Frankfurt University of Applied Sciences

Faculty of Computer Science and Engineering

Implementation and Evaluation of an Enterprise Architect Chatbot

Using a RAG-Based Approach

Thesis to Obtain the Academic Degree

Master of Science (M.Sc.)

Submitted by

Hendrik Gruber

Matriