem owner. For Huima Drive Solutions, in the context described above, the following action problem is identified:

Operators at Huima often cannot resolve recurring production issues without direct consultation of one senior expert.

[1] Huima Drive Solutions, Enschede, the Netherlands [https://huima.nl/]
[2] The use of Willy's first name has been approved by the company.

1.1.2 Problem Cluster and Core Problem

The action problem does not arise from a single cause but from a cluster of interconnected problems. with clear cause-and-effect relationships, as shown in figure 1.

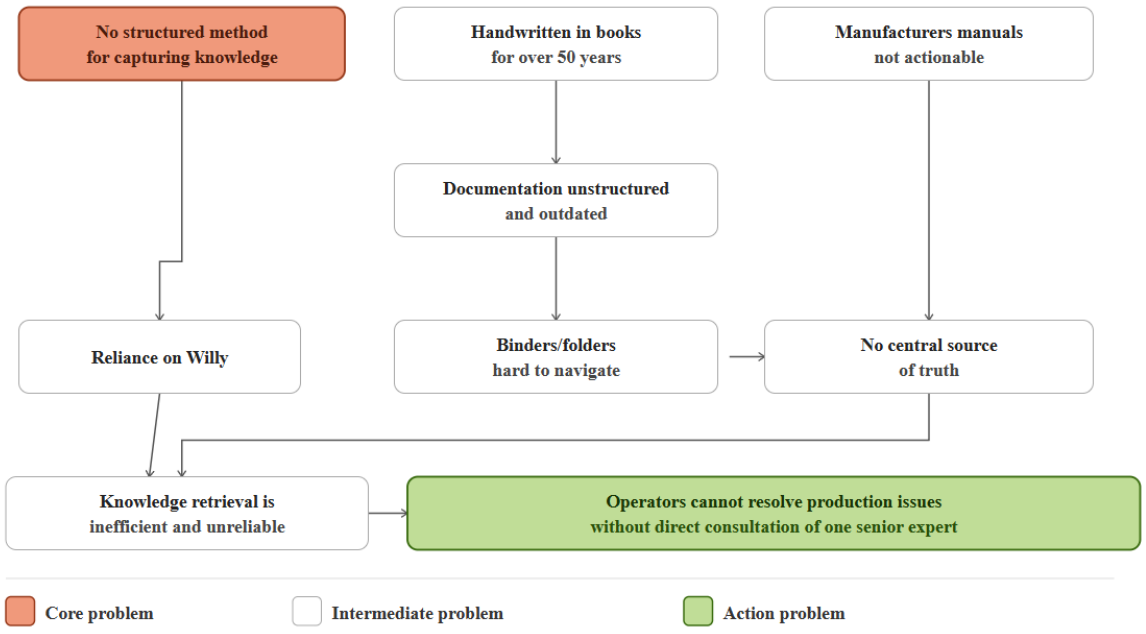


Figure 1: Problem Cluster

Following the criteria of Heerkens & van Winden (2021) the core problem is the deepest root cause in the causal chain – the problem with only outgoing arrows and no incoming causes. Three root causes are visible in the cluster. the two external ones – handwritten notes accumulated over fifty years and non-actionable manufacturer manuals – fall outside of Huima's direct control and cannot be undone. The third root cause, the absence of a structured method for capturing knowledge is directly addressable and most explanatory of the action problem. It is therefore selected as the core problem:

There is no structured method for capturing and organizing critical production knowledge within Huima Drive Solutions.

1.1.3 Norm and Reality

To assess the severity of the action problem, this research compares the current situation at Huima to a desired situation across three dimensions defined for this research: knowledge coverage, standardization and expert dependency.

The normative situation, critical troubleshooting and operational knowledge is systematically documented, stored in a structured and searchable format and retrievable by any operator within approximately the same time it would take to consult Willy directly. Knowledge should be recorded consistently, kept up to date and accessible regardless of which employee is present. Dependency on any single individual should be minimal – no person's absence should create a knowledge gap that disrupts operations.

The current situation at Huima deviates from this norm across all three dimensions. In terms of coverage, a large portion of operational knowledge exists only in the minds of experienced employees and has never been documented. In terms of standardization, existing documentation is spread across handwritten notes, personal binders and manufacturer manuals with no unified structure or version

control. In terms of expert dependency, many troubleshooting situations cannot be resolved without direct consultation of Willy.

The measurable gap between norm and reality is most visible in retrieval speed and the frequency of expert consultation. Initial site visits and informal conversations with operators and management suggest that finding a solution through existing documentation is slow and unreliable while consulting Willy is fast and reliable. The norm – that documented knowledge should approximate the speed and reliability of expert consultation – is not met.

1.2 Problem-Solving Approach

This research aims to develop a structured approach for capturing, organizing and retrieving critical operational knowledge. The research is guided by the Managerial Problem-Solving Method (MPSM), as described by (Heerkens & van Winden, 2021). While the seven phases are presented sequentially, the prototype development began in parallel with fieldwork due to time constraints of this ten-week project. Additionally, this research created four phases that capture the total framework of the MPSM.

Phase 1 focuses on understanding the current situation. Through workplace observations, walkthrough cases, semi-structured interviews, questionnaires and document analysis, this phase maps how knowledge is currently created, stored and retrieved within Huima. It produces a current-state analysis and knowledge map (MPSM phase 3: analyzing the problem).

Phase 2 identifies the main bottlenecks and translates them into requirements for a solution. Based on the current-state analysis and stakeholder input, this phase produces a requirements specification using MoSCoW analysis[3] (Must have, Should have, Could have, Won't have), defining what a knowledge capture and retrieval system must satisfy in functional, technical and organizational terms (MPSM phases 3 and 4: analyzing the problem and generating requirements for solutions).

Phase 3 develops a theoretically grounded knowledge capture framework and selects a solution concept. Based on literature review and findings from phases one and two, existing approaches for capturing, structuring and retrieving tacit operational knowledge are compared and evaluated. This phase results in a motivated solution design (MPSM phases 4 and 5: generating solutions and choosing a solution).

Phase 4 consists of building and validating the prototype. This research evaluates the proposed system against the requirements specification using real troubleshooting cases at Huima, assessing whether it improves knowledge accessibility, reduces retrieval time and aims to support reduced dependency on Willy (MPSM phase 7: evaluating the solution; phase 6, full implementation, is out of scope and addressed as implementation advice in Section 7.2).

1.3 Research Questions

Based on the core problem identified in Section 1.1 and the problem-solving approach described in the Section 1.2, this study formulates the following central research question:

> *How can a structured knowledge capture and retrieval system be developed to help reduce expert dependency and improve knowledge accessibility at Huima Drive Solutions?*

To answer this central question, five sub-questions are formulated.

[3] MoSCoW is a requirements prioritization method, used to support scope management by distinguishing essential requirements from less critical features

I. *What critical production and troubleshooting knowledge currently exists at Huima, and how is it currently created, stored and retrieved?*

This question maps the existing knowledge landscape. It is answered through semi-structured interviews, workplace observation and document analysis. The description of the current situation helps define the improvements that can be implemented for a better system of documenting and retrieving knowledge.

II. *What are the main bottlenecks in the current knowledge retrieval process, and what functional, technical and organizational requirements must a solution satisfy?*

This question identifies the gaps and inefficiencies and translates them into requirements. It is answered through stakeholder interviews, questionnaires and MoSCoW analysis.

III. *What approaches and technologies exist in literature for capturing, structuring and retrieving tacit knowledge in manufacturing environments?*

This question provides the theoretical and empirical foundation for the solution design. It is answered through a literature review.

IV. *Which solution concept best fits Huima's operational context, requirements and constraints?*

This question evaluates candidate solutions against the requirements from sub-question 2 and the literature findings from sub-question 3. It is answered through stakeholder evaluation.

V. *To what extent does the developed system capture expert knowledge and retrieve it correctly and reliably on representative troubleshooting cases?*

This question is answered by validating the mechanical and functional system in accordance with the requirements set out in the requirements specification in Chapter 5.1.

1.4 Scope of the Research

The scope of this research is limited: the goal is not to capture the full breadth of Willy's expertise during this project, but to develop and validate a system capable of doing so, demonstrated through a representative selection of knowledge entries: documented situations in which Willy's knowledge would otherwise be required.

A significant technical constraint shapes the entire system design: the stored proprietary Huima knowledge base – including error codes, machine-specific manuals and operational records – must stay inside Huima's internal environment and may not be hosted on external servers. The only data that may leave is operator-created raw input (photos and voice notes). This constraint directly limits the range of feasible technical solutions and is treated as a hard design requirement throughout the research.

1.5 Validity and Reliability

This study defines validity as the extent to which data collection methods accurately measure what they intend to measure and reliability as the extent to which data collection techniques yield consistent findings if repeated under similar conditions (Saunders et al., 2023, p. 215-216).

To support reliability, interviews are scheduled at moments without production pressure during the working day and conducted individually in a private setting, reducing situational pressure and social desirability in responses (Saunders et al., 2023). Member checking – sharing key findings with respondents to verify whether identified themes accurately reflect their situation (Lincoln & Guba, 1985) – reduces researcher bias. The literature review was conducted as a targeted narrative review, using academic and practice-oriented sources to compare approaches for capturing, structuring and retrieving troubleshooting knowledge in manufacturing contexts.

Internal validity is supported through methodological triangulation (Saunders et al., 2023): observations, interviews and questionnaires address overlapping constructs from different angles. The most relevant threat is the testing effect – operators may behave differently when observed. This is mitigated by supplementing observation with retrospective interviews. Whether the testing effect actually occurred is not analyzed within this research, it remains a residual threat and is suggested as point of attention for future research.

Construct validity is supported by grounding all key constructs in established academic definitions, as described in chapter 2. The risk that proxies such as the frequency of Willy consultation do not fully capture expert dependency is acknowledged: multiple operationalizations are used where possible.

External validity is limited by the single case study design. This is acceptable given the primary aim of solving a concrete problem at Huima. To support transferability, this research describes the context, design choices and findings in sufficient detail for future replication in similar setting.

1.5.1 Ethical and data-handling considerations

This research involved interviews, workplace observations, questionnaire responses and prototype testing in a company setting. Participation in interviews and the questionnaire was voluntary, and the data were used only for the purpose of understanding Huima's knowledge-capture problem and validating the prototype. The use of Willy's first name was approved by the company, but personal data were otherwise minimized where possible. Because the prototype uses external AI services for OCR, structuring and transcription of newly created operator input, the data boundary was explicitly defined: the accumulated Huima knowledge base remains local, while only operator-created raw input may be processed externally under the approved design constraint. This distinction is important because the system handles operational company knowledge and because unvalidated AI output should not enter the searchable knowledge base without human review.

1.6 Deliverables

This research produces six deliverables, each linked to a phase of the problem-solving approach.

- **Current-state analysis and knowledge map** (phase 1). An overview of how troubleshooting and operational knowledge is currently stored, retrieved and applied within Huima, including the main bottlenecks and an inventory of which knowledge exists and which parts depend primarily on Willy.
- **Requirements specification** (phase 2). A structured analysis of the functional, technical and organizational requirements a knowledge capture and retrieval solution must satisfy, derived

from the current-state analysis and stakeholder input, and organized using MoSCoW prioritization.

- **Knowledge capture framework** (phase 3). A structured, theoretically grounded approach for eliciting and documenting tacit and explicit knowledge from experienced employees, describing which methods are used, why they are chosen and how captured knowledge is structured for storage and retrieval.
- **Functional prototype** (phase 4). A working knowledge capture and retrieval system with a natural-language search interface and an interface for adding and updating knowledge entries, demonstrated on a representative selection of troubleshooting cases rather than the full breadth of Huima's knowledge.
- **Validation report** (phase 4). A functional validation of the prototype, reporting capture accuracy (transcription and OCR word-error rate), retrieval accuracy and retrieval time on representative troubleshooting cases.
- **Implementation advice** (phase 4). Practical recommendations on how the system can be maintained, updated and expanded after this project ends, including guidance on knowledge base population and long-term governance.

2. Theoretical Framework

This chapter introduces the theoretical foundations that explain why the problem at Huima exists and what conditions a solution must meet. Three complementary perspectives are discussed, followed by the technical concept that builds on them. First, Polanyi (1966) theory of tacit knowledge explains why Willy's expertise cannot simply be written down through conventional documentation. Second, Nonaka et al.'s (2000) SECI (Socialization, Externalization, Combination, Internalization) model explains how knowledge moves between individuals and organizations and where that process breaks down in practice. Third, the MOA (Motivation-Opportunity-Ability) framework of Gavrilova & Andreeva (2012) identifies the conditions under which knowledge elicitation succeeds or fails. Finally, Section 2.4 introduces retrieval-augmented generation (RAG) as the technical architecture that responds to these three perspectives.

2.1 Tacit Knowledge: Why Expert Knowledge Is Hard to Transfer

The most central concept in this research is tacit knowledge, as introduced by the philosopher Michael Polanyi in The Tacit Dimension (1966). Polanyi's central argument is that human knowing cannot be fully reduced to explicit statements or written instructions, captured in his well-known phrase: "we can know more than we can tell". This deceptively simple observation has significant implications for any organization that depends on experienced employees. It means that expert performance cannot be fully explained by the expert themselves, and therefore cannot be captured through conventional documentation alone.

Polanyi distinguishes between two layers of awareness that are always present in skilled action: subsidiary awareness (background cues and prior experiences) and focal awareness (the resulting diagnosis or action). This mechanism relies on "attending from" the subsidiary cues in order to "attend to" the focal outcome, meaning the background layer remains structurally invisible during expert performance (Polanyi, 1966).

To illustrate, when reading a text, a person attends *from* the individual letters (subsidiary) to attend *to* the meaning of the words (focal). Focusing on the physical shape of a single letter instantly disrupts sentence comprehension. Similarly, an experienced manufacturing operator recognizes a machine fault by attending *from* a tacit combination of sound, vibration and visual appearance *to* the focal diagnosis. Just like the letters in the sentence, these sensory cues are real and consistent, but they function as internalized tools rather than conscious objects of focus.

This distinction between subsidiary and focal awareness is directly applicable to Willy's situation at Huima. His ability to diagnose recurring faults quickly is not the result of consciously recalling documented procedures. It is the result of decades of pattern recognition that has become automatic. When an operator asks Willy what is wrong with a machine, Willy's answer is reliable precisely because it bypasses explicit reasoning and draws on internalized experience. This is also why the answer is difficult to document: the explicit form of the knowledge is not the knowledge itself, but only its surface expression. As Polanyi notes, even when experts attempt to explain knowledge, "this very act of communication displays a knowledge we cannot tell" (Polanyi, 1966, p.5).

For knowledge management research, Polanyi's (1966) framework has two important methodological implications. First, the objective of knowledge capture cannot be the complete codification of an expert's expertise, as some dimensions will inevitably remain tacit even after an intensive elicitation effort. The realistic aim of such initiatives is to make the most frequently needed and repeatable knowledge accessible in a structured form. Second, elicitation methods that rely solely on asking experts to explain what they do are likely to produce incomplete results. consequently, methods that involve direct

observations, demonstrations, and case walkthroughs are required to surface the cues and reasoning steps that verbal explanation alone would leave out (Hoffman et al., 1995).

2.2 Knowledge Conversion: Why Informal Transfer is Not Enough

Whereas Polanyi explains why individual expertise is hard to articulate, Nonaka et al. (2000) explain what happens at the organizational level when knowledge conversion does or does not occur. Their SECI model describes knowledge creation as a continuous cycle of conversion between tacit and explicit form through four modes. Socialization transfers tacit knowledge from one person to another through shared experience, observation and imitation. Externalization converts tacit knowledge into explicit form through articulation, documentation and structuring. Combination integrates existing explicit knowledge from different sources into new explicit knowledge. Internalization converts explicit knowledge back into tacit knowledge through learning by doing and practice.

This model is useful not only as a description of how knowledge moves, but as a diagnostic tool for identifying where organizations fall short. Suhonen & Kankaanpää (2025) applied the SECI model to examine how knowledge transfer is organized in practice across 163 Finnish manufacturing companies and found that socialization and internalization are well-established in the sector, while externalization is consistently underdeveloped. Most companies transfer knowledge informally through master-apprentice arrangements, side-by-side learning and on-the-job training. This works as long as the expert is present and available. It fails the moment the expert is absent, changes role or leaves the organization.

The SECI model therefore positions externalization as the primary organizational bottleneck in manufacturing environments where knowledge retention relies heavily on a single expert. Consequently, for any knowledge management intervention to be effective, it must explicitly focus on bridging the gap between socialization and externalization.

2.3 Conditions for Successful Knowledge Elicitation

Knowing that externalization is the bottleneck does not automatically tell us how to achieve it. Gavrilova & Andreeva (2012) address this question directly through the MOA framework, which states that effective knowledge sharing depends on three conditions being met simultaneously: motivation, opportunity and ability.

Motivation refers to the expert's willingness to share their knowledge. This is not guaranteed even when management actively encourages it. (Suhonen & Kankaanpää, 2025) found that some workers deliberately withhold knowledge as a form of job security, and (Hoerner et al., 2023) note that not all employees are willing to participate in externalization efforts. Beyond active resistance, motivation can also be undermined by poorly designed elicitation processes: if the process is perceived as burdensome or irrelevant to daily work, even a cooperative expert will disengage over time.

Opportunity refers to the organizational and situational conditions that allow knowledge sharing to take place. These include the availability of time, the right physical or social setting and sufficient organizational support. Production environments in particular tend to constrain opportunity, as operational demands routinely compete with time that could otherwise be dedicated to knowledge capture. Even when management recognizes the need for systematic knowledge transfer, the conditions for sustained elicitation are not always present in practice. (Suhonen & Kankaanpää, 2025)

Ability refers to the expert's capacity to articulate what they know. This is the dimension most directly related to Polanyi's insight: because tacit knowledge is embedded in skilled action rather than in conscious reasoning, experts often struggle to describe what they do accurately or completely. (Gavrilova & Andreeva, 2012) argue that different elicitation methods suit different types of knowledge.

Observation, demonstrations and role-based exercises are more effective for tacit knowledge because they allow the researcher to observe behavior directly rather than relying on retrospective self-report. Structured walkthroughs and case-based recall are also effective because they anchor the conversation in concrete past situations rather than abstract generalization. Questionnaires, by contrast, are explicitly identified as unsuitable for tacit knowledge elicitation, because they require respondents to articulate what they know in advance and in isolation, precisely the condition under which tacit knowledge is least accessible.

The MOA framework therefore functions as both a diagnostic and a design tool. As a diagnostic, it helps explain why simply asking experts to document their knowledge produces unreliable results: any one of the three conditions may be absent. As a design principle, it implies that elicitation processes should minimize burden, be embedded in the natural rhythm of work and use methods matched to the type of knowledge being captured.

2.4 Retrieval Augmented Generation as a Technical Response

The three frameworks above (Sections 2.1, 2.2 and 2.3) establish what a knowledge system must achieve conceptually: it must support externalization of tacit knowledge and make the resulting knowledge retrievable in a way that matches how operators describe problems, not how those problems were originally documented. Traditional keyword-based search fails at this because it can only return records that contain the exact words the user typed – an operator who does not know the technical term for a fault will find nothing, even if the solution is stored. Semantic search addresses this by converting both the user's query and all stored records into numerical vectors, called embeddings, that represent the meaning of a text rather than its literal words. This means that a query such as "machine stops unexpectedly" can match a stored record that says "unexpected production halt" because the two phrases are close in meaning, even if they share no words in common.

Retrieval-augmented generation (RAG) was systematically formalized by (Lewis et al., 2020) as an architecture that combines this semantic retrieval capability with a generative language model. A RAG system consists of three core components working in sequence. First, an embedding model converts both the stored knowledge entries and the incoming user query into vector representations. Second, a vector database stores those representations and performs fast similarity search to identify the most relevant entries for a given query. Third, a generative language model receives those retrieved entries as context and produces a natural language response directly answering the user's question. The critical property of this architecture is that the model's answer is grounded in the retrieved content rather than in its general training data. This allows the system to respond accurately about domain-specific knowledge it was never trained on while minimizing hallucinations[4], which is not achievable with a generative model used in isolation.

2.5 Key Concepts and Variables

The key constructs used in this research are derived from the theoretical framework above and operationalized as follows:

- **Tacit knowledge** is defined as knowledge that individuals possess through experience and practice but cannot fully articulate or transfer through written instructions (Polanyi, 1966). It is the central construct around which the problem is structured. In this research, tacit knowledge

[4]A hallucination refers to the generation of incorrect, inconsistent or unverifiable information by a large language model (LLM), often presented in a plausible manner (Zhang et al., 2025).

is treated as a qualitative construct and is surfaced through observation, situated demonstrations and case-based elicitation rather than measured directly.

- **Expert dependency** is defined as the degree to which organizational operations rely on the knowledge of one or a few individuals, creating structural vulnerability when those individuals are absent or leave (Joe et al., 2013). It is operationalized through two variables: the frequency with which operators consult the expert and the proportion of troubleshooting situations that cannot be resolved without expert involvement. Both are measured through questionnaires, interviews and observation.
- **Knowledge retrieval efficiency** is defined as the speed and success with which a user can locate and apply relevant knowledge in response to an operational need (Davenport & Prusak, 1998). It is operationalized through two variables: retrieval time and retrieval success rate. Both are measured during prototype validation and compared against the search times reported in the current-state analysis.
- **Knowledge accessibility** is defined as the extent to which documented knowledge is findable, understandable and applicable by users within their operational context (Dalkir, 2025). It distinguishes between knowledge that is technically stored and knowledge that is actually usable in practice. It is operationalized through operator assessments of existing documentation in phase one and through informal operator feedback on the prototype in phase four.
- **Semantic retrieval** is defined as the retrieval of information based on learned vector representations of meaning, rather than exact keyword overlap. Queries and stored records are both converted into numerical representations and compared in a shared space, allowing texts that are similar in meaning to be matched even when they use different words (Karpukhin et al., 2020;Reimers & Gurevych, 2019). In this research, semantic retrieval is the mechanism through which documented troubleshooting and set-up knowledge can be found using operators' natural language descriptions of a problem. It is operationalized through retrieval relevance: the extent to which the prototype returns knowledge entries that users judge to be relevant to their query.
- **Retrieval-augmented generation** (RAG) is defined as an architecture that combines generative language model with access to an external knowledge source – meaning knowledge stored outside the model itself, such as a database of documents – from which relevant passages are retrieved and used to inform the generated answer (Lewis et al., 2020). Rather than relying solely on what the model learned during training, RAG first retrieves the most relevant stored entries and then produces an answer based on those entries. In this research, RAG is the technical solution concept that links knowledge storage, retrieval and user-facing explanation. It is operationalized through the prototype's ability to retrieve relevant stored knowledge and generate an understandable answer to troubleshooting or set-up questions.

2.6 Synthesis

The three frameworks (Polanyi's theory of tacit knowledge, Nonaka's SECI model and the MOA framework) are complementary. Polanyi (1966) establishes that expert knowledge is inarticulate by nature, which sets a realistic ceiling for what any documentation effort can achieve. Nonaka et al. (2000) identify externalization as the organizational stage most commonly underdeveloped in manufacturing contexts, which defines where a knowledge management intervention should focus. Gavrilova & Andreeva (2012) specify the conditions – motivation, opportunity and ability – that determine whether externalization can succeed in practice, which informs how elicitation should be designed and how a knowledge system should support contribution. RAG (Lewis et al., 2020) provides the technical architecture that closes the retrieval gap. Together, these perspectives suggest that an effective solution must go beyond technical functionality: it must account for the partly inarticulate nature of expert

knowledge, target the externalization bottleneck and retrieve knowledge in a way that is accessible to non-technical users on the production floor.

3. Current-State Analysis

This chapter presents the empirical findings of Phase 1 and answers sub-question 1:

I. What critical production and troubleshooting knowledge currently exists at Huima, and how is it currently created, stored and retrieved?

Data collection combined a structured operator questionnaire (see Appendix A) distributed among eight production floor operators, semi-structured interviews with Willy and management[5], workplace observations during site visits and analysis of existing documentation available on the work floor. The questionnaire was distributed to all operators working on the relevant machines, meaning that it covered the full relevant production-floor group at the time of data collection rather than a selected sample. Respondents were therefore not chosen by the researcher or by the company, participation followed directly from which machines are in scope, and their experience levels are reported in Section 3.3. However, because the group was small (n=8), the results cannot be used for statistical generalization. The findings are therefore interpreted as indicative of Huima's specific situation and are used to support the current-state analysis rather than to make broader statistical claims.

Each method serves a distinct purpose. The questionnaire identifies patterns across the production-floor team that would be difficult to establish from individual conversations alone, while the semi-structured interviews with Willy and management provide the reasoning and context behind those patterns. Workplace observation was added to capture actual retrieval behavior rather than relying only on self-reported accounts, since operators may describe troubleshooting differently from how they carry it out in practice. Document analysis establishes what knowledge is already recorded and how usable that material is. Together, these methods triangulate the same underlying phenomenon – expert dependency – from multiple perspectives, strengthening the credibility of the findings despite the small workforce.

3.1 How Knowledge Is Created, Stored and Retrieved

When a problem occurs during production, operators at Huima do not follow a formally documented troubleshooting procedure. Instead, they rely on a sequence of steps that has developed informally over years of practice within the company. For this thesis, understanding this sequence matters because it reveals both where the process breaks down and why dependency on Willy has become structurally embedded.

The process begins when an operator notices a deviation: a machine stops, displays an error code or behaves in an unexpected way. From there, operators attempt to resolve the problem through a combination of independent reasoning, consulting available notes or logbooks and asking a colleague. When these avenues do not produce a solution, the operator escalates to Willy. The detailed questionnaire findings on how often each step is taken, how long it lasts, and how reliably it succeeds are presented in Section 3.3; this section describes the shape of the process and where knowledge sits within it. Figure 2 models this process in BPMN notation.

[5] The full questionnaire and interviews are provided in Appendix A

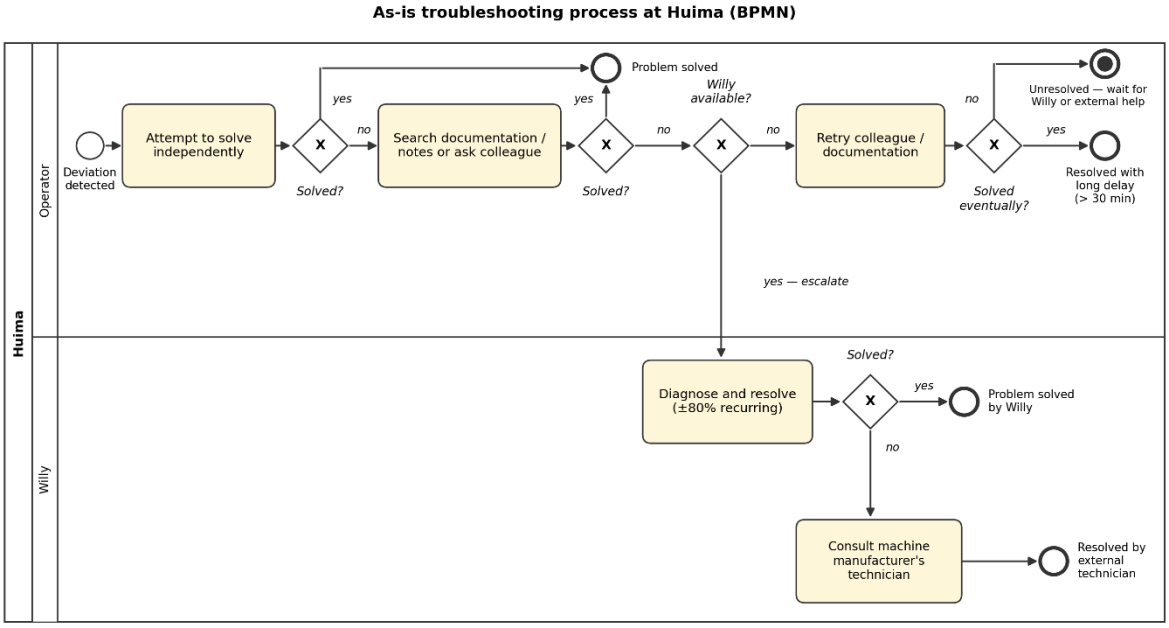


Figure 2: As-is process model

This process is also where new knowledge is created. Each genuinely new problem that Willy or an operator works through produces a new piece of troubleshooting knowledge – a recognized symptom, a tried solution, an outcome. At Huima, however, this moment of creation is also the moment the knowledge is lost: the reasoning happens in someone's head and, once the machine is running again, is rarely written down anywhere. The only formal mechanism intended to capture it – the machine logbooks – depends on operators recording disturbances after the fact, which, as shown below, happens inconsistently. The result is that knowledge creation and knowledge capture are not connected: the natural moment to capture a solution is precisely the moment it is least likely to be documented, whereas the literature recommends capturing knowledge during or directly after task execution (Kernan Freire et al., 2023).

The operational knowledge at Huima can be divided into three broad categories based on its location and accessibility (see Figure 3).

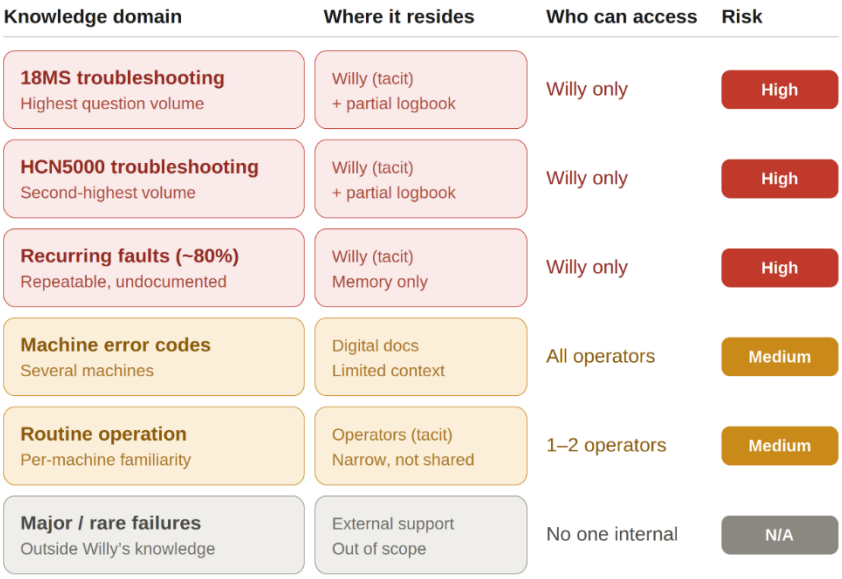


Figure 3: Knowledge Map

Willy's tacit knowledge is the most critical and least accessible category. Willy has worked with every machine in the production environment and has accumulated detailed troubleshooting and configuration knowledge across all of them over decades of hands-on practice. According to his own estimate, approximately 80% of the questions he receives from operators concern problems that he has answered before, in the same or nearly identical form. This is a practically important observation, because it suggests that a large share of the most frequently needed knowledge is in principle repeatable and therefore capturable. It is not that every question is unique or requires fresh reasoning: the same problems recur, Willy recognizes them, and he gives the same answer he has given before – but that answer exists nowhere except in his memory.

Willy identifies the 18MS machine as the one generating the highest volume of troubleshooting questions, making it the most critical single machine from a knowledge-dependency perspective, with the HCN5000 second. He estimates that across both machines roughly 99% of problems are caused by human error – specifically, insufficient understanding of machine behavior and settings rather than mechanical defects or hardware faults. Both figures are Willy's own expert estimates rather than measured values, and are treated as such: nonetheless they are directly relevant to the design of a retrieval system, because they imply that the knowledge to be captured is mainly about recurring operator mistakes and their solutions rather than rare or complex failures.

Handwritten and digital documentation is the second category (the first yellow category in Figure 3). This includes logbooks kept near machines that record error codes and previous troubleshooting cases, as well as personal notes and binders developed over years of use. In addition to these physical records, error-code documentation for several machines is maintained digitally, although it is limited in scope. For this research, the 18MS logbook is prioritized for analysis because the 18MS generates the highest volume of troubleshooting questions and therefore represents the most critical case for knowledge capture.

Operational knowledge is the third and most distributed category (the second yellow category in Figure 3). Through daily practice, operators develop working familiarity with the one or two machines they operate most frequently. This gives them a degree of practical self-sufficiency for routine situations on

their own machine, but the knowledge is narrow, rarely articulated and not shared systematically across the team. A formal process for recording troubleshooting cases does exist – operators are expected to write resolved disturbances and relevant observations into the machine logbooks – but it is not consistently followed and is neither actively motivated nor enforced. As a result, the logbooks contain an incomplete, outdated and uneven record of operational knowledge: useful when a relevant entry happens to exist, but unreliable as a comprehensive reference.

3.2 Scope of the Dependency

In this chapter, expert dependency refers to the degree to which troubleshooting at Huima relies on Willy's knowledge, as operationalized in Section 2.5. Before quantifying the dependency, it is important to define the type of problem this dependency concerns. The majority of troubleshooting situations at Huima are relatively small in scope: recurring configuration errors, minor machine faults, or operator mistakes that Willy has seen many times before. These are the situations that consume operator search time and are then resolved more efficiently once Willy is involved.

A smaller number of problems are considerably more serious – major technical failures that can halt production for a full day or longer, and in some cases fall outside Willy's knowledge entirely and require external technical support. This distinction between small recurring problems and major incidents is based on the interviews with Willy and management. The dependency analyzed in this chapter therefore applies primarily to the high-frequency, small-to-medium category of problems. Major incidents represent a different kind of challenge that lies largely outside the scope of what any internal knowledge system could be expected to resolve independently, and are not the target of the proposed solution.

3.3 Expert Dependency: Questionnaire Findings

The questionnaire was distributed among eight production floor workers operating the relevant machines. Participation was voluntarily and was pushed once initially from management and once more as reminder. Eight operators were willing to respond. As noted, the sample is small and findings are treated as indicative rather than statistically representative; nonetheless, the results provide a consistent picture that aligns with the observations made during site visits.

The operators are an experienced group: five out of eight have worked at Huima for more than five years and the other three between one and five. The questionnaire therefore captures the perspective of workers who are familiar with the existing documentation and with Willy's role, rather than newcomers still learning the machines.

First response to a problem. When asked what the operators do first on encountering a problem they cannot immediately resolve, responses were split across three strategies: 50% of operators attempt to solve it independently without documentation, 25% first consults documentation or notes, and 25% asks a colleague. Notably, no respondent selected "asking Willy" as a first step. This is an important nuance, because it complicates the assumption that operators turn to Willy the moment something goes wrong. In practice they try other avenues first and escalate to Willy only when those avenues fail.

How quickly avenues fail. The questionnaire shows how fast that escalation happens: five out of eight operators reported spending between 15 and 30 minutes searching for a solution before consulting Willy. One reported searching for less than five minutes and two reported usually not consulting Willy. Because operators do not consult Willy immediately, the dependency is partly hidden: from a management perspective the floor may appear to handle problems autonomously for the first 15 to 30 minutes, even though reliable resolution still depends on Willy being reachable at the end of that period – and that delay is accumulating as lost production time before the dependency even becomes visible.

Why consulting Willy is faster. Once Willy is consulted, resolution tends to be more efficient than independent searching. This follows directly from his estimate that around 80% of incoming questions concern previously resolved problems: for these recurring situations he draws on established knowledge rather than reasoning from scratch, which gives him a clear advantage over an operator searching documentation that was never written with retrieval in mind. For genuinely new problems, the remaining 20%, resolution takes longer and the outcome is less predictable. The contrast between 15 to 30 minutes of operator searching and Willy's near-immediate handling of familiar problems is the most concrete expression of the gap between the current and normative situations.

Frequency of consultation. Three operators reported consulting Willy approximately once per week, three once per month, 1 more than once per week and one undefined. These figures should be read with some caution. Consultation frequency is self-reported and likely underestimates the true frequency of knowledge-seeking interactions, since self-reported behavioral data is known to be subject to recall bias (Saunders et al., 2023). Willy is frequently present on the production floor, so brief informal exchanges – a quick question while passing a workstation – may not be registered by operators as deliberate consultation. The reported frequencies therefore probably reflect memorable, intentional moments of asking for help rather than the full range of interactions in which his knowledge is drawn upon. Even with this caveat, they represent a consistent and recurring pattern of expert consultation across the team.

Self-sufficiency. When asked how much of their technical problems they could resolve independently, responses were spread widely. One operator mentions 0-25%, one 25-50%, two 50-75%, two 75-100% and 2 don't know the answer. The key point the data supports is that self-sufficiency is not a team-level characteristic but varies considerably between individuals: at the lower end, at least one operator can independently handle fewer than a quarter of the troubleshooting cases they encounter, which represents a real organizational vulnerability. This variation may partly reflect the machine each operator primarily works on. An operator assigned to a machine with a higher fault frequency or greater configuration complexity will encounter more situations requiring outside knowledge than one working on a more stable, easier machine. The spread therefore captures both individual competence and machine-related variability, and should not be read as reflecting differences in skill or experience alone.

When Willy is unavailable. Five respondents reported turning to a colleague, two reported returning to documentation and one reported a different approach, no respondent reported waiting for Willy to return or being able to solve the problem independently in this scenario. The time cost of his absence is substantial: three operators reported that resolving a problem takes more than 30 minutes longer when Willy is not present, 2 reported between 15-30 minutes, 1 reported no longer and 2 reported they don't know. In the majority of cases, then, Willy's absence adds at least half an hour to a problem he would otherwise have solved in minutes. Combining the reported consultation frequency with the reported delay per incident gives a conservative estimate of the production time lost to expert dependency. Taking the lower bound of each reported band and excluding the two operators whose answers fell outside the quantifiable ranges, the minimum aggregate delay amounts to approximately 460[6] minutes – almost eight hours – per month. Because this figure uses the lowest value of each band and omits two respondents, the true cost is necessarily higher; 460 minutes should therefore be read as a floor rather than an estimate.

[6] Six out of eight respondents reported quantifiable frequency and delay values. For each respondent, the lower bound of the reported band was used: "more than 1x/week"→ 2x/week. Summing frequency x delay per respondent yields approximately 460 minutes of minimum delay per month.

Documentation use. Most operators report using documentation frequently – four said “often”, two said “almost never” and two said “always”. When it is used, three indicated that they “sometimes” find the solution and four “often”, one said they don’t use the documentation to find a solution. Documentation is therefore not useless, but it is inconsistent. Asked to identify its main problem, four mentioned it is hard to find the right solution, four said there is lack of clear explanation, two said it is not up-to-date, one said it is not located at a logical location, five said it is unclear where to look and one said he does not experience any problems with current documentation. Operators know documentation exists, attempt to use it, but still regularly fail to find what they need – at which point Willy becomes the necessary fallback.

What would reduce dependency. An open question asked what would help operators become less dependent on Willy. Six out of eight operators responded, with varying solutions including step-by-step procedures, photo’s with instructions, better search functionality, logbooks up-to-date and a way to capture information by the operator himself in a digital environment. The responses pointed clearly in the same direction: a structured digital system with better search functionality, step-by-step troubleshooting guides, photographs and visual instructions. These answers directly reflect the findability and accessibility gaps identified in the closed questions and provide operator-grounded support for the solution direction developed in later phases.

3.4 Summary: Current-State Assessment

The findings of Phase 1 can be summarized across the three dimensions defined in Section 1.1.3: knowledge coverage, standardization and expert dependency.

In terms of coverage, the majority of critical operational knowledge is undocumented. Willy’s pattern-based troubleshooting expertise exists entirely as tacit knowledge, even though the fact that around 80% of questions are recurring suggests much of it is capturable. In terms of standardization, the documentation that does exist is inconsistently structured, difficult to navigate and maintained without indexing or version control. In terms of expert dependency, the typical pattern is 15 to 30 minutes of unsuccessful searching followed by escalation to Willy, who resolves the issue quickly. When he is unavailable, most operators experience a resolution delay of more than 30 minutes. The dependency is not the result of operators being passive: it is the result of every available alternative being insufficiently reliable to replace Willy.

Together, these findings establish the empirical basis for the requirements specification in Phase 2. The core problem is not that operators fail to use documentation or that the knowledge does not exist – it is that the knowledge exists in a form that cannot be found, navigated or applied independently (see Figure 4). A solution must therefore address findability and contextual accessibility before it can meaningfully reduce dependency on Willy.

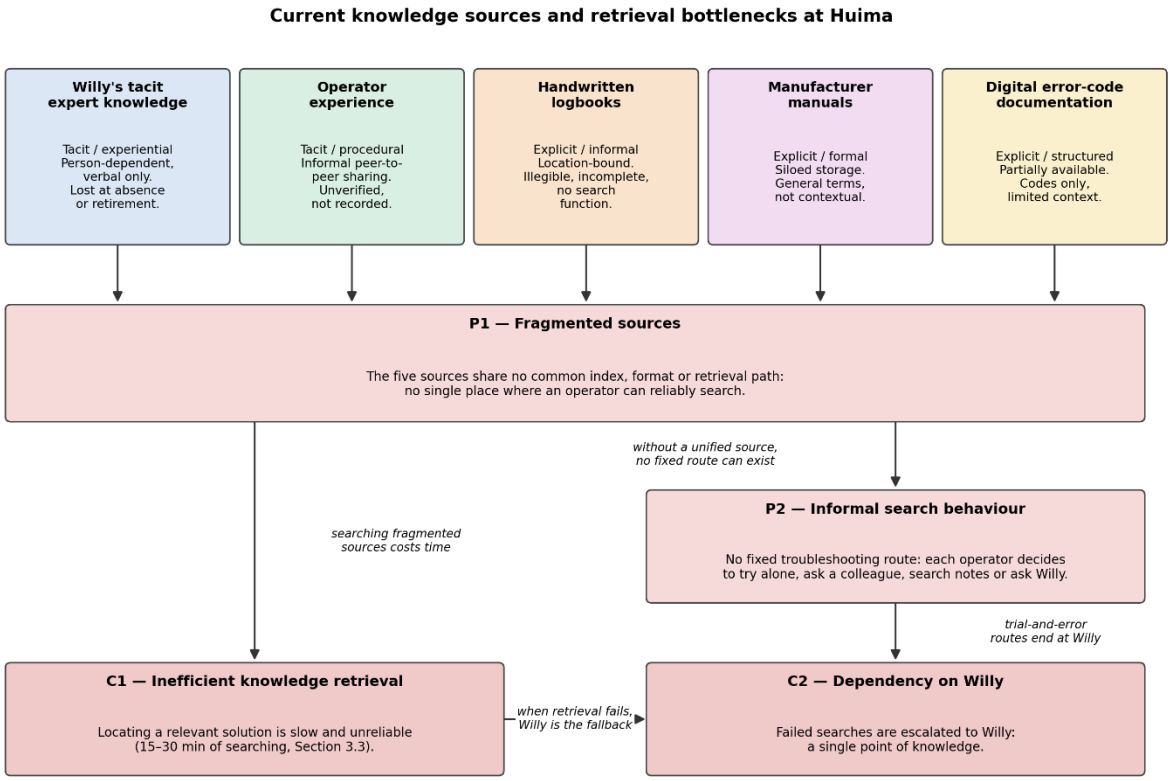


Figure 4: Current knowledge sources and retrieval bottlenecks

4. Literature Review

This chapter surveys the existing approaches and technologies for capturing, structuring and retrieving tacit knowledge in manufacturing environments, in order to answer sub-question 3: *What approaches and technologies exist in literature for capturing, structuring and retrieving tacit knowledge in manufacturing environments?* Where chapter 2 established why the problem exists and what conditions a solution must satisfy, this chapter is concerned with what has actually been built and tested. The theoretical concepts introduced earlier – Polanyi's tacit dimension, Nonaka's externalization bottleneck and the MOA conditions for elicitation – are used here only as evaluative lenses.

The chapter is organized around the three stages of the knowledge process. Section 4.1 reviews capture approaches, ranging from traditional analyst-driven methods to recent AI-assisted techniques. Section 4.2 reviews structuring approaches. Section 4.3 reviews retrieval approaches, including the limits of traditional knowledge management systems and the emergence of semantic and retrieval-augmented methods. Finally Section 4.4 synthesizes the findings.

4.1 Capturing Tacit Knowledge

The reviewed literature shows a clear spectrum of capture approaches that differ mainly in how much analyst effort and specialized equipment they require. They can be grouped into three categories: analyst-driven interview and observation methods, sensor- and recording-based methods, and AI-assisted elicitation.

Analyst-driven methods remain the most established and best-validated. Hoerner et al. (2023) applied a four-step externalization approach – unstructured interviews, observation combined with semi-structured interviews, structured textual analysis and protocol-based validation – across 36 operators over six weeks, producing a knowledge base of 261 structured if-then troubleshooting rules. (Johnson et al., 2019) applied hierarchical task analysis combined with Rasmussen's skill-rule-knowledge framework[7] to visual inspection in two aerospace facilities, classifying each task step by the type of cognition it required. They found that up to 50% of performance in the search phase of inspection was skill-based, meaning it relied on automatic, internalized routines rather than documented procedure, and that this knowledge was not present in any standard operating instruction. Madsen et al. (2008) studied three manufacturing transfer cases using 59 interviews and 1.5 years of observation, and proposed a method for identifying hidden knowledge by defining triggers, enacting the operation and designing a training program around the results.

Sensor- and recording-based methods capture expert behavior through recording equipment. Yoshida et al. (2011) combined field-oriented interviews with human motion capture and video analysis in a whetstone manufacturing factory, comparing expert and beginner performance to identify physical techniques the expert was not consciously aware of, such as using different regions of a polishing disk for different operations. The expert's productivity was nearly three times that of the beginner. To confirm which behaviors were tacit, the findings were shown back to the expert. Sarhan et al. (2022) proposed using augmented-reality head-mounted devices with 3D cameras, gesture tracking, spatial mapping and voice commands to capture knowledge during task execution for later delivery to less experienced workers: the authors present this as a conceptual proposal without empirical evaluation. Burggraef et al. (2024) presented a concept for automatically generating assembly instructions by analyzing process recordings through object detection, activity detection and post-processing, merged into a structured

[7] Rasmussen's skill-rule-based framework classifies three levels of human performance: skill-based behavior, rule-based behavior and knowledge-based behavior (Rasmussen, 1983)

process model by a semantic reasoning engine; the authors describe this as a concept overview focused on procedural rather than tacit knowledge.

AI-assisted elicitation uses conversational and machine-learning systems to elicit knowledge with less analyst involvement. Kernan Freire et al. (2023) developed a cognitive assistant for manufacturing operators using voice and text interaction. The assistant prompts workers after task completion or during downtime, summarizes their performance relative to historical averages and asks reflective questions, then uses natural language processing to extract entities such as machine names, solutions and strategies, populating the knowledge graph[8]. In a comparison of voice-based and paper-based reporting, the authors found that report quality was significantly higher for voice reporting and that voice reporting during tasks affected task completion time less than writing reports afterwards. Wang et al. (2026) proposed a method for analyzing visual tacit knowledge in welding inspection, using eye-tracking to capture expert behavior and a "visual chain-of-thought" approach with large language models. The result is a structured map of the expert's inspection strategy – which cues are checked, in what order and what each one signals – validated by experts as accurate.

4.2 Structuring Captured Knowledge

The literature shows three main approaches to organizing captured knowledge so it can be searched and reused:

- Hierarchical and rule-based structures organize knowledge into navigable categories. (Hoerner et al., 2023) arranged their 261 rules into a tree structure: operators first select a broad error category, then narrow it by selecting the error's appearance, and the system returns a list of possible solutions ranked by how often they previously worked. (Madsen et al., 2008) classified knowledge by situation type – normal operation, disturbance, breakdown and improvisation – and argues that each type requires a different structuring approach, with normal-operation knowledge held in standard documentation and other types requiring situation-specific guides or embedded training.
- Ontologies and knowledge graphs represent knowledge as connected entities and relationships. (Wan et al., 2019) built a case-based reasoning[9] system for CNC machine-tool maintenance in which each stored case consists of a problem description, the solution previously applied and a step-by-step procedure: the system identifies relationships between concepts rather than relying on keywords and adapts a similar past case when no exact match exists. (Kernan Freire et al., 2023) store extracted knowledge in a knowledge graph in which machines, actions and outcomes are represented as connected nodes, with context added automatically so that relevant past experience can be returned when a similar problem occurs.
- AI-driven structuring organizes unstructured text automatically. (Cho, 2026) used generative AI models (BART, Qwen, T5) to extract structured information from unstructured equipment maintenance texts and sort it into predefined categories without manual labelling, with the Qwen model performing most consistently. (Naqvi et al., 2024) fine-tuned a BERT model on industrial maintenance records so that it could understand domain-specific language and organize inconsistently written records for retrieval. Across the structuring literature, several authors report that human validation of structured knowledge remains necessary before use: (Hoerner et al., 2023) and (Kernan Freire et al., 2023) both state that unverified knowledge in a manufacturing context can lead to errors or safety incidents.

[8] A knowledge graph stores information as networks of entities, connected by relationships (Hogan et al., 2021)
[9] Case-based reasoning – a system solves a new problem by retrieving a similar past case and adapting its solution, instead of applying general rules. (Aamodt & Plaza, 1994)

4.3 Retrieving Tacit Knowledge

Retrieval receives the least attention in the reviewed literature, although retrieval in general is a well-studied field. The reviewed approaches range from traditional knowledge management systems to semantic, case-based, generative and conversational retrieval.

Traditional knowledge management systems are the conventional response to knowledge loss. (Durst & Wilhelm, 2012) studied how medium-sized manufacturers respond to the risk of losing knowledge when key individuals depart, finding that operational knowledge is typically concentrated in a few individuals whose departure causes immediate disruption, and that most such firms lack a systematic approach to capturing that knowledge beforehand. (Suhonen & Kankaanpää, 2025) found across 163 Finnish manufacturing companies that the dominant form of knowledge transfer remains informal socialization through master-apprentice arrangements, with the externalization stage of the SECI model consistently underdeveloped. Document-based systems such as shared drives, wikis and manuals rely on keyword search, which returns results only when the user's query contains the same terms used in the stored record.

Semantic and case-based retrieval match meaning rather than exact wording. Semantic retrieval converts both query and stored records into vector representations (embeddings) so that texts with similar meaning are matched even when they share no vocabulary (Karpukhin et al., 2020). (Naqvi et al., 2024) demonstrated this with a BERT-based semantic search fine-tuned on maintenance records, reaching precision scores of 0.94 and 0.97 on mining and aviation datasets and returning relevant records even when the searcher and original author used different terminology. (Wan et al., 2019) showed that a case-based system can retrieve the most similar past case using semantic similarity and adapt its stored solution through a causal adaptation algorithm, providing retrieval beyond keyword matching.

Retrieval-augmented generation (RAG), explained by (Lewis et al., 2020) and introduced in chapter 2, combines semantic retrieval with a generative language model: relevant entries are retrieved for a query and a language model generates a natural-language answer grounded in those entries rather than in its training data. (Lee et al., 2025) applied a multi-agent RAG framework in injection molding, retrieving relevant knowledge from documentation and using it as context to answer operator queries without the model having been trained on that content; they evaluated it on 160 tasks and reported that more capable models achieved higher accuracy and that a multi-agent architecture outperformed a fine-tuned single agent.

Conversational retrieval delivers knowledge through dialogue. (Kernan Freire et al., 2023) allowed operators to query knowledge by voice or text, using awareness of which machine and product were involved to return shorter, more relevant responses; a study with 83 participants found that prior training and interaction affected user experience. (Sarhan et al., 2022) conceptualized retrieval through a personal digital assistant that overlays pre-captured knowledge onto the worker's physical environment.

4.4 Synthesis and Gaps

Table 1 compares the reviewed system across the three knowledge stages, their SME fit and their main limitations. The table shows a central gap: each system covers one or two stages well, but none combines low-effort capture, automatic structuring and retrieval in a setting comparable to Huima.

Several patterns emerge across the reviewed literature. First, the reviewed research is concentrated on the capture stage (Hoerner et al., 2023;Johnson et al., 2019; Yoshida et al., 2011), while structuring and especially retrieval receive substantially less attention within this body of literature (Naqvi et al., 2024; Lee et al., 2025)

Second, the literature shows a shift from analyst-driven methods toward technology-assisted ones. Established capture methods (Hoerner et al., 2023; Johnson et al., 2019; Yoshida et al., 2011) produce validated results but report substantial analyst time and observation periods, while more recent AI-assisted methods (Kernan Freire et al., 2023;Cho, 2026;Naqvi et al., 2024) reduce this effort, though several are reported by their authors as prototypes or concepts (Sarhan et al., 2022;Burggraef et al., 2024).

Third, to the best of the authors knowledge, no reviewed system covers all three stages – capture, structure and retrieval – in a single integrated, low-effort form. Hoerner et al. (2023) combine externalization and an assistance system but keep capture manual. (Lee et al., 2025) demonstrate strong retrieval but rely on pre-existing documentation rather than newly captured tacit knowledge. Kernan Freire et al. (2023) address capture and retrieval but remains at prototype stage.

Fourth, human validation recurs as a stated requirement across both structuring and retrieval: several authors report that automatically structured or generatively produced knowledge must be checked by an expert before use (Hoerner et al., 2023;Kernan Freire et al., 2023).

Fifth, organizational barriers are reported independently of technology. (Suhonen & Kankaanpää, 2025) found that some workers withhold knowledge as a form of job security, and (Hoerner et al., 2023) report that not all workers are motivated to participate in externalization. These findings describe the current state of the field. The literature offers analyst-driven capture, sensor-based capture and AI-assisted capture; rule-based, graph-based and AI-driven structuring; and keyword-based, semantic, case-based, generative and conversational retrieval. Each is documented at a different level of maturity and with different reported trade-offs. Which of these approaches is appropriate for a given organizational context, and the criteria against which that should be judged, depends on context-specific requirements, which are established in the following chapter.

Table 1: Comparison of reviewed systems and this thesis across the three knowledge stages

Study	Capture	Structure	Retrieval	SME context	Key limitation
(Hoerner et al., 2023)	Yes – analyst-driven: interviews, observation, protocol analysis	Yes – hierarchical rule tree with ranked solutions	Partial – manual category navigation, no semantic search	No – large continuous production line	Capture is highly manual; requires skilled analyst and six weeks of observation
(Kernan Freire et al., 2023)	Yes – AI-assisted: conversational prompts, voice/text reflection	Yes – automatic entity extraction into knowledge graph	Yes – conversational, context-aware	Partial – agile manufacturing pilot	Remains at prototype stage; not validated in long-term production use
(Wan et al., 2019)	No – relies on existing maintenance cases	Yes – case structure (problem, solution, procedure)	Yes – case-based reasoning with semantic similarity	No – CNC maintenance planning context	Depends on pre-existing, well-formed case base
(Naqvi et al., 2024)	No – uses existing maintenance records	Partial – fine-tuned BERT organizes inconsistent records.	Yes – semantic search (precision 0.94-0.97)	No – mining and aviation datasets	Requires large historical text corpus and model fine-tuning
(Cho, 2026)	No – processes existing unstructured texts	Yes – generative AI sorts texts into predefined categories	No – structuring only	No – manufacturing execution system data	Covers structuring only; no capture or retrieval component
(Lee et al., 2025)	No – relies on pre-existing documentation	Partial – documents indexed for retrieval	Yes – multi-agent RAG answering operator queries	No – injection molding industry setting	Does not capture new tacit knowledge; assumes documented input
(Suhonen & Kankaanpää, 2025)	Partial – describes informal socialization practices	No	No	Yes – 163 Finnish manufacturing companies including SMEs	Descriptive study; identifies the externalization gap but offers no system
This thesis	Yes – low effort: voice, text and image input	Yes – automatic structuring into validated knowledge entries	Yes – semantic search and RAG-based natural language answering	Yes – SME with low data maturity; local data boundary	Prototype scope: validated on representative cases, not full knowledge coverage

5. Solution Design

The preceding chapters established what the solution must achieve: Chapter 3 showed where the current troubleshooting process fails and why the dependency on one expert has become structural. Chapter 4 showed which approaches exist in the literature for capturing, structuring and retrieving tacit knowledge. This chapter turns those findings into a concrete solution. It does this in four steps, each answering part of sub-question 2 and sub-question 4. Section 5.1 specifies the requirements the solution must satisfy, gathered from stakeholders and the current-state analysis and prioritized using the MoSCoW method. Section 5.2 explores four solution directions identified in the literature. Section 5.3 selects the solution through a multi-criteria analysis. Finally, Section 5.4 then describes the prototype that was built on that basis, and why each design choice was made.

5.1 Requirement Specification

This section answers sub-question 2: what functional, technical and organizational requirements a knowledge capture and retrieval solution must satisfy. The requirements are derived from two sources. The current-state analysis in Chapter 3, which showed the real problems the solution has to address. The second is a series of stakeholder interactions in which those problems were turned into concrete criteria. Once identified, the requirements are prioritized using the MoSCoW method so that the prototype's scope is defined deliberately rather than by default.

MoSCoW is a prioritization technique, not a method for generating requirements. The requirements themselves follow from the analysis and from stakeholder input; MoSCoW only sorts them into priority bands (Must have, Should have, Could have, Won't have). This matters because the prototype is limited in scope on purpose, and a clear prioritization makes it visible which criteria are essential for a working system and which are valuable but can be left out for now.

5.1.1 How the Requirements Were Gathered

Requirements were not established from a single interview but built up through repeated stakeholder contact. One-to-one semi-structured interviews were held with management and with Willy, supplemented by observations during site visits and by the operator questionnaire reported in Chapter 3. This was done on purpose. Stakeholders rarely state all their requirements at once, and some preferences only come up once a concrete idea is on the table. By returning to stakeholders at several moments rather than relying on one session, the implicit requirements could be found and refined. This is the same elicitation logic applied in the rest of the project, where it is assumed that tacit preferences cannot be fully said in a single meeting (Polanyi, 1966; Hoffman et al., 1995).

The requirements are grouped into three types: functional (what a system must do), technical (constraints the implementation must respect) and organizational (conditions for adoption and maintenance in Huima's context).

5.1.2 Must have

Must-have requirements are non-negotiable: if any is not met, the solution fails to address the core problem regardless of how well it performs otherwise.

A controlled local data boundary. Management permits cloud processing of voice input, notes and manuals, but the contents of the knowledge database may not be exposed to external cloud services. So the solution must enforce a one-way boundary: captured input may be processed externally, but stored knowledge remains local. This is the most restrictive technical requirement and shapes the whole architecture.

Dutch-language capture and retrieval. Operators and Willy work in Dutch, and all source material – logbooks, notes, spoken explanations – is in Dutch, including a regional Twents accent in speech. A system that cannot operate in Dutch is unusable on the floor. This traces directly to the Chapter 3 finding that all existing knowledge is recorded and communicated in Dutch.

Multi-modal input: text, voice and image. Operator input gathered in Chapter 3 clearly pointed to more than one input mode: photographs or visual instructions, written entries and spoken interaction. Voice input is treated as a must-have on both operator and literature grounds. Writing reports is effortful, and operators tend to type only the minimum. Voice lowers this barrier and has been shown to produce higher-quality reports than written reporting, without taking more time (Kernan Freire et al., 2023). It also helps with a known problem: experts under-report the cues they actually use (Polanyi, 1966), and a lower-effort capture mode brings out more of that reasoning. Image input lets operators attach error screens or machines states that are hard to describe in words.

Usability for non-technical operators. Chapter 3 established that operators vary widely in self-sufficiency and that the main complaint about existing documentation is that it is unclear where to look. A solution requiring technical skill would reproduce the existing accessibility gap. So the interface must be usable by an operator with no technical background and only minimal training.

Add/update entries through a validation gate. The system's value depends on knowledge being captured correctly. Captured entries must pass a human validation step before becoming retrievable. An unvalidated entry in a troubleshooting context could spread incorrect or unsafe action, which is a risk the literature review explicitly identifies (Hoerner et al., 2023;Kernan Freire et al., 2023). The validation gate also guards against a known limitation of expert self-report: a single expert reconstructing a past case from memory can misremember or rationalize, so captured knowledge is verified rather than trusted on first entry (Hoffman et al., 1995).

Low burden on Willy. Because Willy is the primary knowledge source and is already a bottleneck, the capture process must not consume significant amounts of his time. He said this several times: he wants to spend his last two years working with machines like he always has. Knowledge capture has to fit into that, not becoming a second job. This requirement follows from the MOA framework condition: externalization only happens if the organizational conditions make it easy enough to sustain (Gavrilova & Andreeva, 2012). A capture method that demands Willy's full attention would not survive contact with daily production.

Retrieval faster and more reliable than the current process, for knowledge in the system. The core problem is that operators spend too much time before falling back to Willy. The minimum bar is that, for knowledge already captured, the system returns a relevant answer substantially faster than current documentation search. Retrieval accuracy is important, but it is tested in a limited scope. The prototype is validated on a selection of representative cases rather than across the full breadth of Huima's problems, so accuracy is demonstrated for captured knowledge and not claimed for everything.

Cost proportionate to value. Huima is a small manufacturer with limited budget and low technical maturity. Management indicated a willingness to invest in a system provided it works, rather than a fixed budget. As an upper reference, the commercial alternative Artific[10] starts at about 600 euro per month. So the requirement is therefore that the total cost of ownership remains below that commercial benchmark, favoring low-cost, maintainable components over recurring license fees.

[10] Artific is an Ai-agent company based in Enschede. Huima has been exploring Artific's solution offerings as a possible commercial alternative. Pricing starts at approximately €600 per month.

5.1.3 Should Have

Should-have requirements add substantial value but are not essential for a working prototype: their absence reduces quality or adoption without breaking the system.

A structured, searchable organization of entries so that knowledge entered by one person can be found by others. Operators said the system should be a shared, navigable resource, not a personal store. A basic structure is essential, but the richer organizational refinements belong here and not in the Must band.

Category-based navigation mirroring the physical logbook. Operators already understand the logbook's organization, so reflecting it reduces adoption friction. The system functions without it – search alone returns answers – but navigation lowers the barrier for operators who prefer browsing to querying.

A domain-specific glossary for abbreviations and jargon. Operators and source material use shorthand and machine-specific terms, so a glossary supports retrieval accuracy. It can be partially implemented (for example by supplying the glossary to the model that structures captured input) and expanded later, which is why it is a Should rather than a Must.

A clear visual distinction between validated and unvalidated entries. This complements the must-have validation gate: the gate controls what enters the system, while the marker communicates entry status to the user so they can judge how far to trust an answer.

Search history and recently viewed items. These help operators return quickly to recurring problems. This is relevant because, on Willy's estimate, roughly 80% of questions concern issues seen before. This is a usability enhancement rather than a core function.

A mobile-friendly interface. Operators work on the floor, not at a desk, so phone access is practically important. It is a Should because the system remains functional on a desktop display. But a bad mobile experience would lower usage, so it is treated as a high-priority Should.

5.1.4 Could Have

No could-have requirements were identified. The requirements that emerged from the analysis and stakeholder input were either essential to a working prototype (Must) or clear quality and adoption improvements (Should). Features beyond these were deliberately placed out of scope (Won't).

5.1.5 Won't Have

Won't-have requirements are deliberately excluded. Recording them defines the boundary of the prototype and prevents scope creep; it is not a judgement that they lack value.

Full population of knowledge base. The prototype demonstrates the system on representative troubleshooting cases. Capturing the entirety of Willy's knowledge is an implementation task beyond the project timeframe, as stated in the project scope.

Automated or sensor-based capture (video analysis, AR, motion capture). The literature shows these methods are either conceptual or limited to procedural knowledge and require infrastructure that Huima lacks (Sarhan et al., 2022;Burggraef et al., 2024); they are discussed as forward-looking options in the implementation advice rather than built.

Integration with existing production or ERP systems. No requirement for live integration was raised, and it would add technical complexity disproportionate to the prototype's validation goal.

5.1.6 Summary: MoSCoW Prioritization

The table below summarizes the prioritized requirements with their category and the source from which each was derived. The source column makes the requirements traceable to either the current-state analysis, a stakeholder or the literature. This prioritized set of requirements forms the evaluation framework used in the rest of the project. Candidate solutions are assessed against it.

Table 2: MoSCoW Summary

Priority	Requirement	Category	Source
Must	Local data boundary (no database content sent to cloud)	Technical	*Management interview*
Must	Dutch-language capture & retrieval	Functional	*Ch. 3 / operators*
Must	Multi-modal input (text, voice, image)	Functional	*Ch. 3 operators; Kernan Freire et al. (2023)*
Must	Usable by non-technical operators	Organizational	*Ch. 3 questionnaire*
Must	Add/update entries via human validation gate	Functional	*Willy; literature review; Hoffman et al. (1995)*
Must	Low burden on Willy	Organizational	*Willy interview; Gavrilova & Andreeva (2012)*
Must	Retrieval faster than current process (for captured knowledge)	Functional	*Ch. 3 (15–30 min gap)*
Must	Cost below commercial benchmark (~€600/mo, Artific)	Organizational	*Management interview*
Should	Structured organization, searchable by others	Functional	*Ch. 3 operators*
Should	Category navigation mirroring the physical logbook	Functional	*Operators / Ch. 3*
Should	Domain-specific glossary for jargon	Technical	*Willy interview*
Should	Validated vs unvalidated entries marked	Functional	*Willy; literature review*
Should	Search history / recently viewed items	Functional	*Ch. 3 (~80% recurring)*
Should	Mobile-friendly interface	Technical	*Site observation*
Should	Image analysis of photos (OCR / transcription)	Functional	*Operators; conditional on security decision*
Won't	Full population of the knowledge base	Scope	*Project plan*
Won't	Automated / sensor-based capture (AR, video)	Scope	*Literature review (Sarhan 2022; Burggräf 2024)*
Won't	ERP / production-system integration	Scope	*No requirement raised*

5.2 Solution Exploration

This section explores possible solution directions before the final design is chosen. It addresses sub-question 4: which solution concept fits Huima's context, requirements and constraints the best. The alternatives are compared as complete solution concepts rather than as isolated features, because each would have to perform the same full task – capturing troubleshooting knowledge from the expert, structuring it and making it retrievable on the floor. Comparing them as full approaches keeps the discussion close to the core problem instead of to single technical details.

The exploration is guided by the current-state analysis, the requirements specification and the literature review. Chapter 3 showed that Huima's problem is not only that knowledge is undocumented, but that existing and newly created knowledge is not sufficiently findable, understandable or reusable during production. The requirements in Section 5.1 add hard constraints: the solution must be usable by non-technical operators, support Dutch input and retrieval, keep the knowledge database under local control, reduce the documentation burden on Willy, and retrieve relevant knowledge faster than the current process. The goal here is to first widen the view of what is possible and then narrow it down to the direction that fits Huima best. Directions that clearly do not fit – for example approaches that need large amounts of pre-existing structured data, dedicated knowledge-management staff, or a permanent internet connection – are not discussed in detail, because the requirements already rule them out.

The first direction is a **rule-based expert system**. This is the oldest computational answer to exactly Huima's problem: capturing the knowledge of a scarce expert into a form that less experienced staff can use. Such systems store expertise as explicit if-then rules or as a decision tree that an operator walks through. This is familiar from printed repair manuals, where each answer leads to the next question until a fault is found. A close example from the reviewed literature is the system of Hoerner et al., (2023), which stores 261 troubleshooting rules in a tree that operators walk through, from a broad error category down to a specific symptom. Outside knowledge management, the same pattern is used in decades of diagnostic systems for machine fault-finding, vehicle repair and equipment maintenance, which combine a decision tree with an inference engine and sometimes fuzzy logic to handle uncertain inputs (Möller, 2015; Bassil, 2012).

The advantage of this direction is transparency. The rules are readable, each rule is a separate unit that can be added or removed on its own, and the system can explain how it reached a conclusion based on these rules. The weaknesses, however, are well documented and matter directly to Huima. The first is the knowledge acquisition bottleneck: extracting an expert's knowledge and hand-coding it into formal rules is slow, expensive and effortful. Fernandes et al. (2018) name this as the main obstacle to building such a system, and the same problem is what the MOA framework describes as the ability barrier (Gavrilova & Andreeva, 2012). For Huima this is decisive, because it would shift the entire burden onto Willy and a rule-writer, which is the opposite of the requirement to keep Willy's load low. The second weakness is brittleness: rule-based systems fail when a situation falls slightly outside their hand-coded rules, and large rule bases get hard to keep consistent and free of conflicting rules when they grow (Fernandes et al., 2018). A rigid structure also assumes that tacit expertise can be reduced to explicit rules, which Chapter 2 established – following (Polanyi, 1966) – it mostly cannot. So a rule-based system gives the outer shell of a solution but loses the reasoning that makes it useful, and it loses the easy capture which the project needs.

The second direction is a **codified knowledge management system**. (Hansen et al., 1999) describe a codification strategy as one in which knowledge is made explicit on purpose, stored in databases and reused many times. This is a people-to-documents approach, and they contrast it with a personalization strategy that is built on direct contact between people. A codified system for Huima would store each

troubleshooting entry in fixed fields such as machine, symptom, error code, cause, solution, source and validation status, and operators would retrieve entries through filters, categories and keyword search. This is less rigid than a pure rule based system, because entries are records rather than executable logic. It fits several requirements well: the entries are explicit, locally stored, structured and easy to retrieve, which supports both local data control and a shared, searchable resource.

The limitation of codification is smaller than that of rule-based direction but similar in kind. It still assumes that what matters about a solution can be written into predefined fields. In practice, a fixed template stores the outcome of Willy's reasoning, the cause and the solution, but it loses contextual cues and judgement that led to it, and that is precisely the tacit layer the project wants to preserve. A strict template also raises the effort of entering knowledge, because the person must fit what they know into the form, which works against the low-burden requirement. Codification is therefore a useful component, but on its own it is not enough as the capture mechanism.

The third direction keeps knowledge closer to its original, experiential form, instead of forcing it into rules or fixed fields. A pure personalization strategy in Hansen et al. (1999) mentions sharing knowledge only through people and direct contact, is basically Huima's current situation, where operators ask Willy, and is exactly what the project wants to reduce. A technical middle ground exists in **case-based reasoning (CBR)**: a new problem is solved by finding a similar past case and adapting its solution, instead of applying a fixed rule. Each case stores a problem, the solution that worked and the steps taken, and the system relates cases by meaning rather than by exact wording. This goes back to the foundational work of Aamodt & Plaza, (1994), who describe CBR as a cycle: find a similar past case, reuse its solution, adjust it where needed and store the new case for next time. CBR fits expert knowledge well, because an expert can usually tell you what they did in one specific situation more easily than write a general rule for every situation.

Two studies show this direction in manufacturing. Wan et al. (2019) built a CBR system for CNC machine-tool maintenance that retrieves the most similar past case and adapts its solution when no exact match exists, Kernan Freire et al. (2023) store captured knowledge in a knowledge graph, a network of connected concepts such as machine, action and result, so that a later similar problem can be returned together with its context. This direction captures relationships that a flat database misses and it fits the requirement of a structured, reusable organization of knowledge. The drawback is effort and maintainability. A knowledge graph is built on modelling decisions, which concepts exist and how they relate, and knowledge representation is itself a specialized skill that is hard to learn on the job (Noy et al., 2019). For Huima this is a problem less because of scale and more because there is nobody in-house to design and maintain such a structure, which works against the requirement that the system must stay supportable at a reasonable cost.

It is also useful to look outside the manufacturing environment literature, at how organizations in other sections solve a similar problem: helping employees find scattered internal knowledge without asking a colleague. The mainstream answer in these sectors is enterprise search, often combined with an AI assistant. In IT support, customer service and onboarding, these tools scan company's existing documents and tickets, and let users ask questions in plain language. They use semantic search, so users get relevant answers without having to ask a colleague (Gomstyn & Jonker, 2026). Two things make these tools unsuitable for Huima. First, they only work on knowledge that is already written down, but Huima's main problem is the knowledge not written down yet. So such a tool would search an almost empty database and leave the real problem, capturing knowledge, unsolved. Second, they are mostly cloud services that store company content on external servers, which breaks the must-have requirement that the knowledge stays local. So this direction cannot be used as a product, but it does confirm one

useful principle: the best general answer to a scattered knowledge answer is semantic search over a central source. Huima needs that same principle, but rebuilt for local capture and local storage

RAG is the last direction. It takes the same retrieval principle as enterprise search, but rebuild around local capture and local data control. RAG, introduced in Chapter 2, combines semantic retrieval with a generative language model. When an operator asks a question, the system finds the stored entries whose meaning is closest to the question, and a language model then formulates an answer based on those entries rather than in its own training data (Lewis et al., 2020). So the retrieval step keeps the answer tied to Huima's own knowledge, and the generation step turns a set of matches into a readable answer in the operator's own language.

RAG handles several of the weaknesses left open by the other directions. Because retrieval works on free text, knowledge does not have to be hand-coded into rules or forced into rigid fields to be findable. This avoids the knowledge acquisition bottleneck and the brittleness of the rule-based direction, and it lets an operator describe a problem in their own words. Unlike a knowledge graph or a curated case base, the knowledge lives in a local database that can be added to or corrected at any time while the language model only reformulates what is retrieved, so updating knowledge is a database task rather than a model training task. Unlike a cloud enterprise-search product, the database and the models can run locally, which is compatible with the data-control requirement. Finally, basing answers on retrieved entries lowers the risk of made-up answers (hallucinations), because the model is told to answer from the supplied entries rather than from memory. This is often named as main advantage of RAG over fine-tuning a model on company data, because fine-tuning bakes knowledge into the model, cannot show its source and must be retrained every time the knowledge changes. (Balaguer et al., 2024).

RAG also has trade-offs. It needs a generation model, which adds response latency and requires local computing capacity, and – like every direction above – it needs knowledge to be captured before it can retrieve anything. In the literature, Lee et al. (2025) showed a RAG framework for knowledge transfer in injection molding, which confirms the approach works for manufacturing knowledge, although their system uses existing documentation rather than newly captured tacit knowledge.

5.3 Solution Selection

Section 5.2 set out four candidate directions: a rule-based expert system, a codified knowledge management system, a case-based or knowledge-graph approach and a RAG. This section selects between them based on multi-criteria evaluation. The prioritized requirements from chapter 5.1 are used as criteria, and each direction is judged by how well it meets them rather than by its general merits. Enterprise search is left out, because Section 5.2 already showed it is unsuitable as a product for Huima. It is treated as a principle rather than a candidate.

The evaluation follows the logic of the MoSCoW rules directly. As said in Section 5.1.2, a must-have is non-negotiable: if a direction fails even one of them, it fails to address the core problem regardless of how well it performs elsewhere. The must-have requirements are therefore applied first as knock-out criteria. A direction that cannot meet one of them is removed, rather than being compensated for by strength on another criterion. This is a choice not to use a single weighted score: averaging the criteria would let a direction that fails a must-have outrank one that meets all of them, which goes against what a must-have means.

Table 3 applies the eight must-have requirements to the four directions. Each cell says whether the direction meets the requirement fully ("Yes"), only partially or with a lot of extra effort ("Partial") or not at all ("No"). The validation-gate requirement is read in its refined form: a human validation step is only needed for knowledge produced by automated transcription or structuring, while manually entered

knowledge is trusted on entry. Because a must-have failure is decisive, the table is read row by row: a single “No” removes that direction.

Reading Table 3 row by row removes one direction on must-have grounds: the rule-based expert system fails on two requirements. It cannot take voice or image input, only text and logic. And building it would mean hand-encoding Willy’s knowledge into formal rules, which is slow and demanding work that puts exactly the load on Willy which the requirements try to avoid (Fernandes et al., 2018). The case-based and knowledge-graph direction is different: it is a viable option that could meet the must-haves. Modern LLM-driven tools have even removed much of the manual modelling effort knowledge graphs once required (Bian, 2025; Kommineni et al., 2024). It is set aside not because it fails a requirement, but because building a knowledge graph – defining the schema, the entities and the relationships, and validating them – is a substantial task on its own, and it is hard to justify with the very limited data available during this research. It is therefore considered as a strong alternative for future development, but not carried forward here.

Table 3: Selection boundary

Must-have requirement	Rule-based	Codified KMS	CBR/knowledge graph	RAG
Local data boundary	Yes	Yes	Yes	Yes
Dutch capture and retrieval	Yes	Yes	Yes	Yes
Multi-modal input	No	Yes	Yes	Yes
Usable by non-technical operators	Partial	Yes	Yes	Yes
Validation gate	Yes	Yes	Yes	Yes
Low burden on willy	No	Yes	Yes	Yes
Retrieval faster and more reliable	Yes	Partial	Yes	Yes
Cost proportionate to value	Partial	Yes	Yes	Yes
Outcome	Not viable	Viable	Viable	Viable

This leaves two directions: codified knowledge management system and RAG. Both pass the knock-out step, with the codified KMS scoring only partial on retrieval. With current technology the two are closer than they look, because in both cases an AI layer can turn free text, speech or media into structured fields automatically (Cho, 2026), so the codified approach no longer forces operators to fill in a rigid form by hand. So the deciding difference is in retrieval. A codified knowledge system finds entries through exact fields, filters and keywords, which works only when the searches uses the same terms as the stored entry. Semantic retrieval matches on meaning, so an operator who describes a problem in their own wording or dialect reaches the right entry, which the Chapter 3 finding that operators and source material describe the same problem in different terms makes decisive. This points to semantic search as retrieval mechanism. RAG builds on that retrieval by adding a generation step: a language model formulates a single readable answer in the operator’s own language. The semantic retrieval delivers the right

knowledge, the generation layer makes it directly usable on the floor. RAG is therefore selected as the solution direction

5.4 Prototype Design

Section 5.3 selected RAG as the solution direction. This section describes the design of the prototype that was built on that basis. The focus is not only on what was built, but on why each choice was made, because almost every choice is a direct answer to a constraint of Huima's real situation: the operators who use the system, the hardware it runs on, the nature of the data, and the requirement that the system never show a guess as if it is a fact. These constraints are stated first as design drivers, because they come back through the whole section as the reason for individual choices. Code-level details, like the full prompts, SQL statements and endpoint specifications, are documented in the appendices.

5.4.1 Design Drivers

Six constraints shaped the design. They follow directly from the requirements in Section 5.1 and from the practical situation at Huima.

First, tacit knowledge must become explicit and reusable. The core problem is that troubleshooting knowledge mostly sits in the heads of experienced employees and is never written down. The system therefore needs a low-effort capture path – an operator can speak, photograph a note or type – and a retrieval path that returns this knowledge when a colleague faces the same problem later. This is the knowledge-management goal behind the whole prototype.

Second, the human remains the authority. The AI never has the final word on what enters the knowledge base: it proposes a transcription or a structure, and a human can confirm or correct it. The strength of this check is different per input type: speech and OCR input pass a strict validation gate before it becomes searchable, while typed input remains human-editable without such a gate, as described in 5.4.4. This implements the validation-gate requirement from Section 5.1 and follows the member-checking logic of expert knowledge elicitation, where captured knowledge is verified rather than trusted on first entry (Hoffman et al., 1995).

Third, the system must not produce unsupported answers. On a factory floor, a confidently wrong answer is worse than no answer, because it can send an operator in the wrong direction on a real machine. The design therefore treats "I cannot find this in the available knowledge" as a normal, expected outcome rather than a failure.

Fourth, the prototype runs on modest, shared hardware: a single machine with roughly 6GB of video memory. The local transcription model (about 3GB) and the two local models used for retrieval and answer generation (about 6 GB together) do not fit on the GPU at the same time. This single hardware fact dictates the concurrency model (a strict processing queue), the placement of the transcription model on the CPU and the model-warming strategy described in 5.4.7.

Fifth, Huima's accumulated knowledge must remain local. As stated in Section 5.1, management permits cloud processing of newly created operator input, but the contents of the knowledge database may not be exposed to external services. The design draws this boundary clearly: only raw input that an operator has just created (their own photo, their own typed text, their own audio) may be sent to a cloud model, while retrieval, similarity search and answer generation over the knowledge base run locally. One nuance to this boundary is in 5.4.4.

Sixth, the system has to degrade gracefully. Transcription, OCR[11] and model services can and will fail – a missing API key, a rate limit, an unreachable model. A failure may never lose operator input and may never crash the system. At worst, a task is marked as failed and can be retried.

5.4.2 System Architecture

Figure 5 shows the troubleshooting process once the Kennisapp is in place. The rest of this section describes the architecture behind it.

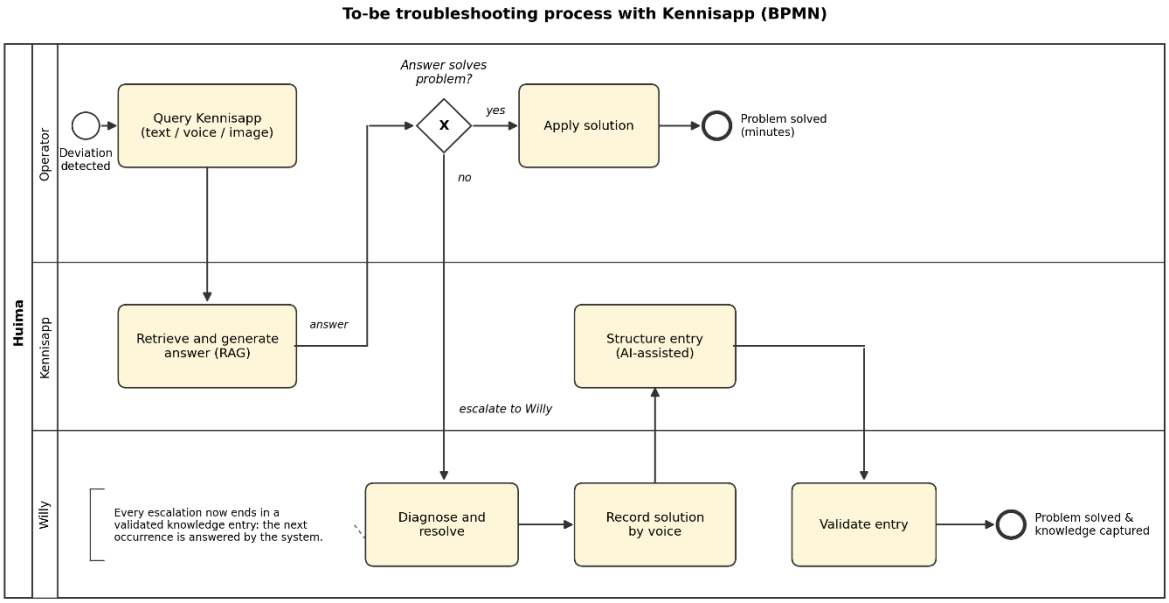


Figure 5: To-be process model

The prototype consist of several components that work together. The frontend is a progressive web app (PWA) that operators open in a browser or install on a phone. A PWA was chosen over a native app because it installs without an app-store, updates instantly and works across devices, which fits the mobile-friendly requirement. The trade-off is that a PWA cannot use every native phone feature, but the prototype needs only the camera and microphone, which the browser already provides. A separate desktop management webapp provides a read-and-edit view of the knowledge base for office use. The frontend contains interface logic only: it holds no secret keys and has no direct access to the database or the model services. The backend is a single Node.js/Express[12] API that owns all server-side logic. This API is the trust boundary of the system. It is the only component that holds secrets. The cloud API key is stored on the server and is never sent to the browser, and the browser only communicates with this API, never directly with the cloud service or the database. Keeping the trust boundary in one place makes the data-governance claim checkable instead of assumed.

Storage uses PostgreSQL with the pgvector extension[13], so one database holds both the readable knowledge and the embedding vectors[14]. Semantic search then becomes a normal SQL query ordered

[11] OCR stands for Optical Character Recognition, it is software that reads text from images or scanned documents.
[12] Node.js allows JavaScript to run outside the browser. Express is used to define the API routes that the frontend communicates with.
[13] A PostgreSQL extension that adds vector storage and vector similarity search to the database.
[14] A numerical representation of text in which texts with similar meaning receive similar numbers.

by vector distance[15], rather than a separate vector-database service. The alternative, a dedicated vector database like Pinecone next to the main database would add retrieval features the prototype does not need at this scale, and it would add a second system to install, back up, secure and understand. For a small manufacturer without IT staff, avoiding that second system is a deliberate simplification that supports the maintainability and cost requirements. The accepted trade-off is that this would have to be revisited if the knowledge base ever grows to a very large size.

Local models run through Ollama[16], which hosts both the embedding model and the generation model. Because of this the retrieval and answer-generation paths have no cloud dependency at all, and that is what makes the local data boundary work in practice. Google Gemini and OpenAI are used only for three tasks on newly created raw input: OCR of photographed notes, structuring of free text into fields and primary transcription of voice notes. A local Whisper[17] model (faster-whisper) serves as transcription fallback when the cloud path fails. Finally, the server is reachable only over Tailscale[18] private network. An earlier version of the application used a cloud database (SupaBase). The move to a fully local API on a private network was a deliberate redesign and is the infrastructure side of the data boundary. The knowledge base sits on-premises, and only an authenticated device on the private network can reach it.

Authentication is intentionally lightweight: a name and PIN, with access tokens that are valid while the server runs. For a single-site prototype with a handful of named users this is enough. The accepted risk is that credentials are not yet stored in hashed form[19](they are not cryptographically transformed before storage), and that sessions do not survive a server restart. This is only acceptable because the server is reachable solely over the private network. Hardening (hashed credentials, persistent sessions) is in the implementation advice rather than built now, and this scope decision is acknowledged again as a limitation in 5.4.8. The full architecture overview, including a route-level description of the API and the design reason for each route, is in Appendix C.

5.4.3 Data Model

All knowledge converges on one central table, LogBoekKennis (“knowledge logbook”), in which one row represents one knowledge point. Two child tables are attached to it through a reference field: foto for photos and spraaknotitie for voice notes. A knowledge point can have any number of attached photos and voice notes.

The most important column of a knowledge point is “originele_notitie”, the raw text. All readable input accumulates here: typed text, OCR text from photographed notes and, after validation, transcribed speech. This field is the raw record of what was originally written or said, it is not itself embedded. From this raw text, a language model (currently Gemini 2.5 Flash) derives the structured fields: problem, suspected cause, what was tried, solution, suggestions and error code(s). The embedding for semantic search is made from these structured fields (see 5.4.5), so searchability goes through the structuring step. The knowledge point also stores its embedding vector and its context: which machine it concerns, who added it and when. This context is what later allows source attribution at retrieval.

This parent-plus-attachments shape was chosen for three reasons. First, it keeps one incident together as one source of truth: a combined report consisting of some typed text, a photo and two spoken remarks

[15] Here cosine distance: a measure of how dissimilar two embeddings are; lower means closer in meaning.
[16] Software that runs open-source AI models locally on the company’s own machine.
[17] An open-source speech-to-text model; faster-whisper is a local implementation of it.
[18] A service that creates a private, encrypted network between approved devices only.
[19] Hashing means storing a one-way cryptographic representation of a password or PIN instead of the original value. This way, even if the stored credential data is exposed, the original PIN is not visible.

is a single knowledge point, not three disconnected records. Second, because all input text lands in one raw field and, through the structuring step, is condensed into one embedded vector, a knowledge point is retrievable no matter if its content originated as typing, handwriting or speech. After capture the input type does not matter anymore, only the meaning is searched. Third, the shape reuses the existing data model: photos were already attached to a knowledge point, and voice notes were attached in the same way, so existing search and structuring kept working without changes.

Two smaller modelling decisions are worth explaining. The two attachment types are linked in different ways on purpose. Photos use a hard, database-enforced foreign key with cascading delete (ON DELETE CASCADE): a photo cannot exist without its knowledge point, and deleting a knowledge point removes its photos. The link between a voice note and its knowledge point is kept flexible on purpose: a soft, nullable reference that the application maintains, rather than a hard foreign-key constraint. A stricter relationship would block unattached voice notes, but standalone voice memos must remain possible and a hard constraint would also make deletion more complicated. The consequence is that the database itself does not guarantee referential integrity for voice notes, so deleting a knowledge point can leave orphaned voice notes behind. For a prototype, the flexible option is acceptable, a stricter and more normalized database design is therefore listed as future work. Second, a situation photo – a picture of the machine itself for example – is stored and viewable as evidence, but it is not made text-searchable, because it contains no text. Only a photographed note contributes, after OCR, to the raw text and, through structuring, to retrieval. In other words, the system only searches what it can actually read. The full table structure, the role of each field and an entity-relationship overview is in Appendix B.

5.4.4 Knowledge Ingestion and Validation

Ingestion is the process in which tacit knowledge becomes an explicit, searchable record – the externalization step of the SECI model made concrete in software (Nonaka et al., 2000). The pipeline is the same for every input type: raw input → OCR or transcription where needed → raw text → LLM structuring into fields → human check → embedding → stored and searchable. The three inputs differ only in their first steps.

Typed text is the simplest path: the operator or Willy types a free-text description of a problem, which goes directly to the structuring step.

A photographed note follows the OCR path. All pages of one handwritten note are sent to the cloud model (Gemini 2.5 Flash) in a single request as ordered pages, so the model reads them as one document and returns one continuous text. The OCR instruction is constrained in two ways: it must transcribe only the handwriting and ignore the pre-printed text of the standard logbook-note form, so that it does not pollute the knowledge, and it must mark every uncertain reading – illegible handwriting, ambiguous words, crossed-out-text – with a visible marker. So instead of a flat transcription that hides its own confidence, the OCR shows where it is unsure. The interface highlights the markers for the person correcting the text, and they are removed before structuring. The OCR prompt logic is in Appendix D.

Voice notes got the most engineered path, because speech is the richest source of tacit knowledge: Willy will say considerably more than he is willing to type, which is why voice capture was classified as a must-have in Section 5.1. A voice note is stored immediately with a queued status, and Willy or the operator can continue working, because the transcription runs in the background and not synchronously. A strict one-at-a-time queue processes the notes, which is the direct answer to the hardware constraint: two heavy AI tasks never run simultaneously on the shared GPU. Transcription uses a primary/fallback strategy. The primary engine is OpenAI (model: gpt-4o-transcribe). During capture testing the initially chosen cloud engine (Gemini) returned transcription errors on a non-trivial share of voice notes, so the

primary engine was changed to OpenAI, which transcribed the same difficult audio reliably. If OpenAI fails for any reason, Gemini is tried; if both engines fail, a local run Whisper model takes over on the CPU, in a separate subprocess, so that it cannot push the retrieval models off the shared GPU. If all three engines fail, the note is marked as failed with the reason and can be retried later, it is never lost. The engine that made each transcription is recorded per note, which makes engine comparison possible. The complete transcription design, including the fallback logic and the tuning choices, is in Appendix F.

A raw transcription does not become knowledge automatically. The speech is stored first and validate before it becomes searchable. Storing it on arrival makes sure that no input is lost, while the validation status guarantees that no unverified machine output enters the searchable knowledge. Only an explicit allow-list of users (for now Willy and the researcher) may correct the text and approve it. Only after approval is the text added to the raw text of its knowledge point, after which the point is re-structured and re-embedded so that the spoken contribution becomes findable. A knowledge point therefore grows with each validated spoken addition, and the human remains the gatekeeper throughout. This implements the validation-gate requirement in the refined form stated in Section 5.3 –transcribed speech is checked by a human before it enters the searchable knowledge base, OCR text passes through mandatory human correction of the marked uncertainties before structuring, while typed input is human-authored and remains editable without an approval step. Together this is the system's member-checking mechanism (Hoffman et al., 1995).

Structuring is shared logic for all inputs. The cloud model converts the raw text into the fixed fields under tight constraints: deterministic settings and a forced JSON[20] output format. An explicit instruction that a field may only be filled with information literally present in the text. Absent information becomes an empty field, and the model returns a separate list of fields it was uncertain about, so the interface can flag those for the human. Structuring stores nothing by itself, saving is a separate step and the structured fields remain fully human-editable. In the earlier form-based input path the operator reviewed the proposed fields before saving. In the current combined capture flow, structuring runs in the background after saving, so the operator corrects the fields afterwards rather than being blocked on review. By letting the model propose the structure, the operator does not have to translate their own words into fixed fields themselves, which lowers the ability barrier that the MOA framework identifies as a condition for successful externalization (Gavrilova & Andreeva, 2012), while the member-checking principle is kept: the model proposes, the human confirms. The structuring prompt logic is in Appendix D.

It is worth addressing directly why a cloud model is used here while retrieval and generation are local. OCR, structuring and primary transcription all operate on raw input that the operator has just created – their own photo, their own words. Sending that to a cloud API falls within the boundary that management approved. The knowledge base is the accumulated, shared asset of the company and is never used as context for cloud retrieval or cloud answer generation: query embedding, similarity search and answer generation all run entirely local. One nuance has to be stated: when a validated voice note is merged into an existing knowledge point, the combined raw text of that knowledge point – the new speech together with the previously stored raw text, all of it operator-authored input – is re-sent to the cloud structuring model. The operating rule is therefore: the cloud may process raw input that operators have created, including the re-structuring of a knowledge point's own raw text after validation, but the collective knowledge base never feeds a cloud model at question time. This rule is documented at every relevant place in the code, which makes the governance claim auditable instead of just a promise.

[20] JavaScript Object Notation is a structured text format used to represent data in predefined fields and values

5.4.5 Embedding and Retrieval Design

Semantic retrieval is necessary because of how operators ask questions. Chapter 3 showed that operators and source material describe the same problem in many different wordings, including dialect and personal shorthand. A keyword search only finds an entry when the searcher happens to use the same words as the writer, semantic search compares meaning, so a differently phrased question can still reach the right entry. This was the deciding advantage in the solution selection, and it is what makes the stored knowledge accessible in the operator's own language, directly serving the must-have requirements of Dutch-language retrieval and usability for non-technical operators.

For each knowledge point, a composite text is built by joining its filled structured fields (problem, suspected cause, actual cause, what was tried, solution, suggestions), and this composite text is embedded into a single vector. The same text-generation function is used both for embedding old entries and for new entries. This consistency is important because semantic retrieval compares vector representations in embedding space; differences in how source text is constructed may add variation that has nothing to do with the actual knowledge content, and it would lower retrieval reliability (Pinecone).

Embeddings are made at write time. When a knowledge point is created or edited, it is embedded immediately so that it is findable at once, and re-embedded on every edit so that it is not retrieved on old text. If the embedding model is unreachable, the embedding is left empty and the knowledge point is still saved – losing the knowledge would be worse– and a backfill fills the gap later.

The embedding model itself, bge-m3 (1024 dimensions[21]), was selected over the initially used nomic-embed-text (768 dimensions) through a small side-by-side comparison. The same composite texts were embedded by both models into separate database columns, so both could be evaluated on identical data. On Dutch workshop text, the multilingual bge-m3 separated the genuinely relevant entry from irrelevant ones more clearly than the English-oriented nomic-embed-text. The trade-off is much higher memory use, and switching embedding models is never free: it changes the vector space and forces all embeddings to be regenerated (Qdrant). The comparison setup and the measured values are in Appendix E.

Retrieval itself involves two design decisions. First, the machine is the primary filter: almost every question is about a specific machine, so filtering on machine before ranking by similarity strongly reduces noise. A question without a machine searches across all machines. Second, a distance threshold turns "nearest" into "relevant". A similarity query always returns something, because the nearest entry exists even when it has nothing to do with the question. So results above a maximum cosine distance of 0.50 are dropped and the system reports that no good matches exists. The threshold was derived from small comparison tests, where correct matches landed clearly below it and the first irrelevant entries above it. It is clearly a starting value: it can only be tuned reliably on a well-filled knowledge base, and with the current small dataset it is provisional. Retrieval is hybrid: the vector search runs in parallel with a lexical whole-word search (for jargon, machine specific names, abbreviations etc.) and the error code route described below. These three ranked lists are merged with Reciprocal Rank Fusion (Cormack et al., 2009). Section 6.4 describes the decision applied to the fused result.

Error codes are handled separately from semantic search, because an exact code requires deterministic matching: a query for code 227 must find exactly the entries that mention code 227, not entries that are semantically nearby. The error-code field is free text and sometimes contains several codes in one cell (for example "272 and 227"), so a simple equality check would miss those rows. The code is therefore matched as a whole number inside the field which finds 227 inside "272 and 227" but not inside 1227

[21] Embedding dimensions are numerical values that represent a text in vector space.

or 2270 for example. When no exact hit exists, the system proposes the neighboring codes as fallback, because related faults often sit on adjacent codes. A normalized error-code model with one code per row is the cleaner relational design, because it avoids storing multiple values in a single field and improves integrity, querying and future scalability. However, for this prototype the existing free-text structure was retained to avoid migrating original data during the thesis period. The current approach is therefore a compatibility choice, while normalization is recommended for future development.

5.4.6 Answer Generation and Safeguards

The question-answering endpoint does the full RAG cycle: it retrieves the most relevant entries and lets a local generation model (Qwen2.5 7B via Ollama) write one coherent answer. A larger cloud model would produce more fluent answers quicker but it would require sending retrieved knowledge to an external service, which the data boundary forbids. The weaker local model is the accepted price of keeping generation local. Retrieval logic is shared with the plain search endpoint, so the same retrieval behavior can be evaluated with and without generation in the validation chapter. Generation runs with deterministic settings to reduce variation, and each retrieved entry is passed to the model as a numbered fragment listing its structured fields plus who added it and when.

The anti-hallucination design has two layers, built so that "I do not know" is structurally easy for the system to reach. The first layer is that without context there is no model call at all: if retrieval returns nothing under the distance threshold, the model is not called and the endpoint returns a fixed answer stating that the information cannot be found in the available knowledge. No context means no opportunity to invent. The second layer is prompt-enforced abstention: when fragments are present but do not contain the answer, the model is instructed to return that same fixed sentence, to use only the supplied fragments, and to add no outside knowledge. Both layers implement the third design driver – the system may say that it does not know, but it may never present an unsupported answer as a fact – and both produce a clean, observable "no answer" outcome, which is itself useful evidence in the validation, because it shows the guardrail working. The prompt logic that enforces this behavior is in Appendix D.

The answer is shaped as advice rather than instruction. The model must present the possibilities from all relevant fragments instead of dictating one fix, and must distinguish between what was tried and worked, what was tried without success, suggestions and error codes. This mirrors how the knowledge base actually grows – multiple people's attempts, not one canonical solution – and keeps the system in its intended advisory role.

Source attribution is decoupled from the model. Self-citing language models can struggle to assign generated claims to the correct supporting sources, which makes model-generated attribution unreliable in settings where traceability matters (Qi et al., 2024). Because who tried what, and when, matters to the users, source attribution itself is never generated by the model. The model may mention names and dates in its text, but guaranteed source list is rendered directly from database records by the application. That list cannot be hallucinated, because it does not come from the model. Each numbered source is clickable and opens the original knowledge point, including any photographed original note, so an operator can always check the evidence behind an answer.

Finally, vague questions trigger clarification instead of a forced answer. A general question such as "fault while running" can return matches under the threshold while none of them is truly on target, or several that are all equally half-relevant. The system detects this from the retrieval distances, not by asking the model to judge its own relevance: either the best match is still relatively far away (nothing is really on target), or the top matches lie very close together (no clear winner). When either signal fires, the system

does not answer. Instead, the model is called with a separate, strict instruction to formulate exactly one focused follow-up question with two to four answer options drawn only from the retrieved fragments, which the interface shows as clickable choices that refine the search. This is like how an expert narrows a symptom toward a diagnosis, and aligns with the conversational approach identified in the literature review (Kernan Freire et al., 2023). The point is the same as elsewhere: the decision to ask back is based on a measurable signal, while the model only does what it is good at – phrasing a natural question. The threshold values behind this detection are in E.

5.4.7 Operational Robustness

The robustness measures follow directly from the hardware and graceful-degradation drivers. The one-at-a-time queue is the primary overload defense: a burst of voice notes stacks up and is processed in order, so the shared GPU memory can never be used up by parallel tasks. The local transcription model runs in a separate subprocess, so a crash on a corrupt audio file kills only that subprocess while the server keeps running and marks the note as failed. A hard timeout prevents a stuck transcription from blocking the queue. Every failure path ends in a recorded status and a preserved record rather than an unhandled error: a failed transcription never blocks saving, an unreachable embedding never blocks saving, and an overloaded cloud API is retried with increasing delays before the user receives a clear message. On startup, half-finished notes are re-queued, which is safe because transcription can be repeated without side effect, and audio uploads are checked on size and file type before any processing starts. Finally, the slowest single step in answering turned out to be loading a model from disk (about 16 seconds for the generation model), not the search itself, so both local models are pre-loaded at server start and kept in memory, which keeps the next questions fast.

5.4.8 Limitations of the Prototype Design

Several limitations remain and are acknowledged here. The knowledge base is almost empty, which means the retrieval thresholds are provisional cold-start values and retrieval quality can only be demonstrated on the representative cases used in validation, not claimed across the full breadth of Huima's problems. Authentication is prototype-grade and would need hardening before broader use, and the management webapp has no full production-grade authentication and relies on the private network for access control. Because the raw text is not directly embedded, retrieval quality depends on the quality of the structuring step. Voice notes are linked to their knowledge point without a database-enforced foreign key, so deleting a knowledge point can leave orphaned voice notes. The local 7B generation model is weaker than larger cloud models, this is the accepted trade-off for keeping answer generation local. The fast transcription path depends on a cloud service, so an offline situation drops back to the slower local path. Finally, the time available for validation within the thesis period limits how thoroughly the tuned parameters and the validation workflow could be tested under real daily use.

5.4.9 Summary of Design Choices

Table 4 summarizes the main design choices and connects each one to the requirements or constraint it addresses. Together they show how the design satisfies the must-have requirements from Section 5.1 within the solution direction selected in Section 5.3

Table 4: Summary of Design Choices

Design Area	***Choice made***	***Reason / requirement addressed***
Architecture	Single local API as trust boundary, browser never accesses cloud services	Local data boundary

Network	Local server on private network	Local data boundary
Storage	Pgvector inside PostgreSQL database, no separate vector database	Cost and maintainability
Data model	One knowledge point with attached photo(s) and voice note(s)	One record per incident, modality-independent retrieval
Capture	Text, photo and voice input	Multi-modal input, low capture burden and externalization
Transcription	Cloud model (OpenAI, Gemini), local Whisper fallback	Dutch accent with regional accent, graceful degradation
Structuring	LLM structures free text into fields, uncertainty flags, operators/Willy validate before saving	Low capture effort, ability condition of MOA(Gavrilova & Andreeva, 2012)
Validation	Speech stored first, validated before it becomes searchable; typed/OCR input human-editable	Validation gate, member-checking (Hoffman et al., 1995)
Embedding	Bge-m3	Dutch retrieval quality
Retrieval	Hybrid retrieval, machine filter, distance threshold	Relevant results in operator language, honest “no match”
Generation	Local model grounded only in retrieved entries. Two-layer abstention with a fixed “cannot find” answer	No unsupported answers, no cloud call at question time
Attribution	Source list rendered by code from database, not by the model	Trustworthy provenance, independent of model behavior
Clarification	Vague questions detected from retrieval distances, follow-up question with options from the fragment sources	Avoids forced, misleading answers
Robustness	One-at-a-time queue, subprocess isolation, model warming	Shared 6GB hardware (current system), no lost input

The design is consistent in one way. Wherever the system makes a judgement – whether a match is relevant, whether a question is too vague, whether knowledge may enter the database – that judgement is based on a measurable signal or in a human decision. Wherever the model is used, it is kept to what it is good at: reading, transcribing and phrasing. It is never trusted with what it is unreliable at: attribution, deciding relevance, or knowing what it does not know. That separation is the main idea of the design.

6. Validation

This chapter presents the validation of the prototype and answers the fifth and final sub-question:

To what extent does the developed system capture expert knowledge and retrieve it correctly and reliably on representative troubleshooting cases?

Where Chapter 5 described what was built and why, this chapter looks at whether it actually works. It is important to be clear about what kind of validation this is. This is a functional validation of the mechanism, not a field deployment. It tests whether the system does what it was designed to do, capture expert knowledge correctly and retrieve the right knowledge in response to a troubleshooting question, and whether it does this without producing answers that are not supported by the stored knowledge. it does not test adoption, day-to-day use on the work floor or the long-term effect on Willy-dependency, because those depend on the knowledge base being populated and on the system being in real use (on local Huima network). As set out in Section 1.4 and Section 5.1.5, populating the knowledge base and deploying the system were placed outside the project scope on purpose, they are implementation questions and are addressed as such in Section 7.2. The validation therefore shows that the mechanism works, which is the essential first step, rather than the dependency at Huima has already been reduced in practice.

6.1 What This Validation Does and Does Not Prove

Because the same researcher designed, built and evaluated the system, and because the knowledge base was populated by that researcher instead of in daily production, it is important to state the boundary of the validation. Defining the boundary explicitly is what keeps the claims in this chapter clear: every result that follows should be read as evidence for the items in the first list below, and never as evidence in the second list.

What the validation demonstrates:

- Technical feasibility. The capture-to-retrieval mechanism runs entirely on the used hardware within the local data boundary
- Correct capture. Spoken and photographed input is transcribed and structured into findable knowledge points, with errors that human validation is designed to catch.
- Correct retrieval. For knowledge that is present in the knowledge base, the system returns the right knowledge point in response to a question, including misspelling and rephrasing.
- Reliable answer generation. The system grounds its answers in retrieved entries and abstains rather than fabricating when the answer is absent.

What the validation does not demonstrate:

- A long-term reduction in expert dependency. This requires a populated knowledge base and observing the work floor over weeks or months.
- Organizational adoption. Whether operators and Willy will actually use the system in daily work is an implementation question.
- Productivity gains. Any real reduction in lost production time can only be measured once the knowledge base is populated and the system is genuinely being used.

These boundaries follow directly from the functional scope defined in Section 1.4 and from the decisions in Section 5.1.5 to place the knowledge base population and deployment outside this project. The results below are reported as demonstrations of a working mechanism on representative cases.

6.2 Validation Method

The validation evaluates the prototype against the requirements specification from Section 5.1, using real troubleshooting cases from Huima. Because the same researcher both designed and evaluated the system, the validation was set up to limit researcher bias as much as possible: with predefined cases and criteria instead of loose, ad-hoc testing (Saunders et al., 2023).

The validation consists of two parts that together cover the full mechanism. The first part is the capture validation: a demonstration, with Willy, that a spoken troubleshooting explanation can be turned into a stored, structured and searchable knowledge point. This is the externalization step the whole project is built around (Section 2.2), so it is validated end-to-end. The second part is the retrieval validation: a retrieval measurement on a set of representative cases that tests whether the system returns the right knowledge, faster, in response to a natural language question. A third smaller test checks the abstention behavior. None of these parts require the system to be in live use on the floor, which matches the functional scope set out above.

Case selection. A set of representative troubleshooting cases was fixed before testing began. These cases were taken from the logbook of multiple machines and real recent troubleshooting cases, and were chosen so that they reflect the dependency described in Chapter 3: recurring, small-to-medium configurations and operator-error problems rather than the major incidents that Section 3.2 placed outside the scope of the system. Each selected case was first entered into the system as a knowledge point, so that a known, correct answer exists in the knowledge base. The selection criteria were: a problem confirmed by Willy to be resolved before, the machines 18MS, I200 and the I400, of which most test data exists, if data allows a problem that can have multiple solutions.

Capture validation (with Willy). To validate the capture path, Willy recorded multiple real troubleshooting or instruction cases as voice notes, both in normal Dutch and regional Twents accent as well as machine specific jargon. For each note, it was checked whether the full chain worked: the note is transcribed, the raw text is structured, the entry follows the human validation step and the resulting knowledge point becomes retrievable. The point here is not retrieval speed, but completeness and correctness of the captured speech entry: does what Willy said end up in the system as a usable, findable knowledge point, and is the structured result authentic to what he actually meant. Correctness of the structuring was checked together with Willy himself, which doubles as the member-checking step from the elicitation literature (Hoffman et al., 1995) and is consistent with how Yoshida et al. (2011) confirmed captured behavior by showing it back to the expert.

Retrieval measurement. For each case, the problem is described in natural language and the system's retrieval outcome and time are recorded. The current process is not re-measured per case, the comparison baseline is the 15-30 minutes search time established in Chapter 3. This is an approximation, not a controlled head-to-head benchmark, and is treated as such in Section 6.6.

The following were measured per case:

- Retrieval time – the time from starting the search to reaching a usable answer. This operationalizes knowledge retrieval efficiency (Section 2.5)
- Retrieval success – whether a relevant, correct answer was reached at all. This operationalizes retrieval success rate.
- Retrieval relevance – whether the returned knowledge point was the correct one for the query. This operationalizes the quality of semantic retrieval (Section 2.5).

Who carried out the tests: both the capture and retrieval tests were carried out by the researcher, together with Willy for the capture part. Operators were used as informal test subjects in order to gather feedback on the prototype which supports the implementation Chapter. Operators were not used as strict test subjects in this phase. This is the consequence of the functional scope: the tests check whether the system retrieves the right kind of information for a given question and whether the capture chain works, neither of which requires an operator.

Separately from the captured cases, the system is also tested with questions whose answer is not in the knowledge base, to check the two-layer abstention design from Section 5.4.6. A correct outcome here is the system returning its fixed answer that the information cannot be found in the available knowledge, instead of some made-up answer. A working "no answer" is a positive result and not a failure, because on the factory floor, a confidently wrong answer is worse than an honest no-answer.

6.3 Capture Validation

This chapter validates the system against the research questions and the action problem. The questions contain five central concepts: whether the system retrieves knowledge correctly, whether retrieval time is improved, whether Willy is consulted less frequently, whether expert dependency is reduced, and whether knowledge accessibility is improved.

Two of these concepts cannot be measured at the current scale of the prototype. Reducing expert dependency can only be demonstrated by observing the workplace over a period of weeks or months, which falls outside of the available time. A controlled before-and-after over that horizon is the only way to establish a real reduction in dependency. The closely related reduction in expert consultation cannot be measured either, because the knowledge base is not yet populated. Once it is, reduced consultation could be measured over a longer period. These two concepts are therefore addressed by design argument rather than by measurement, which is stated here explicitly as limitation of the validation.

This leaves three measurable concepts: whether the system retrieves knowledge correctly, whether it reduces retrieval time, and whether it improves knowledge accessibility. Correct retrieval is a must-have requirement of the system, and several safeguards protect the retrieval result. These were described in Chapter 5. In short they are the validation gate, the abstention gate, the warning on uncertain transcriptions and a vector-distance threshold together with a warning that the answer is generated by an AI model.

6.3.1 OCR Validation

Before retrieval can be validated, the captured input must enter the system correctly. Capture validation therefore checks whether an input becomes a correct, structured and findable knowledge point. For each entry, four checkpoints are assessed: whether the text was read correctly, measured as word-error rate (WER) against a manual reference, whether the structuring step produced filled field, whether the entry was confirmed as authentic by a validator, and whether it became findable through search after embedding.

The OCR capture channel was validated on nineteen knowledge points created from photographed notes. Table 5 summarizes the four checkpoints. Table 6 gives the word-error rate per entry.

Checkpoint	Result (n/N)
Read correctly (WER = 0)	0/19 (median WER 8.8%, mean 21.7%)
Structured into filled fields	19/19
Authenticity confirmed by validator	19/19
Findable after embedding	19/19

Table 5: OCR capture checkpoints

None of the nineteen entries was read with a word-error rate of zero, but this figure is misleading on its own. The median WER was 8.8% while the mean was 21.7%. The gap between the two shows that the overall error is driven up by a small number of poorly readable sources rather than by a general inability to read the notes. The high-WER outliers (for example 61.6%, 57.9% and 70.0%) all correspond to handwritten notes whose handwriting is so poor that even a human reader struggles to decipher them. For these cases the limitation lies in the source material, not in the OCR step.

Entry id	Machine	WER	Word errors/total
4	18MS	17.5%	10/57
5	18MS	61.6%	85/138
6	18MS	57.9%	132/228
7	18MS	32.5%	13/40
8	18MS	8.1%	10/124
9	18MS	8.0%	4/50
11	18MS	1.4%	1/73
24	18MS	51.0%	52/102
27	I400	21.2%	7/33
35	18MS	1.6%	2/128
36	18MS	5.7%	6/106
40	18MS	1.7%	2/118
42	I200	3.5%	5/141
44	I200	8.8%	11/125
46	I200	70.0%	63/90
47	I200	8.3%	3/36
48	I200	22.6%	14/62
52	18MS	1.6%	2/123
53	18MS	27.7%	33/119

Table 6: Word-error rate per OCR entry

Taken together, these results show that OCR capture is usable but not reliable enough for unchecked storage: the raw OCR quality varies, and handwritten sources in particular produce high error rates. The human validation gate is therefore essential, since it allows the OCR output to be corrected before an entry becomes searchable.

Why this matters and what remains. This result satisfies the multi-modal input must-have for the image channel: photographed notes do become structured, findable knowledge points. It matters because handwritten logbook notes are one of the two richest existing knowledge sources at Huima, so an unusable OCR path would have cut the system off from that source entirely. The limitation remains that the raw OCR accuracy is source-dependent and cannot be trusted blindly. This means that the capture channel depends on a human being willing to correct the text, which is an adoption assumption instead of a proven behavior.

6.4 Retrieval Validation

When an operator submits a question, the application sends it to a single backend endpoint, referred to here as the /ask route. This route carries out the full question-answering process: it retrieves the most relevant knowledge points from the knowledge base and, when a usable result is found, it has a local language model formulate one answer from that result. The route returns one of three outcomes, defined

in Table 7, this measurement validates the retrieval step in particular: whether the system finds the correct knowledge point, and whether it correctly abstains when the answer is not present.

Twenty-one test questions were formulated against twenty-five knowledge points present in the database at the time of measurement. The questions are deliberately phrased in operator language so that the retrieval is tested on meaning rather than exact word overlap.

The /ask route uses hybrid retrieval: three searches run in parallel and their ranked result lists are merged with Reciprocal Rank Fusion (RRF) (Cormack et al., 2009). The vector channel measures semantic similarity through the bge-m3 embeddings, the lexical channel performs whole-word matching, which captures jargon, abbreviations and part names, and the error-code channel performs an exact whole-number code match within the free-text field. The selected machine acts as a hard filter on all three channels. The decision of what the route then does with the fused result is governed by the cosine distance of the best candidate, the number of candidates and the presence of a strong signal as defined below.

Cosine distance	**Outcome**
>0.50	No usable result, the system abstains
0.42-0.50 AND 2 or more matches AND no strong signal	The system asks one clarifying question
<0.42 OR one single match	The system answers directly

Table 7: cosine thresholds

A strong signal is a Boolean condition, true when at least one of two predicates holds. The first is an exact error-code match: every three- or four-digit integer is extracted from the query, and at least one occurs as a whole number in a knowledge point's error code field, so 227 matches within "272 and 227" but not within 1227. The second is a complete lexical match: a knowledge point contains, as whole words, all query tokens of at least two characters across its combined text field. When a strong signal is present, the match is treated as unambiguous and the clarifying question branch is bypassed, the route then answers from the best vector match if one lies below the 0.50 distance threshold, otherwise abstains.

For each test question the following quantities are recorded. The two retrieval-quality measures, hit@1 and hit@k, follow the standard definitions used in information retrieval. Hit@1 is true when the knowledge point that was expected to answer the question is returned as the single best result. Hit@k is true when the expected knowledge point appears anywhere in the returned list of results. The remaining quantities describe the system's behavior and timing. The full set is given in table 8.

Metric	**Meaning**
Status	The route's outcome: answers, asks a clarifying question or abstains
hit@1	Expected knowledge point returned as best result?
hit@k	Expected knowledge point present anywhere in returned results?
Cosine	Cosine distance (<0.50 is usable)
ms	Retrieval and generation time
Abstention	Did the system correctly abstain on a question whose answer is not in the knowledge base?

Table 8: definitions

6.4.1 Test Set

The test set contains three groups of questions: questions that should be answered, robustness questions that probe the limits of the retrieval, and questions that should be declined. Each row lists a case label (an identifier such as G1, used only to refer to the case), the question as an operator would type it, the machine filter, the expected knowledge point (as in the database). The final column states what the case is designed to test.

Two terms recur in that column. A semantic match is found by meaning, through the vector channel, so it works even when the operator uses different words than the stored note. A lexical match is found by exact words, such as a part name or abbreviation. The third channel matches error-codes as exact numbers.

Group 1 – questions that should be answered

ID	Question	Machine	Expected knowledge point	What this case tests
G1	Smeeralarm van de leibanen blijft steeds terugkomen	18MS	6	Finding the note by meaning when no error-code is given
G2	Hoofdschakelaar valt steeds uit, relais QF11 springt eruit	18MS	4	Combining an exact part name (QF11) with meaning
G3	Spindel maakt een piepend geluid bij het opstarten	18MS	18	Finding the note purely by meaning (rephrased)
G4	Freesspil koppeling sensor fout 251	18MS	36	Matching an error-code (251) without confusing it with nearby alarms 227/272
G5	Magneetschakelaar relais knop blijft ingedrukt hangen storing 350 351	18MS	8	Matching a code when several codes are stored in one field (“350,351”)
G6	Motor van de schroefaandrijving is kapot, voedingskabel stuk	I200	42	Finding the note by meaning on a different machine
G7	Machine staat stil, SSD vervangen en mazak image teruggezet	I200	47	Matching technical jargon (SSD, MAZAK image)
G8	Turret staat scheef na een botsing	I400	27	The machine filter: the same problem also exists on 18MS, the right machine must win
G9	Robot groep 2 en 3 false nod established	I200	46	Matching literal jargon by exact words
G10	Alarm 227, handmatig gereedschap wisselen lukt niet, parameter SP7	18MS	11	Choosing the right note when the code 227 appears in several notes
M2	Product is verkeerde maat, halve millimeter te diep gemaakt	any	29	A quality problem rather than a machine fault

Table 9: answerable questions

Group 2 – robustness questions

These cases test how the retrieval behaves under imperfect or minimal input: a vague question, spelling mistakes, a synonym and a bare code without surrounding words.

Case	Question	Machine	Expected KP	What this case tests

M1	Machine smeert niet goed	18MS	6	A vague question: the system may ask a clarifying question, but the right note should still rank first.
T1	Spindel maakt piepnd geluid bij opstten	18MS	18	Spelling mistakes: does the model still find the note when exact matching fails
T2	Hoofdschakelr valt steeds uit door relais	18MS	4	A spelling mistake in a key word
S1	De geleidingen worden niet gesmeerd	18MS	6	A synonym ("geleidingen" for "leibanen"): finding the note by meaning with no shared words.
E1	260	18MS	6	A bare error-code without other words: the smallest possible input

Table 10: robustness questions

Group 3 – questions that should be declined

For these questions the answer is not present in the knowledge base, so the correct behavior is to abstain rather than to produce an answer. They are split into two types. Absent-topic questions (A1-A3) concern subjects that are entirely outside the knowledge base. Near-miss questions (N1, W1) resemble something in the knowledge base but are not actually answered by it, which is the harder test.

Case	Question	Machine	What this case tests
A1	Hoe vraag ik mijn vakantie aan	-	A subject outside the knowledge base
A2	Errorcode 8888 op de machine	18MS	A code that does not exist
A3	Wat is het wachtwoord van de wifi in de kantine	-	A completely off-topic question
N1	Gereedschap breekt steeds af in de turret	18MS	A question that sounds like a fault but is not in the knowledge base: the system must not invent an answer
W1	Spindel piept bij opstarten	I400	A known problem asked on the wrong machine: the machine filter must force abstention

Table 11: unanswerable questions

6.4.2 Results

Metric	Result
hit@1 (answerable cases)	14/16 (88%)
hit@k (answerable cases)	14/16 (88%)
Correct abstention (true abstention cases)	4/4
Fabrications (answer invented while abstention was required)	0 (1x clarifying question instead of abstaining)
Retrieval + generation time (answer cases, warm)	Median 4.7s – range 3.0-9.8s (n=12)
Cosine top-1 on correct hit	Median 0.283 – range 0.167-0.427 (n=12)

Table 12: Summary of retrieval and abstention results

Interpretation

Of the sixteen answerable cases, fourteen produced a correct hit@1 (88%). The two misses are the synonym case (S1) and the bare code case (E1), which are discussed below. The cosine distances on all correct hits are well below the threshold value of 0.50 (median 0.283; all ≤ 0.427). Typographical errors were absorbed (T1, T2 both produced hit@1): even with the misspellings the system retrieved the correct knowledge point. The hard machine filter functioned as intended: in G8, it retrieved id 27, belonging to the machine I400, rather than the id 7 which belongs to the machine 18MS, and in W1 the same problem queried on the wrong machine led to abstention. Off-topic questions were rejected without exception

(A1, A2, A3) and no answer was fabricated, consistent with the intended design preference for abstention over an unsupported answer. The vague question (M1) led to a clarifying question rather than a direct answer, with the correct note still ranked first among the candidates.

Together these results satisfy several must-have requirements directly. The 88% hit@1, at a median of 4.7 seconds against the 15-30 minute search time from Chapter 3, supports the 'retrieval faster than the current process' requirement for captured knowledge, though without a controlled head-to-head measurement (Table 15). The sub-0.427 cosine distances show the semantic-retrieval mechanism separating relevant from irrelevant entries cleanly. The 4/4 correct abstention with zero fabrications demonstrates the anti-hallucination safeguard and with it the design decision that the system may never present an unsupported answer as fact. What is not demonstrated here is that these figures hold on a fully populated base or on questions the researcher did not author, which is stated as threat in section 6.6.

The two missed cases

S1 (synonym). "geleidingen worden niet gesmeerd" found nothing, although it concerns the same matter as note 6 ("leibanen smering"). The two phrasings share no words, so the lexical channel cannot match them, and the match therefore depends entirely on the vector channel.

E1 (bare code "260"). The error-code channel does locate note 6, because the number 260 occurs in its text. The answer is built from the best result of the vector channel, not the error-code channel. A query with only a single three-digit number produces an embedding with almost no meaning, so its vector distance stays above 0.50. As a result the system abstains. The cause is therefore not that the note is missing but that the bare code carries too little linguistic context for the vector channel, which is the channel currently in control of the answer.

N1 (near-miss). The question "gereedschap breekt steeds af in de turret" describes a plausible fault that is not present in the knowledge base. The system did not invent an answer: it asked a clarifying question with the best candidate at distance 0.462. The content of the clarifying question, however, was not useful. The options offered to the user were drawn from several unrelated knowledge points rather than from variants of the same problem. The clarifying question is built from the top candidates without first checking whether those candidates actually describe the same problem, so when the candidates are unrelated the resulting options do not help the user narrow the question. This is a limitation of how the clarifying question is composed, not of the retrieval ranking.

Recommendations

1. Use the synonym/jargon layer in retrieval (S1). A jargon list already exists in the management app: enriching the query or note text with it would let synonyms without word overlap still be found
2. Let error-code hits count towards the answer (E1). Let /ask use a strong error-code match as the source even when the vector signal is weak, so that a bare code still leads to an answer (though, this error-code search is something that is easier in normal search).
3. Compose the clarifying question only from the candidates that are close to each other. A clarifying question is only useful when several candidates are genuinely comparable, that is when two or more results lie within a small distance of each other (for example within 0.05).

Limitations

- Small knowledge base (25 knowledge points): cosine thresholds are starting values and can only be calibrated reliably once a larger dataset exists.

- Single measurement moment: the measured times are warm-state values from one run. An average of multiple runs would be cleaner.
- Mechanism demonstration, not a field benchmark: the system is tested on knowledge that has been entered into it. A correct, fast retrieval is therefore the expected outcome. The comparison with manual documentation is an approximation and not a fair performance benchmark.
- Self-authored test set: the researcher wrote both the stored knowledge points and the test questions, so hit-rates should be read as an upper bound on real performance and are treated as such in Section 6.6.

Why this matters and what remains. Retrieval correctness is the core of the value proposition: a system that stores knowledge but cannot return it in the operator's own words would copy the findability gap identified in Chapter 3. The 88% hit-rate under rephrasing and misspelling shows the semantic mechanism closing that gap for captured knowledge. What remains open is calibration and independence, the test questions share the same author, so the mechanism is demonstrated, while its performance ceiling on real, messy, third-party queries is not yet established.

6.5 Voice Validation

The voice notes were validated on seven recorded notes. Six were short, focused notes in which the speaker describes a single problem. The seventh was a long recording of about eight minutes, included on purpose to find where the capture method breaks down. This one is also the most realistic form of voice audio since it is recorded by Willy during a real workshop conversation. For every voice note, the transcription was compared to the human validated text using the same concept as for OCR: Word Error Rate (WER). The word errors are split into three types: substitutions (a word read or heard wrong), insertions (a word that the model added which was not spoken) and deletions (a word that was spoken but left out). Table 13 reports the results per note. Notes 6 and 15 predate the engine switch and were transcribed by Gemini, note 12 fell back to Whisper after a cloud failure.

Id	Machine	Engine	Ref. words	Transcription time	WER	Ins/del/sub
6	I200	Gemini	54	2,2	3.7%	1/0/1
12	18MS	Whisper	287	241,6	2.4%	0/1/6
15	Overige	Gemini	204	20,8	2.0%	0/0/4
19	Buiten	OpenAI	200	5,0	3.5%	0/0/7
20	Buiten	OpenAI	196	9,3	3.1%	0/0/6
23	Overige	OpenAI	55	3,0	0.0%	0/0/0
21	Overige	OpenAI	1045	27,0	44.1%	429/0/32

Table 13: WER results voice transcription

On the six short notes, the transcription is strong. The WER lies between 0.0% and 3.7% with a median of 2.75%. More important than the rate is the type of error. Almost every error is a substitution, and the substitutions are domain words and proper names that sound alike, such as Camware read as Camber, AJV as AIV, Seiki as Seki, and spindel as spin nul. There are almost no insertions and almost no deletions, which means the engine did not invent content and did not silently drop content. It misheard specific technical words, which is the error type a maintained word list and human validation are meant to capture. And so it does: voice note ID 23 contained all technical jargon that is present in the jargon list, it passed the transcription test fully without any different interpretations of the technical terms. In addition, the engine separated English and Dutch words correctly, which is common during workshop conversations. It also handled regional Twents accent well, there was no noticeable difference between the amount of mistakes on normal Dutch accent compared to Willy's Twents accent.

6.5.1 Long Recording

The long recording, note ID 21, is the boundary case. It was not a single troubleshooting explanation but a live conversation between Willy and several colleagues that moved between topics at once: a machine program, a repair, the day's planning, who should finish a part and at one point a television show. It was transcribed by OpenAI gpt-4o-transcribe. Its WER was 44.1%, far above the short notes. To see where the errors sit, the alignment between raw and validated text was divided into four equal quarters from start to end. Table 14 shows the results.

Quarter	Matches	Substitutions	Insertions	Deletions
1	360	6	3	0
2	358	4	6	0
3	269	18	82	0
4	26	4	338	0

Table 14: results of long recording

Two problems appear. The first is technical and it can be located in the numbers. Out of 461 word errors, 429 are inserted words that the model produced on its own. When the errors are spread over the recording, 338 of those insertions fall in the final quarter. The cause is visible in the raw output: a phrase of about nine words is repeated roughly 15 times in a row. This is a repetition loop, and it is a documented failure mode of a transcription model instead of a property of this particular recording. Radford et al. (2022) describe how long-form audio is transcribed in shifting thirty-second windows, where an inaccurate transcription in one window can carry over and degrade the windows that follow, and they report that repetition looping occurs and that the first or last words of a window can be dropped. So the conclusion here is that the loop was not caused by unclear speech or poor microphone, but by feeding the model a recording long enough to trigger a known weakness of long-form decoding.

The second problem is more fundamental and not caused by the transcriber. During validation it turned out that the first few sentences were not transcribed. This does not show up in the WER and the reason is built into how WER is measured here. The reference text is the human-validated transcription. This reference misses the dropped sentences as well. This effect is once again known to be one of the limitations of long-form recording (Radford et al., 2022). Beyond the transcription itself, even the parts that were transcribed correctly cannot be turned into a usable knowledge point because the recording contains no single problem-and-solution pair. The structuring step has nothing clean to extract. Context is interwoven and partly meaningless to anyone who was not present.

6.5.2 What the boundary case shows

Note ID 21 marks the boundary of the current capture method. It confirms two design decisions made earlier. First, voice capture is intended to be for short audio files, in which the speaker clearly defines a problem, solution, cause and other separate fields of structuring. It is not designed for recording a long working conversation and the measured results match the intended use: the six short voice notes stayed between 0.0% and 3.7% and only the unfocused long note breaks down. The failure is therefore a misuse of the method, not evidence against it. In addition, the boundary case shows why the validation is a must-have and not an optional safeguard. Had this transcription entered the knowledge base unchecked, the repeated loop would have added noise instead of knowledge.

Two limits apply to this conclusion. The audio was recorded with an external microphone in the loud factory hall, and the result was clean and easy to follow, so poor microphone quality is not the cause of the long-recording failure. And this was a single long recording tested on a single engine. The exact rate across long audio in general is not established by one case.

Why this matters and what remains. Voice is the highest-value capture channel because speech is the lowest-effort way for Willy to externalize knowledge. So its reliability on short notes directly supports the low-burden for Willy and multi-modal input requirements. The near-absence of insertions and deletions on short notes means the residual errors are which the jargon list and human gate are built to catch. What remains is that the failure modes are shown on a single case and a single AI engine, so they are demonstrated rather than quantified as a generalized rate.

6.6 Threats To Validation

The limitations mentioned throughout this chapter share a number of common roots. They are consolidated here as threats that bound how far the results can be read. Each with the direction in which it could distort the findings and how it was mitigated where possible.

Small dataset. Retrieval was tested on 25 knowledge points and 21 questions, capture on 19 OCR entries and 7 voice notes. This is enough to demonstrate that each mechanism works, but too small to calibrate thresholds reliably or to conclude results as generalized rates. The cosine-distance threshold in particular is a cold start value based on a near empty knowledge base and will need retuning once real data volume exists. The direction of the distortion: small n makes individual results noisy and can flatter the hit-rate. Mitigation through predefined cases, reporting individual results rather than the aggregate.

Single SME, single case study. All data comes from one company, one expert and a handful of operators. The system is shaped tightly around Huima's constraints, which is appropriate for this project but limits how far the numbers can translate. Direction of distortion: results reflect Huima's specific data and language, this may not hold in another SME. Mitigation by describing design, context and constraints in enough detail for replication.

Limited and self-populated knowledge base. The knowledge base was populated by the researcher, and the same researcher authored the test questions. This is the most important threat because it makes correct retrieval a partly expected outcome. The questions were written, knowing what was stored in the knowledge base. Hit rates should be read as an upper-bound on real-world performance. Mitigation: questions were rephrased into operator language, misspellings and synonyms were included, and abstention was tested on absent topics.

Single-researcher design. Design and evaluation were carried out by the same person, and inherent bias in a bachelor project. Mitigated through standardized cases, predefined criteria, member-checking of captured knowledge with Willy and code-level auditability of the data-boundary.

6.7 Overall Validation Conclusion

Taken together, the tests provide consistent evidence that the mechanism works as designed. Capture succeeds end-to-end, photographed notes and short voice notes become structured, findable knowledge points, with residual errors on which the jargon list and validation gate are built to catch. Retrieval succeeds on captured knowledge: 14 out of 16 answerable cases returned the correct knowledge point first, misspellings and rephrasings were absorbed, the machine filter worked, and the two misses have addressable causes. Reliable answering succeeds: the system abstained correctly on all four absent-answer cases and fabricated nothing, which is the safeguard the whole design is organized around.

6.7.1 Generalizability

Because this is a single-SME case study, the numerical results do not transfer to other companies and are not intended to. What is transferable is the design pattern rather than the measurements. The combination of low-effort, multi-modal capture, human validation gate, local semantic retrieval and abstention-by-default addresses conditions that may be common across small manufacturers: knowledge

concentration in one or multiple ageing experts, undocumented tacit troubleshooting knowledge, low technical maturity and a requirement to keep data in-house.

The limits of this transferability are the Dutch tuned accent, a specific set of machines and error codes, and hardware with a particular boundary on GPU. A company with different fault taxonomy or stricter latency needs would have to re-evaluate the embedding model, the thresholds and the local AI-model choice. The transferable contribution is therefore the validated architecture together with the design reasoning behind it.

6.7.2 Requirements Against Validation

Table 15 separates the requirements by priority and states, for each, how it was tested, what the result was and which limitation remains. “not validated” means no separate test was run, not that the feature is missing. The validation covers every must requirement at scope level and leaves lower priority requirements as future work.

The cost estimate behind the cost requirement is as follows: the local components (PostgreSQL, Ollama, the embedding and generation models) can run on existing hardware and carry no license fees. The only recurring costs are the cloud calls on operator input: at roughly €0.005 per transcribed voice minute, a volume of 100 minutes per month would cost an estimated €0.50, well below the €600 per month from the commercial benchmark in Section 5.1.2.

Requirement	Priority	Test	Result	Limitation remaining
Local data boundary	Must	Chapter 5.4 architecture; auditable in code	Met by design. Auditable in code	Architectural, not measured
Dutch language	Must	All cases in Dutch, Chapter 6.3	Validated	Twents accent tested on limited voice data
Multi-modal input	Must	Chapter 6.3, 6.4 OCR, text and voice	Validated	Small n per channel. Image channel is source dependent
Usable by non-technical operators	Must	Interface built; informal operator feedback only	Partially validated	No structured operator usability test
Validation gate	Must	Chapter 6.3	Validated	Depends on a human correcting the flagged text: adoption assumption
Low burden on Willy	Must	Voice capture designed for it;	Partially validated	Burden not measured in field use
Retrieval faster than current process	Must	Retrieval times (hit@1) in Chapter 6.4;	Partially validated	No fair head-to-head timing vs real documentation search
Cost below benchmark	Must	Estimate based on local stack + cloud calls	Met by estimate	Actual monthly cost at real volume not yet measured.
Structured organization, searchable by others	Should	Chapter 6.4	Validated	Structuring quality bounds retrieval quality; validation of right structuring
Domain specific glossary	Should	Both in OCR and voice notes in Chapter 6	Validated	Glossary partial, not extensively tested.

Search history/recently viewed items	Should	--	Not validated	Built: no separate test
Mobile-friendly interface	Should	--	Not validated	Built: not tested among operators
Image analysis of photos (OCR)	Should	Chapter 6.3, 19 OCR entries	Validated	Accuracy is source-dependent, gate required

Table 15: summary of requirements against validation

7. Conclusion

This chapter returns to the central research question and summarizes what the research has established. The central question was: how can a structured knowledge capture and retrieval system be developed to reduce expert dependency and improve knowledge accessibility at Huima Drive Solutions? The answer is built up from the five sub-questions, each tied to a phase of the research.

The first sub-question asked what critical knowledge exists at Huima and how it is created, stored and retrieved. The current-state analysis in Chapter 3 showed that a large part of the troubleshooting knowledge is tacit and held by one senior expert, Willy, while the documented part is spread across handwritten notes, binders and manufacturer manuals with no unified structure or version control. Operators either consult Willy directly, which is fast and reliable, or search the documentation, which is slow and often does not lead to a usable answer. This confirmed the dependency that motivates the whole project.

The second sub-question asked about the main bottlenecks and the requirements a solution must satisfy. The bottlenecks are that knowledge is not captured in a structured way, that existing documentation is hard to search and incomplete, and that retrieval depends on one person. These were translated into a requirements specification using MoSCoW prioritization (Section 5.1), which defined what the system must do, including low maintenance costs, an interface usable on the work floor, human control over what enters the knowledge base, and the hard requirements that proprietary knowledge stays inside Huima.

The third sub-question asked which approaches exists in the literature for capturing, structuring and retrieving tacit knowledge in manufacturing. The literature review in Chapter 4 indicated that existing approaches often focus on separate parts of the problem, while no reviewed approach was directly suitable for Huima's combination of low-effort capture, local data control, limited technical infrastructure and retrieval on the production floor. This gap justified building an integrated, low-barrier system instead of adopting an existing one, and pointed toward speech as a low-friction capture method and retrieval-augmented generation as a retrieval approach.

The fourth sub-question asked which solution concept best fits Huima's context, requirement and constraints. Chapter 5 compared the alternatives against the requirements and selected a RAG system running locally, with cloud services used only for transcribing and reading operator input. A knowledge-graph approach was considered but rejected, mainly because it requires modelling and maintenance skills that Huima does not have. The selected design keeps proprietary knowledge local, keeps human in control of what becomes searchable and stays advisory instead of replacing operator judgement.

The fifth sub-question asked to what extent the developed system captures and retrieves knowledge correctly and reliably on the representative troubleshooting cases. Chapter 6 validated this on representative cases. Capture works: voice transcription of short, focused notes reached a median word error rate of 2.75%, with the errors almost entirely substitutions of jargon and proper names rather than invented or dropped content. Retrieval also works: out of sixteen answerable cases, fourteen returned the correct knowledge point first and the system fabricated nothing on any of the five absent-answer cases, abstaining on four and asking a clarifying question on one.

Taken together, the five sub-questions answer the central research question. A structured capture and retrieval system can be developed for Huima by combining low-friction speech capture, a human validation gate and local retrieval augmented generation. The validation shows that this system captures and retrieves correctly on representative cases. What the research demonstrates is a feasible path towards reduced expert dependency, not an achieved reduction. The step from a working mechanism to a

measurable drop in how often willy is consulted depends on populating the knowledge base, which is an implementation task instead of a research result.

The contribution of this thesis is therefore threefold. First, it provides an empirical diagnosis of expert dependency in a small manufacturing company, showing how recurring troubleshooting knowledge remains difficult to access when it is held mainly by one senior expert. Second, it translated this problem into a requirements-driven design for low-effort knowledge capture, human validation and local retrieval-augmented generation. Third, it demonstrated the feasibility of this design through a working prototype validated on representative troubleshooting cases. The contribution is not a measured reduction in expert dependency during daily production, but a validated architecture and implementation path through which such a reduction can be pursued.

7.1 Limitations

This research acknowledges several limitations. First, the single case study design means findings cannot be directly generalized to other organizations without further validation (Yin & Campbell, 2018). Second, the number of employees available for interviews is small, which limits the diversity of perspectives and increases the influence of individual views (Saunders et al., 2023). Third, the ten-week timeframe limits the depth of both knowledge elicitation and prototype validation: a longer study would allow more extensive testing under real production conditions. Fourth, document analysis was included as data collection method but is restricted in practice by Huima's data security policies. Fifth, because this researcher both designs and evaluates the solution, there is an inherent risk of researcher bias (Saunders et al., 2023) in the validation phase, mitigated through standardized test cases and pre-defined evaluation criteria.

7.2 Implementation Advice

This chapter delivers the sixth and final deliverable: practical recommendations for how the system can be maintained, updated and expanded after this project ends. Section 7.2.1 covers the most important condition for the system: populating the knowledge base. Section 7.2.2 covers the system running and trustworthy over time (maintenance and governance). Section 7.2.3 covers the technicalities that were left out of the prototype scope. Finally, Section 7.2.4 looks further ahead at capture methods and model choices that did not fit Huima's context yet but may become realistic later.

7.2.1 Populating the Knowledge Base

The single most important recommendation is: the value of the system depends entirely on the knowledge that is put into it. As established in Section 5.1.5, populating the knowledge base was placed outside of the project scope, and the validation in Chapter 6 is set up to confirm the mechanism on representative cases rather than on a filled base. The system as delivered is therefore a working but largely empty container. Filling it is an implementation task, and how it is approached will decide whether the system becomes an alternative to consulting Willy or remains a demonstration.

The recommended approach is to start with the highest-value, highest-frequency knowledge first, instead of attempting to capture everything. Chapter 3 already identified this: the recurring troubleshooting problems on the 18MS which generates the highest volume of questions, and Willy's own estimate that around 80% of questions are recurring. Capturing that recurring core means the system can start answering the most common questions early, which is what build operator trust and adoption. A practical way to drive this is to log which questions are actually asked to Willy, and to capture those first.

7.2.2 Maintenance and Governance

For the system to stay trustworthy, two things need an owner: the knowledge base and the system itself. The validation gate ensures that AI-transcribed and AI-structured entries are checked by a human before they enter the knowledge base, but it does not keep existing entries correct over time. As machines, parts and procedures change, entries can become outdated. The recommendation is to designate a knowledge owner, practically most logically Willy while he is still present, who can approve new entries and review older existing ones.

A second governance point concerns the boundary between captured knowledge and human judgement. As argued in Section 5.1 and based on Hoerner et al. (2023), the system is advisory: it supports the operator but does not replace the operator's judgement at the point of use. This should be made clear when the system is rolled out, so that operators understand that a returned answer is a strong suggestion based on past cases, not an instruction to follow blindly. Keeping this framing matters most for safety-relevant problems, where an unverified answer could otherwise be over-trusted.

Finally, the data boundary that the CEO of Huima confirmed as an organizational decision should be treated as a rule, not a one-off setting. Stored proprietary knowledge stays local, and only transcription and OCR may use cloud service. Any future change to the system, such as adopting a new language model, should be checked against this boundary rule.

7.2.3 Technical

Several technical simplifications were accepted on purpose for the prototype and are documented in Section 5.4.8 and the appendices. Before the current system is relied upon in everyday production, the following should be addressed, in roughly this order of importance.

- Authentication and access. The current login is lightweight: a name and PIN. With credentials not stored in hashed form and sessions that do not survive server restart. This is only acceptable because the server is reachable over the private network only. Before any wider extension, credentials should be hashed and sessions made persistent.
- External dependencies. The backend relies on third-party npm packages, which could contain vulnerable or malicious code (a known supply-chain risk). These were not formally audited within this project, before wider deployment the dependencies should be scanned and pinned.
- Database integrity and cleanup. Appendix B lists a set of improvements that were deferred: a nullable foreign key on voice notes, a separate error-code table, removal of the columns from earlier iterations, and moving media to external file storage.
- Retrieval scalability. The prototype searches vectors with a sequential scan, which is sufficient at the current state. Once the knowledge base grows, a vector index should be added so that retrieval stays fast. Provisional distance threshold (Section 5.4.5) was tuned on a nearly empty base and should be re-evaluated once real volume exists.
- Usability testing with operators. Chapter 6 validates the mechanism but limited operator usability on the floor. The operator-usability must-have is currently supported by design rather than measured. A short, structured test once the knowledge base is populated would close this gap and is recommended before full roll-out.

7.2.4 Future Capture Methods and Model Choices

This section looks at directions that the literature points to but that did not fit Huima's context, budget or technical maturity within this project's period. They are presented as options to revisit, not as immediate steps, and they connect back to the forward-looking note already made in Section 5.1.5.

The MOA framework (Gavrilova & Andreeva, 2012) frames the goal clearly: the less effort capture requires of the expert, the more knowledge actually gets externalized. The prototype already reduced this effort with speech, the next step in the same direction would be capture methods that require even less active work from Willy. The literature describes sensor- and recording-based capture such as augmented-reality head-mounted devices (Sarhan et al., 2022), automatic generation of instructions from process recordings (Burggraef et al., 2024) and motion capture with video analysis (Yoshida et al., 2011). For example, a wearable microphone or smart glasses could let Willy narrate hands-free while actually working on a machine, capture knowledge in the moment rather than in a separate session. The reason these were not built is consistent across the reviewed literature: they are either still conceptual or limited to procedural knowledge, and they require infrastructure and integration effort that Huima does not yet have. They become realistic once the basic system is established and the organization has appetite for further investment.

Stronger or alternative speech models. Transcription quality on difficult audio (Twents accent and jargon) is central to capture quality, so it is worth revisiting it periodically. The current primary/fallback design (Section 5.4.4, Appendix F) already makes the engine a replaceable component. As speech models improve, alternative engines should be tested side by side on real Huima voice notes, and the best-performing one adopted. Keeping more than one as fallback remains sensible given the hardware constraint. The same applies to the local generation and embedding models: as more capable models that fit within the GPU budget become available, they can be swapped in, bearing in mind that changing the embedding model forces embeddings to be regenerated.

Connecting the knowledge. RAG was selected over a knowledge-graph approach in Section 5.3, primarily because a knowledge graph requires modelling and maintenance skills that Huima does not have. That reasoning still holds for now. However, if the knowledge base grows substantially and the relationships between problems, machines, causes and solutions become valuable to navigate explicitly, a hybrid that adds a graph layer on top of the existing retrieval layer could be reconsidered, ideally only if external support is available to build and maintain it. It is listed here as long-term option, rather than recommendation.

Appendix A: Empirical Data Collection

Two initial interviews were conducted with two key operators to gauge the sentiment around the troubleshooting situation at the time. The interviews were held one on one in a separate room and without anyone in a higher hierarchical position present, so that operators could speak freely without any influence of management. This setup follows the recommendation that experts are best interviewed individually and away from situational pressure. The operators were aware of the researcher's position within the company.

The interviews surfaced several points. The first was the lack of searchable data, which was confirmed independently later in the research. The second was the dependency on Willy: both operators indicated that if Willy were to leave, the company would face serious problems with troubleshooting, processes and machine knowledge. Towards the end, one operator demonstrated how the current troubleshooting process roughly works. It starts with a fault on the machine, sometimes signaled by an indication and sometimes not. For the machine this operator worked on, one of the first steps is to search the logbooks kept at a central, openly accessible location on the work floor. The operators stressed that the notes in these logbooks are old, scattered and often hard to read or understand. A further point raised was that a fire would destroy this knowledge permanently, which indirectly underlines the absence of a digital knowledge system.

These implicit, handwritten notes are not digitized, whereas manuals, settings and drawings are already kept in a digital environment. The digital documentation is not the core issue Huima faces and is therefore treated as out of scope.

The two interviews served mainly to establish the urgency and direction of the problem instead of collecting exhaustive data. After them, further input was gathered informally during site visits and no additional structured interviews were held.

A.1 Document Analysis

Document analysis is used as complementary data collection method alongside observations and interviews. Following Bowen (2009), document analysis is a systematic procedure for reviewing existing material to gain insight and corroborate findings from other sources. In this research it was applied to the documentation already present on the work floor, most importantly the 18MS machine logbook, the digital error-code lists and manuals. The document analysis served as test for the prototype.

A.2 Observation Examples

During a period of approximately 10 weeks, the researcher was able to experience firsthand troubleshooting occurrences while being on the work floor. These moments are the perfect moment to gather data about the troubleshooting process, with Willy as well as with the operators. A couple examples are shown here together with the structural observations.

A.2.1 Troubleshooting During a Machine Error

During an observed machine error, the operator first restarted the machine, which appeared to be a common initial response to a technical problem. When the error remained unresolved, the operator asked Willy for help. Willy advised checking the logbook and error codes, after which the issue was initially interpreted as air in the grease tube. The tube was bled and the sensor temporarily indicated correct pressure, but the problem returned after restarting the machine. Willy then continued the diagnosis by physically inspecting the tube and identified that the head of the tube had broken off. As a result, the machine was drawing in both grease and air, meaning that the earlier assumption did not address the real cause. Willy also noticed minor play in the tube and corrected it immediately.

This case shows multiple thing, first of all, the initial assumption is not always the final solution, it also shows that this operator escalated to Willy too quickly, yet Willy also owned the problem too quickly. This is another point of attention during the tacit knowledge capturing process. Willy is programmed to solve things himself, he likes it really much he says as well. This is something that obstructs the learning process of the operator operating the specific machine. So, in order to accelerate the process of tacit knowledge capture, Willy should be supporting the operator instead of doing it himself: this is a process that he is currently undertaking, Huima hired an external consultant for this that leads willy in the direction of a coach/teacher rather than a maintenance man.

A.2.2 Troubleshooting Error Code 220

During an observed troubleshooting situation on machine 18MS, an experienced operator encountered fault code 220, which appeared to be related to a sensor. Although the operator had worked at Huima for many years, he did not know the exact solution and first consulted the available logbook. However, fault code 220 was not clearly listed. As a result, the operator searched through the entire logbook rather than being able to use a clear category or search path.

The operator found two potentially relevant fault sheets: one for code 221 and one that appeared to refer to 220, but the available information did not provide a clear or directly applicable solution. He also indicated that the online fault-code logbook exists, but was mainly written in English and lacked important information, as Willy also acknowledges. Searching the available documentation took at least ten minutes, after which the outcome was still uncertain.

Willy was absent during this situation (holiday), so direct expert consultation was not possible. The problem remained unresolved on the same day and was only solved the following week. This observation illustrated that the current documentation is difficult to search, incomplete and not always sufficient for independent troubleshooting. It also raises the question of whether the problem would have been resolved faster if Willy had been available, which further indicates the dependency on tacit expert knowledge.

A.3 Willy's Absence Period

The two-week period in which Willy was absent could have functioned as a useful measurement period for assessing the practical consequences of expert unavailability. During such a period, it would have been possible to observe how often operators encountered unresolved technical problems, how long they spent searching for solutions, which documentation sources they used, and whether problems remained unsolved for longer than usual. However, this opportunity could only have been used systematically if the absence had been known in advance and if a simple observation or logging method had already been prepared. Since this was not the case, the observations from this period can still be used as indicative evidence, but not as controlled or complete measurement.

A.4 Questionnaire Findings

As part of the empirical data collection, a short questionnaire was used to gather information about how employees at Huima currently deal with technical problems, machine settings, uncertainties, and the available documentation. The questionnaire focused on three main themes: employee independence in problem-solving, dependency on Willy as the main knowledge holder, and perceived limitations of the current documentation system. The questionnaire consisted of both closed and open-ended questions.

A.4.1 Questionnaire Items

1. How long have you worked at Huima?
2. What do you do when you encounter a problem that you cannot solve directly yourself?

3. Of the technical problems or uncertainties you encounter, what proportion can you roughly solve yourself without Willy's help?
4. How often do you ask Willy for help with a technical problem, machine setting, or uncertainty?
5. If you cannot solve a technical problem immediately, approximately how long do you spend looking for a solution before asking Willy for help?
6. Do you use the available documentation, such as manuals, notes, or binders?
7. If you use the documentation, do you usually find the solution?
8. What is usually the problem with the current documentation?
9. What happens if Willy is not available and you do not know how to solve the problem yourself?
10. Approximately how much longer does it take to solve a problem when Willy is not available?
11. What would help you to solve problems yourself faster?

A.4.2 Open-Ended Responses

The final question asked respondents what would help them solve problems independently and more quickly. The response are shown below.

Table A.1: Responses Open question

RESPONDENT	RESPONSE
R1	I don't know
R2	Problems in a step-by-step guide and a digital structured system
R3	Pictures with instructions
R4	Better search functions for the logbooks and maybe digital
R5	Logbooks up-to-date
R6	Step-by-step guide
R7	All of the examples of the previous question and an easier way to save your own information digitally which can also be found back easier by others

The open-ended responses suggest that employees mainly need clearer, more accessible and better-structured knowledge support. Several respondents referred to step-by-step guidance, indicating a need for procedural knowledge rather than only general documentation. Other responses point to the importance of visual support, such as pictures with instructions, and improved digital searchability of existing logbooks. The responses also indicated that keeping knowledge up to date and making newly added information easier to retrieve are important requirements for a future knowledge management solution.

A.4.3: Questionnaire Findings (Dutch)

R	Dienstjaren (Q1)	Eerste actie (Q2)	Zelf op te lossen (Q3)	Frequentie Willy (Q4)	Zoektijd vóór Willy (Q5)	Documentatie-gebruik (Q6)	Vindt oplossing (Q7)	Probleem met documentatie (Q8)	Vertraging zonder Willy (Q10)
1	5–10 jaar	Collega vragen	0–25%	Meer dan 1×/week	< 5 min	Zelden (gebruikt niet)	–	Onduidelijk waar te zoeken	15–30 min langer
2	1–5 jaar	Documentatie/notities zoeken	25–50%	± 1×/week	15–30 min	Vaak	Vaak	Moeilijk juiste oplossing te vinden	> 30 min langer
3	5–10 jaar	Collega vragen	Weet niet	Geen vast moment	15–30 min	Vaak	Soms	Onduidelijk waar te zoeken; geen duidelijke uitleg; moeilijk juiste oplossing te vinden	Verschillende tijden
4	> 10 jaar	Documentatie/notities zoeken	75–100%	± 1×/maand	Vraagt Willy meestal niet	Vaak	Soms	Niet up-to-date; geen duidelijke uitleg	> 30 min langer
5	> 10 jaar	Zelf proberen (zonder docs)	50–75%	± 1×/week	15–30 min	Altijd	Vaak	Onduidelijk waar te zoeken	> 30 min langer
6	> 10 jaar	Zelf proberen (zonder docs)	50–75%	± 1×/maand	15–30 min	Vaak	Vaak	Geen problemen met documentatie	Niet langer
7	< 1 jaar	Zelf proberen (zonder docs)	Weet niet	± 1×/maand	15–30 min	Zelden	Soms	Moeilijk juiste oplossing te vinden; onduidelijk waar te zoeken; soms slecht leesbaar; geen duidelijke uitleg	Afhankelijk van collega’s

R	Dienstjaren (Q1)	Eerste actie (Q2)	Zelf op te lossen (Q3)	Frequentie Willy (Q4)	Zoektijd vóór Willy (Q5)	Documentatie-gebruik (Q6)	Vindt oplossing (Q7)	Probleem met documentatie (Q8)	Vertraging zonder Willy (Q10)
8	1–5 jaar	Zelf proberen (zonder docs)	75–100%	± 1×/week	Vraagt Willy meestal niet	Altijd	Vaak	Geen duidelijke uitleg; moeilijk juiste oplossing; niet op één plek; onduidelijk waar te zoeken; niet up-to-date	15–30 min langer

A.4.4: Questionnaire Findings (English)

R	Years of service(Q1)	First action(Q2)	Can solve alone(Q3)	Frequency Willy(Q4)	Search time before Willy(Q5)	Documentation use(Q6)	Finds solution(Q7)	Problem with documentation(Q8)	Delay without Willy(Q10)
1	5–10 yrs	Ask a colleague	0–25%	More than 1×/week	< 5 min	Rarely (does not use)	–	Unclear where to look	15–30 min longer
2	1–5 yrs	Search documentation/notes	25–50%	± 1×/week	15–30 min	Often	Often	Hard to find the right solution	> 30 min longer
3	5–10 yrs	Ask a colleague	Don’t know	No fixed moment	15–30 min	Often	Sometimes	Unclear where to look; no clear explanation; hard to find the right solution	Varying times
4	> 10 yrs	Search documentation/notes	75–100%	± 1×/month	Usually doesn’t ask Willy	Often	Sometimes	Not up-to-date; no clear explanation	> 30 min longer
5	> 10 yrs	Try to solve alone (no docs)	50–75%	± 1×/week	15–30 min	Always	Often	Unclear where to look	> 30 min longer

R	Years of service(Q1)	First action(Q2)	Can solve alone(Q3)	Frequency Willy(Q4)	Search time before Willy(Q5)	Documentation use(Q6)	Finds solution(Q7)	Problem with documentation(Q8)	Delay without Willy(Q10)
6	> 10 yrs	Try to solve alone (no docs)	50–75%	± 1×/month	15–30 min	Often	Often	No problems with documentation	Not longer
7	< 1 yr	Try to solve alone (no docs)	Don’t know	± 1×/month	15–30 min	Rarely	Sometimes	Hard to find the right solution; unclear where to look; sometimes hard to read; no clear explanation	Dependent on colleagues
8	1–5 yrs	Try to solve alone (no docs)	75–100%	± 1×/week	Usually doesn’t ask Willy	Always	Often	No clear explanation; hard to find right solution; not in one place; unclear where to look; not up-to-date	15–30 min longer

Appendix B: Database Design and Relationships

The prototype uses a single PostgreSQL database with the pgvector extension. The database has two purposes: it stores the knowledge itself in human-readable form, and it stores the embedding vectors that make semantic search possible. Keeping both in one database means there is one system to back up, secure, and understand, which fits Huima's requirement of low maintenance cost (Section 5.1). This appendix documents the table structure behind Section 5.4.3 and the reasoning behind the relationships between the tables.

B.1 Main Tables and Fields

The schema consists of one parent table and two attachment tables. One row in the parent table represents one knowledge point: one troubleshooting incident or one piece of knowledge.

Table B.1: Tables of the database

Table	Role	Most important fields
LogboekKennis	The knowledge point (parent). One row represents one troubleshooting incident or piece of knowledge	id; machine; naam (who); aangemaakt_op_datum (when); originele_notitie (raw text); probleem; vermoedelijke_oorzaak(possible cause); oorzaak (confirmed cause), wat_geprobeerd (what was tried); oplossing (solution); suggesties (suggestions); errorcode; embedding_bge (vector)
foto	Photos attached to a knowledge point (0 or more). A photo cannot exist without its knowledge point.	id; kennispunt_id(FK to parent); foto_base64(the image); mimetype;
spraaknotitie	Voice notes attached to a knowledge point (0 or more), or standing alone as loose memo	id; kennispunt_id(soft, nullable reference to parent); audio_base64; transcriptie (raw transcription); transcriptie_status (wachtrij/bezig/klaar); transcriptie_motor (Openai, Gemini or Whisper); gevalideerde_tekst (human-approved text); gevalideerd_door (validated by); gevalideerd_op (validation date)

The fields play distinct roles. The field originele_notitie is the single accumulation point for all searchable text: typed text, OCR text from photographed notes and, after human validation, transcribed speech all end up here, so it always shows what was originally written or said. This field is not itself embedded. The structured fields (problem up to suggesties) are derived from the raw text by the structuring model and remain editable by humans. The embedding (embedding_bge) is generated from a composite of the six structured content fields, as described in Appendix E.1, which means that semantic search operates on the structured representation of a knowledge point rather than on the raw note directly. The context fields (machine, naam, aangemaakt_op_datum) enable the machine filter at retrieval and the code-rendered source attribution under each generated answer.

A few columns from earlier iterations of the prototype are still present in the table but no longer used: a 768-demensional embedding column from the first embedding model, a single-photo column and a category column. Removing these legacy columns is housekeeping and is listed as future work.

B.2 How the Tables are Linked

One knowledge point is treated as the parent record because one troubleshooting incident is the natural unit of knowledge: an operator who later faces the same problem needs the whole incident – what was observed, what was tried, what worked – in one place, not spread over separate records per input type. Photos and voice notes are therefore attached to the knowledge point through the reference field kennispunt_id.

The two attachment types are linked in deliberately different ways. Photos use a hard, database-enforced foreign key with cascading delete. A foreign key means that the database itself requires the kennispunt_id of a photo to match an existing knowledge point, so a photo cannot exist without its parent, and deleting a knowledge point automatically removes its photos. Voice notes use a soft, nullable reference that is maintained by the application instead of by the database: the kennispunt_id of a voice note may also be empty, so that a voice note can stand alone as a loose memo. This was deliberate relaxation, because standalone voice memos were an existing behavior of the application that had to keep working. The consequence is that the database does not guarantee referential integrity for voice notes: deleting a knowledge point leaves its voice notes behind as orphaned records. The two mechanisms next to each other show that the soft link is a choice rather than an oversight: the strict option was available and is used for photos. Adding a nullable foreign key for voice notes is listed as future work.

B.3 Status Model for Voice Notes

The field transcriptie_status implements a small state machine for the voice-note pipeline: wachtrij (queued) → bezig (being transcribed) → klaar (transcribed, waiting for validation) → gevalideerd (approved by human). A failed transcription receives the status “mislukt”, with the reason recorded and can be retried. Only the transition to “gevalideerd” makes spoken content eligible to be appended to the raw text of the parent knowledge point and included in its embedding. The status model is therefore the database-level expression of the design rule from Section 5.4.4: speech is stored first and only becomes searchable after human validation.

B.4 Lifecycle of a Combined Report

A combined report illustrates how the tables work together. Suppose an operator creates a report on machine 18MS with a typed text block, one voice note and one situation photo. At the moment of saving, three things are written: one LogboekKennis row (with the typed text in originele_notitie), one foto row pointing to it, and one spraaknotitie row pointing to it with status “wachtrij”. In the background, the structuring model fills the structured fields from the typed text and the embedding is generated from those fields. The voice note is transcribed in the background and waits for validation. After an allow-listed user validates the transcript, the approved text is appended to originele_notitie of the parent, and the knowledge point is re-structured and re-embedded. The knowledge point therefore grows with each validated spoken addition, while the audio and photos remain stored as separate rows – nothing is lost while a transcription is not yet validated. A situation photo never contributes text; it is stored as visual evidence only.

B.5 Media Storage

Photos and audio are stored as base64 text inside the database. Base64 is a way of representing binary files, such as image and audio, as plain text, so that they can be stored in an ordinary database column. For a prototype this is a defensible simplification: a knowledge point and its attachments are saved or rolled back together in one transaction, a backup is a single database dump, and there are no separate files on disk that can get out of sync with the database. The trade-off is a larger database and heavier

data transfers. Two mitigations limit this in practice: photos are compressed on the operator's device before upload (maximum 1600 pixels, JPEG quality 0.8), and audio is excluded from list requests and only fetched per item when it is actually played. Moving media to external file or object storage is the more scalable design and is listed as future work.

B.6 Normalization Choices and Future Work

Two deliberate deviations from a fully normalized schema remain. First, the errorcode field is free text and may contain several codes in one cell, this is handled at query time with word-boundary matching, as described in Appendix E.5, instead of migrating the existing data to a separate code table. Second, the structured fields duplicate content that is also present in the raw field; this duplication is intentional, because the raw field serves as the audit record while the structured fields serve retrieval and display.

A more normalized schema is identified as future work rather than built into the prototype: a nullable foreign key on spraaknotitie.kennispunt_id (so that standalone memos remain possible while orphaning is prevented), a separate error-code table with one code per row, externally stored media instead of base64 columns, removal of the legacy columns, and a vector index[22] (for example HNSW) once the knowledge base grows large enough that sequential scanning becomes too slow. These changes would improve integrity and scalability, but they were not necessary to demonstrate the system on representative cases, and introducing them now would have required migrating existing data during the thesis period.

[22] A database structure that speeds up semantic search by avoiding a full comparison with every stored embedding, HNSW is a common method for this (Neon, n.d.)

Appendix C: Architecture and API Overview

This appendix supports Section 5.4.2 by describing the main components of the prototype and the role of the API routes. It does not include the full source code, but gives enough technical detail to show how the frontend, backend, database and AI services work together.

Node.js with Express was used for the backend because it provides a lightweight and flexible way to build API routes between the frontend, database and AI services. Since the frontend was also written in JavaScript, using Node.js reduced implementation complexity and allowed the prototype to be developed quickly. Express provided the route structure for handling actions such as saving knowledge points, uploading voice notes, searching and generating answers.

Table C.1: Components of the prototype.

Component	*Technology*	*Role*
Operator app	Progressive web app (browser or phone)	Capturing knowledge (text, photo, voice), searching, asking questions and validating transcriptions
Operator API	Node.js / Express, local server	All server-side logic; the trust boundary of the system; the only component that holds secrets.
Management webapp	Node.js / Express, local server	Desktop read-and-edit view of the knowledge base for office use.
Database	PostgreSQL with pgvector, local	Stores the knowledge, the media and the embedding vectors
Local model runtime	Ollama	Hosts the embedding model (bge-m3) and the generation model (Qwen 2.5 7B); fully local
Transcription fallback	Faster-whisper (large-v3), local subprocess	Local speech-to-text when the cloud path fails.
Cloud AI	OpenAI	Primary speech-to-text of operator input only.
Cloud AI	Gemini	Structuring and OCR
Network	Tailscale private network with HTTPS	Only enrolled devices can reach the server; HTTPS is also required by browsers for microphone access.

C.1 API Overview

All routes except logging in and the static interface files require a token that the operator receives after logging in with name and PIN. The table below describes the routes at a functional level, together with the AI step they use and whether that step runs locally or in the cloud. The table makes the data boundary from Section 5.4.4 visible: the only routes that involve a cloud call operate on input that an operator has just created, while search and question answering are fully local.

Table C.2: Functional overview of the operator API routes.

Route(s)	**Purpose**	**AI processing used**	**Cloud or local**
/login, /logout, /mij	Signing in and out with name and PIN; identifying the current user.	-	-
/kennispunten (list, create, read, update, delete)	Managing knowledge points. The embedding is generated at write	Embedding	Local

	time so a new entry is immediately findable		
/kennispunten/:id/fotos, /fotos/:id	Attaching photos to a knowledge point and removing them	-	-
/spraaknotities(create, list, retry, delete)	Storing voice notes and queueing them for transcription.	Speech-to-text	OpenAI, Gemini, Whisper as fallback
/spraaknotities/:id/whisper	Optional local comparison transcription of an already transcribed note.	Speech-to-text	Local
/spraaknotities/:id/valideer	Human validation of a transcription. Merges the approved text into the parent knowledge point and triggers re-structuring and re-embedding.	Structuring, embedding	OpenAI, Gemini
/transcribeer	Voice search: transcribes a spoken query without storing anything.	Speech-to-text	OpenAI, Gemini, Whisper as fallback
/structureer	Structures free text into the fixed knowledge fields	Structuring	Gemini
/lees-notitie, /verwerk-notitie	OCR of photographed notes, and OCR combined with structuring in one step.	OCR (and structuring)	Gemini
/zoek	Semantic search and error-code search	Query embedding	Local
/ask	Full question answering (retrieval plus answer generation)	Embedding, generation	local

The management webapp exposes read and edit pages on the same database, and forwards questions to the operator API with a server-side service account, so that no credential ever reaches a browser. The management webapp has no end-user login of its own and relies on the private network for access control; this is acknowledged as a limitation in Section 5.4.8.

C.2 Trust Boundary and Secrets

The browser communicates only with the application's own API. It never connects directly to the database or to a cloud service. The database credentials and the OpenAI and Gemini API keys are held in an environment file, excluded from version control. The interface files are served through an explicit allow-list of filenames: only the files on the list are sent to the browser, and any other file in the directory is refused, so a configuration file can never be served by accident. Cloud processing is limited to three tasks: transcription of voice notes, optical character recognition (OCR) and structuring of free text into knowledge fields. Each of these goes through a small set of server-side functions that add the API key on the server, so the key never reaches the browser. The two providers are used differently: for transcription the system tries OpenAI first, then falls back to Gemini and finally a local Whisper model, while OCR and structuring run on Gemini only. Everything else stays local, the embeddings used for search and the language model that answers operator questions both run on the server through Ollama

and never leave it. Because the cloud boundary is just this small named set of call sites in the code, the governance claim from Section 5.4.4 can be checked directly in the source instead of being taken on trust.

C.3 Error Handling and Robustness

Errors are handled in a way that prevents the prototype from failing silently or losing operator input. When an AI service returns a temporary error, such as a rate limit or a short outage, the backend retries the request up to three times before showing a clear Dutch error message to the operator. Errors caused by invalid input fail immediately, because retrying would not solve them. Before heavy processing starts, uploaded files are checked: audio files must have an accepted file type and stay below 10MB, OCR uploads are capped at 15MB, and general request bodies at 20 MB. The local Whisper transcription process has a hard timeout of fifteen minutes, so a stuck transcription cannot block the queue indefinitely, and temporary files are removed afterwards. When the server restarts, voice notes that were still queued or being processed are added back to the queue. The local models are also preloaded at startup, because loading a model from disk was slower than the actual search step. Together, these measures make the prototype more robust: failures are reported, inputs are preserved and the system can recover after interruptions.

Appendix D: Prompt Design and Validation Logic

This appendix documents the design logic of the five prompts used in the prototype, and the rules of the human validation step. The prompts are the place where the design principles from Section 5.4 – literal transcription, no invented information, abstention instead of guessing – are actually enforced, which is why their constraints are described here. The verbatim Dutch prompt texts are inserted from the code at the indicated places.

D.1 OCR Prompt

The OCR prompt enforces three constraints. First, literal transcription only: the model must write exactly what is on the note, invent nothing and complete nothing, and copy technical jargon and error code verbatim. Second, the pre-printed text of the standard logbook form must not be transcribed, only the handwriting counts, so that the form's own text does not pollute the knowledge. Third, the model must mark every reading it is not sure about – unclear handwriting, ambiguous words, crossed-out or overwritten text – by wrapping it in a visible [[ ]] marker. The prompt instructs the model to be strict with itself (when in doubt, mark), but not to mark everything. Multiple photos of one note are sent in a single call as ordered pages, so the model reads them as one document. The markers remain visible in the editable text, where the interface highlights them for the person correcting the transcription, and they are stripped before structuring and storage.

const LEES_NOTITIE_PROMPT =

`Lees de tekst van deze notitie(pagina's) letterlijk over naar het Nederlands. De pagina's horen bij één notitie; voeg ze samen tot één doorlopende tekst in de juiste volgorde. Het kan technisch machine-jargon en afkortingen bevatten (errorcodes, SNV, t.l. e.d.) — neem die letterlijk over. Verzin niets, vul niets aan.

Bovenaan de notitie staat voorgedrukte, getypte standaardtekst (een handleiding in een vast lettertype). Transcribeer die NIET. Transcribeer uitsluitend de handgeschreven notitie. Als je twijfelt of iets gedrukt of geschreven is, neem het mee maar markeer het als onzeker (zie hieronder).

Onzekerheid markeren: zet woorden of stukken waarvan je de lezing niet zeker weet tussen dubbele vierkante haken, bijvoorbeeld [[zo]]. Markeer in elk geval bij:

- onduidelijk, slordig of moeilijk leesbaar handschrift;
- een woord dat je op meerdere manieren kunt lezen (kies de waarschijnlijkste en markeer);
- doorgestreepte, overschreven of verbeterde tekst;
- vlekken, vouwen of slecht zichtbare/afgesneden delen;
- een afkorting, code of vakterm die je niet zeker herkent.

Wees streng op jezelf: bij twijfel markeren is beter dan niet markeren. Markeer echter niet álles — woorden die je duidelijk en zeker kunt lezen blijven zonder haken.

Geef ALLEEN de overgeschreven tekst terug (platte tekst, met [[ ]] rond onzekere stukken), geen uitleg.`;

D.2 Structuring Prompt

The structuring prompt converts free text into the seven fixed knowledge fields and an onzeker list for uncertain field assignments. The output is forced into JSON format so the backend can process it reliably. The prompt instructs the model to fill fields only with information that is literally present in the text, missing information must remain empty. It also defines the meaning of each field, including the difference between a suspected and a confirmed cause. The structuring step itself does not store data, saving is handled separately, and the resulting fields remain editable by humans.

const STRUCTUREER_PROMPT =

`Je bent een hulp die werkplaatsnotities van monteurs structureert.
Je krijgt vrije tekst over een machinestoring en zet die om naar JSON.

STRIKTE REGELS:

- Antwoord UITSLUITEND met geldige JSON. Geen markdown, geen uitleg, geen codefences.
- Gebruik exact deze velden: probleem, vermoedelijke_oorzaak, oorzaak, wat_geprobeerd, oplossing, suggesties, errorcode en daarnaast "onzeker".
- Vul een veld ALLEEN met informatie die letterlijk uit de tekst blijkt. Niets verzinnen, niets aanvullen, niet interpreteren.
- Velden die niet in de tekst voorkomen krijgen een lege string "".
- "errorcode" bevat alleen de foutcode(s) als die genoemd worden (bijv. "220"), anders "".
- "onzeker" is een array met de namen van de velden waarover je twijfelt (twijfelachtige of dubbelzinnige toewijzing). Bij geen twijfel: een lege array.
- "probleem" is een KORTE titel: maximaal 8 woorden (richtlijn: onder de 60 tekens). Alleen de kern van de storing, geen volzinnen. Details zoals oorzaak, geprobeerde stappen of de oplossing horen NIET in de titel maar in hun eigen veld.
- Taal blijft Nederlands. Herformuleer niet onnodig; neem de woorden van de monteur zoveel mogelijk over.
Betekenis van de velden:
- probleem: korte omschrijving van wat er mis is (de kern/titel, max 8 woorden).
- vermoedelijke_oorzaak: wat de monteur DENKT dat de oorzaak is (nog niet bevestigd).
- oorzaak: de bevestigde oorzaak, als die is vastgesteld.
- wat_geprobeerd: stappen die al geprobeerd zijn.
- oplossing: wat het probleem uiteindelijk heeft opgelost.
- suggesties: tips voor de volgende keer.

VOORBEELDEN:

Tekst: "Compressor maakte hard geluid. Dacht eerst dat het de lagers waren. Heb hem uitgezet en opnieuw gestart, hielp niet. Bleek uiteindelijk een losgeraakte bevestigingsbout, die heb ik vastgezet en toen was het stil."
JSON: {"probleem":"Compressor maakt hard geluid","vermoedelijke_oorzaak":"lagers","oorzaak":"losgeraakte bevestigingsbout","wat_geprobeerd":"uitgezet en opnieuw gestart","oplossing":"bevestigingsbout vastgezet","suggesties":"","errorcode":"","onzeker":[]}

Tekst: "Alarm 227 kwam op, handmatig wisselen lukte ook niet meer."
JSON: {"probleem":"Alarm 227, handmatig wisselen lukt niet","vermoedelijke_oorzaak":"","oorzaak":"","wat_geprobeerd":"handmatig wisselen","oplossing":"","suggesties":"","errorcode":"227","onzeker":[]}

Tekst: "Turret stond scheef, weet niet goed waardoor."
JSON: {"probleem":"Turret staat

scheef","vermoedelijke_oorzaak":"","oorzaak":"","wat_geprobeerd":"","oplossing":"","suggesties":"","errorcode":"","onzeker":[]}`;

D.3 Transcription Prompt

The transcription prompt asks for a literal Dutch transcription and warns the model that the speaker may have a regional (Twents) accent and use machine jargon. A maintained word list of company-specific terms, such as machine names and parts, gives the model the correct spellings, this is the prompt-side equivalent of adapting a speech model to a domain. The instruction is again to write exactly what is said, invent nothing and complete nothing, and to output only the transcription.

```
const TRANSCRIPTIE_TERMEN =

  'spindel, AJV, turret, 18MS, robotcell (soms "Willem" genoemd), Huima, tools, tooling';

const TRANSCRIPTIE_PROMPT =

  'Transcribeer deze audio letterlijk naar het Nederlands. ' +
  'De spreker kan een Twents/regionaal accent hebben en technische vaktermen of ' +
  'machine-jargon gebruiken (bijvoorbeeld errorcodes en afkortingen). ' +
  'Veelgebruikte termen in dit bedrijf, gebruik deze schrijfwijze als je ze hoort: ' +
  TRANSCRIPTIE_TERMEN + '. ' +
  'Schrijf precies wat er gezegd wordt, verzin niets, vul niets aan. ' +
  'Geef ALLEEN de transcriptie terug, zonder uitleg of opmaak.';
```

D.4 Answer-Generation Prompt

The system prompt for answer generation establishes the role of the model (a technical assistant for operators) and declares the numbered knowledge fragments to be the only permitted source. The answer must present the possibilities from all relevant fragments instead of dictating a single fix, and must explicitly distinguish between what was tried and worked, what was tried but did not help, suggestions and error codes. The model may mention who added something and when, because this information is supplied with each fragment, but it must never attach a name to a fragment it does not belong to. The guaranteed source list under each answer is rendered by the application from database records, as described in Section 5.4.6. Finally, the prompt contains the abstention rule: if the answer is not in the fragments, the model must return the exact fixed sentence: “Dit kan ik niet vinden in de beschikbare kennis” and must not guess.

```
const ASK_SYSTEM_PROMPT =

`Je bent een technische assistent voor operators bij Huima Drive Solutions,
een productiebedrijf. Operators stellen je vragen — meestal een probleem of
storing aan een productiemachine.

Je krijgt bij elke vraag genummerde kennisfragmenten ([1], [2], ...) uit de
interne kennisbank. Elk fragment komt uit een eerder vastgelegd kennispunt en
bevat velden zoals Probleem, Errorcode, wat er Geprobeerd is, de Oplossing,
Suggesties, en een regel "Toegevoegd door" (wie het wanneer toevoegde). Deze
fragmenten zijn je ENIGE bron.

Zo beantwoord je:

- Begin met één korte zin die de vraag duidt (meestal: welk probleem/storing).
- Wees OPLOSSINGSGERICHT, maar leg GEEN enkele oplossing op als "de" oplossing.
  Zet de mogelijkheden naast elkaar op basis van ALLE relevante fragmenten.
- Maak duidelijk onderscheid tussen:
  (a) wat eerder is geprobeerd en werkte (de oplossing);
  (b) wat is geprobeerd maar (nog) niet hielp;
  (c) suggesties, tips of opmerkingen;
```

(d) errorcodes — noem de code apart en koppel die aan wat erbij hoort.
- Verwijs waar nuttig naar WIE iets eerder probeerde en WANNEER, op basis van de "Toegevoegd door"-regel in het betreffende fragment. Koppel een naam/datum nooit aan een fragment waar die niet bij staat.
- Combineer overlappende fragmenten tot één helder overzicht; herhaal niet.

Strikte regels (anti-hallucinatie):

- Gebruik UITSLUITEND de gegeven fragmenten. Verzin niets en gebruik geen kennis van buiten de fragmenten.
- Staat het antwoord niet in de fragmenten? Zeg dan exact: " Dit kan ik niet vinden in de beschikbare kennis" Gok niet.

Vorm:

- Schrijf in het Nederlands, praktisch en voor een operator op de werkvloer.
- Bondig maar volledig: laat de opties zien, geen dictaat.
- Sluit af met een regel die de gebruikte bronnen noemt via hun nummers, bijvoorbeeld: "Bronnen: [1], [3]". De volledige klikbare bronnenlijst komt daaronder vanzelf.`;

D.5 Clarification Prompt

The clarification prompt is used only when the code-level vagueness detection fires (Appendix E.4). The model must then produce exactly one short follow-up question with two to four answer options. These options may only be based on the retrieved fragments, so the model cannot introduce new symptoms, causes or solutions. The purpose of this prompt is to help the operator narrow the question before the system gives an answer.

const ASK_VERDUIDELIJK_PROMPT =

`Je bent een technische assistent voor operators bij Huima Drive Solutions.
De vraag van de operator is te ALGEMEEN om er één antwoord op te geven. Je krijgt enkele genummerde kennisfragmenten ([1], [2], ...) die VERSCHILLENDE problemen beschrijven en allemaal maar half bij de vraag passen.

Je taak: stel ÉÉN korte, concrete VERVOLGVRAAG die de operator helpt onderscheiden welk van deze situaties hij heeft. Geef GEEN oplossing en GEEN antwoord op het probleem zelf — alleen de vervolgvraag plus een paar herkenbare keuzes.

Strikte regels (anti-hallucinatie):

- Baseer de vervolgvraag en de opties UITSLUITEND op de gegeven fragmenten. Verzin geen nieuwe symptomen, codes of oorzaken.
- Goede onderscheidende keuzes zijn bijvoorbeeld: het moment ("bij opstarten" of "tijdens het draaien"), een geluid, een trilling, of een errorcode — maar alleen als die in de fragmenten voorkomen.

Antwoord UITSLUITEND met geldige JSON, zonder markdown of uitleg, in deze vorm:

{"vraag": "<één korte vervolgvraag in het Nederlands>", "opties": ["<keuze 1>", "<keuze 2>"]}
- "opties": 2 tot 4 korte keuzes (elk max ~6 woorden) die uit de fragmenten komen en die aan de zoekvraag geplakt kunnen worden.
- Bij geen zinvolle keuzes: een lege array, maar formuleer dan toch een vraag.`;

D.6 Human Validation Logic

Human validation is used for transcribed speech. Only two named expert users are allowed to approve transcriptions (can be adjusted). They can listen to the original audio, correct the generated transcription and approve the final text. After approval, the voice note receives the status “gevalideerd”. Only then is

the approved text added to the related knowledge point and used for search. The knowledge point is then structured again and receives a new embedding. This status-based approval workflow applies to speech, OCR text from photographed notes passes through mandatory human correction of the marked uncertainties before structuring, so both forms of transcribed input are checked by a human before becoming searchable. Typed input is human-authored and can still be edited, but is not blocked by an approval step.

Appendix E: Embedding, Retrieval and Threshold Design

This appendix documents the technical choices behind the semantic search of Section 5.4.5: how the embedded text is built, how the embedding model was selected, how the retrieval query works, and how the thresholds were derived.

E.1 Composite Embedding Text

The text that is embedded for each knowledge point is the concatenation of six structured fields, in a fixed order, with empty fields skipped and the parts joined by a period and a space: probleem, vermoedelijke_oorzaak, oorzaak, wat_geprobeerd, oplossing, suggesties. Three things are deliberately not embedded. The error code is excluded because it works as a filter, not as meaning. The raw field orignele_notitie is excluded because it is the accumulation and audit record, searchability flows through the structuring step, which also means that retrieval quality depends on the quality of structuring (a limitation acknowledged in Section 5.4.8).

The function that builds this composite text exists identically in the live API and in the backfill script that repairs missing embeddings. This identity matters for correctness: distances between vectors are only comparable when every vector is derived from a text that was built in the same way.

E.2 Embedding Model Selection

The embedding model was selected through a small controlled comparison. The identical composite texts were embedded with both candidate models into two separate columns on the same data, so that the model was the only variable in the comparison. Search runs exclusively against the bge-m3 column. Switching to another embedding model later is never free: the dimensionality and the vector space change, so all embeddings would have to be regenerated.

Table E.1: Comparison of the two embedding models on identical composite text

	Nomic-embed-text	**Bge-m3 (selected)**
Dimensions	768	1024
Orientation	Multilingual	Multilingual
Memory footprint	About 270 MB	About 1.2 GB
Result on Dutch text	Weaker separation between relevant and irrelevant entries.	Clearly separated the relevant entry from irrelevant ones.

E.3 Retrieval Query

A query is embedded with the same local model and compared against the stored vectors using cosine distance[23] in pgvector. The vector channel is one SQL query: rows without an embedding are excluded,

[23] Measures how different two embedding vectors are in direction, lower cosine distance means the texts are closer in meaning.

the machine is the primary filter (no machine selected means all machines are searched), the results are ordered by distance, and the list is limited to the top results (five for search, four by default for question answering). No vector index is currently defined, at the scale of the prototype, a sequential scan is adequate, and adding an index is listed as future work for when the knowledge base grows. Embeddings are written at create and update time, if the embedding model is unreachable, the knowledge point is still saved without vector and the backfill script repairs the gap later.

Next to the vector channel, a lexical channel matches whole words and the error-code route (E.5) matches exact codes, the three ranked lists are merged with Reciprocal Rank Fusion.

E.4 Thresholds

Three constants turn raw distances into decisions. They were derived from comparison tests on small dataset available during development, in which genuinely correct matches landed at distances of roughly 0.32 to 0.44, while the first less relevant entries began at roughly 0.51-0.55.

Table E.2: Retrieval thresholds and their deviation

Constant	Value	Meaning	Deviation
MAX_AFSTAND	0.50	Maximum cosine distance for a real match. Above it, the system reports that no good match exists; weak suggestions are labelled as such.	Between the measured band of correct matches (0.32-0.44) and the first irrelevant entries (from about 0.51)
VAAG_BESTE_ANTWOORD	0.42	If even the best match is above this distance, nothing is truly on target and the system asks a clarification question instead of answering.	Just below the measured band of correct matches.
VAAG_SPREIDING	0.05	If two or more matches lie within this spread of each other, there is no clear winner and the system asks a clarification question.	Small empirical margin from the same tests.

All three values are provisional cold-start values: they were derived from comparison tests on a small dataset, and they can only be tuned reliably once the knowledge base is well filled. They are therefore presented as design parameters with a documented derivation, not as validated optima. In the code, each constant is documented together with the rule that any change must be justified in writing. The vagueness check only runs when multiple matches exist, a single clear match is answered normally.

E.5 Error-Code Route

Error codes are handled by a separate, deterministic route in which no AI is involved, because an exact code requires exact matching. The error-code field is free text and sometimes contains several codes in one cell, so the code is matched as a whole number inside the field using a word-boundary pattern: a search for 227 matches inside “272 en 227” but not inside 1227 or 2270. If no exact hit exists and the query is numeric, the route searches the neighboring codes (one lower and one higher) and labels the results as a neighbor suggestion, because related faults often sit on adjacent codes. The machine remains the primary filter. This is a documented pragmatic choice: a normalized error-code table with one code per row is the cleaner design, but the current approach solves the working problem without migrating the existing data.

E.6 Separation of Retrieval and Generation

One shared retrieval function feeds both the plain search endpoint and the question-answering endpoint. Retrieval therefore remain independently observable and testable: the ranked raw results can always be compared against the generated answer, which is used in the validation chapter to evaluate retrieval quality with and without generation.

Appendix F: Transcription Design and Fallback Logic

This appendix explains how voice notes are processed in the prototype, from recording to validated text. Voice input is important because operators can often explain a problem more easily by speaking than by typing. The transcription pipeline therefore had to be reliable, while also preventing unvalidated speech from entering the searchable knowledge base.

F.1 Pipeline Overview

When an operator records or uploads a voice note, the audio is stored immediately with the status wachtrij (queued). This store-first approach prevents input from being lost and allows the operator to continue working without waiting for transcription. The notes are then processed one at a time. This queue was chosen because the prototype runs on limited hardware, so multiple heavy AI tasks should not run at the same time.

The primary transcription engine is OpenAI, which is used because the audio may contain regional accent, machine noise and company-specific terminology. OpenAI offers great speech-to-text functionality for a considerably low price per transcription. If this cloud transcription fails, Gemini is consulted, if Gemini fails as well, a fallback local-whisper model is consulted. This fallback is multiple times slower, but keeps transcription possible when the cloud path is unavailable.

After a successful transcription, the voice note receives the status klaar and waits for human validation. Only after an approved user corrects and validates the transcript does the status become gevalideerd. The validated text is then added to the related knowledge point, after which the knowledge point is structured again and receives a new embedding. If transcription fails, the voice note is marked as mislukt with the reason recorded, while the original audio remains stored and can be retried later.

F.2 Configuration of the Local Engine

The settings of the local Whisper engine were tuned empirically on the actual workshop audio

Table F.1: Configuration of the local fallback engine.

Setting	Value	Reason
Device	CPU	Running Whisper on the GPU would evict the retrieval and generation models and break search during transcription.
Precision	Float32	The faster quantized setting[24] was measurably worse on dialect and jargon audio, quality goes before speed for a background test
Language	Dutch	All input is Dutch.

[24] Quantized inference uses lower-precision numerical formats, such as INT8 instead of FLOAT32, to reduce memory use and improve speed. (GitHub, n.d.)

Silence detection (vad_filter)	On	Voice activity detection removes silent or non-speech parts before transcription. This reduced noise-related errors during testing.
Condition_on_previous_text	Off	Whisper can use previous transcript segment as context for the next segment. In unclear audio it can cause repeated or invented text. This occurred multiple times during testing, it was therefore disabled.
Beam size	5	Beam size compares multiple possible transcriptions before choosing the most likely. A beam size of 5 is standard quality-oriented setting that balances accuracy and processing time.

F.3 Engine Recording and Comparison

Every transcription records which engine produced it, in the field transcriptie_motor. In addition, a validator can trigger a local Whisper transcription of a note that was already transcribed by the cloud engine, that output is stored in a separate column without touching the primary transcription or the status. This provides paired outputs of both engines on the same audio, which supports future engine comparison.

F.4 Crash Recovery and Idempotency

The queue itself lives in memory, but the status of each note lives in the database, which is the source of truth. On startup, all notes that are still marked wachtrij or bezig are re-queued automatically. This is safe because transcription is idempotent: repeating a half-finished transcription simply produces the same result again, without side effects. During normal operation, the bezig status prevents a note from being processed twice.

F.5 Voice Search

A separate transcribe-only endpoint powers voice search: audio goes in, text comes out, and nothing is stored. It reuses the same primary/fallback chain, but it runs synchronously, because the operator is actively waiting for the search result, and it bypasses the queue. This is a documented exception to the one-at-a-time rule: in the worst case, a voice search can compete with a queued transcription on the CPU.

Appendix G: Technical Glossary

This glossary defines the technical terms used in this thesis, in the meaning they have within this project.

Table G.1: Glossary of technical terms.

Term	Meaning in this thesis
RAG	Architecture in which a language model answers a question using retrieved knowledge entries as the only source.
Embedding	A numeric vector that represents the meaning of a text, so that texts with a similar meaning lie close together.
pgvector	PostgreSQL extension that stores embedding vectors and performs similarity search inside the relational database.
Ollama	Local model runtime that hosts the embedding model and the generation model on the prototype server, no cloud involvement
OCR (Optical Character Recognition)	Reading text, here handwritten notes, and converting it into digital text
API (Application Programming Interface)	The server component through which the interface communicates with the database and the models, in this prototype the single trust boundary of the system
Source attribution	Showing who contributed to a piece of knowledge and when. In this prototype retrieved from database records, never generated by any model.

Reference List

Aamodt, A., & Plaza, E. (1994). *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches* (Vol. 7, Number 1). AI Communications. IOS Press.

Balaguer, A., Benara, V., Cunha, R. L. de F., Filho, R. de M. E., Hendry, T., Holstein, D., Marsman, J., Mecklenburg, N., Malvar, S., Nunes, L. O., Padilha, R., Sharp, M., Silva, B., Sharma, S., Aski, V., & Chandra, R. (2024). *RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture*.

Baranwal, V. (2025). *database normalization*. DigitalOcean.

Bassil, Y. (2012). Expert PC Troubleshooter with Fuzzy-Logic and Self-Learning Support. *International Journal of Artificial Intelligence & Applications*, *3*(2), 11–21. https://doi.org/10.5121/ijaia.2012.3202

Bian, H. (2025). *LLM-empowered knowledge graph construction: A survey*.

Bowen, G. (2009). Document Analysis as a Qualitative Research Method. *Qualitative Research Journal*, *9*, 27–40. https://doi.org/10.3316/QRJ0902027

Burggraef, P., Adlon, T., Steinberg, F., Broehl, F., Moriz, A., Engeln, C., Schutz, M., & Jaworek, F. (2024). Automatic Generation of Assembly Instructions by Analyzing Process Recordings - A Concept Overview. *Procedia CIRP*, *126*, 775–780. https://doi.org/10.1016/j.procir.2024.08.343

Cho, Y. (2026). Automated Structuring and Analysis of Unstructured Equipment Maintenance Text Data in Manufacturing Using Generative AI Models: A Comparative Study of Pre-Trained Language Models. *Applied Sciences (Switzerland)*, *16*(4). https://doi.org/10.3390/app16041969

Cormack, G. , V., Clarke, C. L. A., & Buttcher, S. (2009). *Reciprocal Rank Fusion outperforms Condorcet and individual Rank Learning Methods*.

Dalkir, K. (2025). *Knowledge management in theory and practice*.

Davenport, T., & Prusak, L. (1998). Working Knowledge: How Organizations Manage What They Know. In *Ubiquity* (Vol. 1). https://doi.org/10.1145/348772.348775

Durst, S., & Wilhelm, S. (2012). Knowledge management and succession planning in SMEs. *Journal of Knowledge Management*, *16*, 637–649. https://doi.org/10.1108/13673271211246194

Fernandes, R. C., Machado, T. M., Onisto, H. J., Muñoz, A. D., Silva, R. O., Domingues, L. R., Fonseca, G. C., Bertuzzo, J. E., Pereira, M. T., Biazotto, B., & Costa, E. T. (2018). *A rule-based system proposal to aid in the evaluation and decision-making in external beam radiation treatment planning*.

Gavrilova, T., & Andreeva, T. (2012). Knowledge elicitation techniques in a knowledge management context. In *Journal of Knowledge Management* (Vol. 16, Number 4, pp. 523–537). https://doi.org/10.1108/13673271211246112

GitHub. (n.d.). *Quantization*.

Gomstyn, A., & Jonker, A. (2026, May 6). *What is enterprise search?* IBM.

Hansen, M. T., Nohria, N., & Tierney, T. (1999). What's your strategy for managing knowledge? *Harvard Business Review*, *77*(2), 106–116, 187.

Heerkens, H., & van Winden, A. (2021). *Solving Managerial Problems Systematically*. Routledge. https://doi.org/10.4324/9781003186038

Hoerner, L., Schamberger, M., & Bodendorf, F. (2023). Using Tacit Expert Knowledge to Support Shop-floor Operators Through a Knowledge-based Assistance System. *Computer Supported Cooperative Work: CSCW: An International Journal*, *32*(1), 55–91. https://doi.org/10.1007/s10606-022-09445-4

Hoffman, R. R., Shadbolt, N. R., Burton, M. A., & Klein, G. (1995). Eliciting knowledge from experts: a methodological analysis. *Organizational Behavior and Human Decision Processes*, *62(2)*, 129–158.

Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. De, Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge Graphs. *ACM Comput. Surv.*, *54*(4). https://doi.org/10.1145/3447772

Joe, C., Yoong, P., & Patel, K. (2013). Knowledge loss when older experts leave knowledge-intensive organisations. *Journal of Knowledge Management*, *17*(6), 913–927. https://doi.org/10.1108/JKM-04-2013-0137

Johnson, T. L., Fletcher, S. R., Baker, W., & Charles, R. L. (2019). How and why we need to capture tacit knowledge in manufacturing: Case studies of visual inspection. *Applied Ergonomics*, *74*, 1–9. https://doi.org/10.1016/j.apergo.2018.07.016

Karpukhin, V., O˘ Guz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-T. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. https://github.com/facebookresearch/DPR.

Kernan Freire, S., Wang, C., Ruiz-Arenas, S., & Niforatos, E. (2023, April 19). Tacit Knowledge Elicitation for Shop-floor Workers with an Intelligent Assistant. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3544549.3585755

Kommineni, V. K., König-Ries, B., & Samuel, S. (2024). *From human experts to machines: An LLM supported approach to ontology and knowledge graph construction*.

Lee, J., Kim, J. Y., Kim, H., Lee, I., & Ryu, S. (2025). IM-Chat: A multi-agent LLM framework integrating tool-calling and diffusion modeling for knowledge transfer in injection molding industry. *Journal of Manufacturing Systems*, *83*, 839–855. https://doi.org/10.1016/j.jmsy.2025.11.007

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. https://doi.org/10.48550/arXiv.2005.11401

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Sage.

Madsen, E. S., Riis, J. O., & Waehrens, B. V. (2008). The knowledge dimension of manufacturing transfers: A method for identifying hidden knowledge. *Strategic Outsourcing: An International Journal*, *1*(3), 198–209. https://doi.org/10.1108/17538290810915272

Möller, H. (2015). Expert system to troubleshoot in manufacturing plant: Experiential knowledge simplifies diagnostics and maintenance. *ADHESION ADHESIVES&SEALANTS*, *12*, 10–13. https://doi.org/10.1007/s35784-015-0517-5

Naqvi, S. M. R., Ghufran, M., Varnier, C., Nicod, J. M., Javed, K., & Zerhouni, N. (2024). Unlocking maintenance insights in industrial text through semantic search. *Computers in Industry*, *157–158*. https://doi.org/10.1016/j.compind.2024.104083

Neon. (n.d.). *Optimize pgvector search*.

Nonaka, I., Toyama, R., & Konno, N. (2000). SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. *Long Range Planning*, *33*(1), 5–34. https://doi.org/https://doi.org/10.1016/S0024-6301(99)00115-6

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., & Taylor, J. (2019). Industry-scale knowledge graphs. *Communications of the ACM*, *62*(8), 36–43. https://doi.org/10.1145/3331166

Pinecone. (n.d.). *semantic-search*. Pinecone.

Polanyi, M. (1966). *The Tacit Dimension*.

Qdrant. (n.d.). *Migrate to a New Embedding Model with Zero Downtime in Qdrant*. Qdrant.Tech.

Qi, J., Sarti, G., Fernández, R., & Bisazza, A. (2024). *Model Internals-based Answer Attribution for Trustworthy Retrieval-Augmented Generation*. https://github.com/

Radford, A., Kim, J. W., Xu, T., Brockman, G., Mcleavey, C., & Sutskever, I. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision*. https://github.com/openai/

Rasmussen, J. (1983). Skills, Rules, And Knowledge: signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems,Man, and Cybernetics*, *SMC-13(3)*, 257–266.

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. https://github.com/UKPLab/

Sarhan, A., Martin, L., El Souri, M., & Gao, J. (2022). Capturing Manufacturing Knowledge Using Augmented Reality Technologies. *Advances in Transdisciplinary Engineering*, *25*, 283–288. https://doi.org/10.3233/ATDE220604

Saunders, M., Lewis, P., & Thornhill, A. (2023). *Research Methods for Business Students*.

Suhonen, S. J., & Kankaanpää, R. A. (2025). TACIT KNOWLEDGE TRANSFER IN FINNISH MANUFACTURING COMPANIES. *SEFI 2025 - 53rd Annual Conference of the European Society for Engineering Education: Engineering and Society, Proceedings*, 1822–1830. https://doi.org/10.5281/zenodo.17631535

Wan, S., Li, D., Gao, J., & Li, J. (2019). A knowledge based machine tool maintenance planning system using case-based reasoning techniques. *Robotics and Computer-Integrated Manufacturing*, *58*, 80–96. https://doi.org/10.1016/j.rcim.2019.01.012

Wang, J., Zhuang, W., Wu, X., Chen, C., Bao, J., & Li, X. (2026). Explicating visual tacit knowledge in industrial welding inspection with context-aware cognitive pathway graph. *Journal of Manufacturing Systems*, *85*, 366–385. https://doi.org/10.1016/j.jmsy.2026.02.001

Yin, R. K. ., & Campbell, D. T. . (2018). *Case study research and applications : design and methods*. SAGE Publications, Inc.

Yoshida, I., Teramoto, Y., Tabata, H., Han, C., & Hashimoto, H. (2011). Tacit knowledge extraction of skillful operation from expert engineers. *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 554–559. https://doi.org/10.1109/AIM.2011.6027058

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., Shi, F., & Shi, S. (2025). 🧜 Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Computational Linguistics*, *51*(4), 1373–1418. https://doi.org/10.1162/coli.a.16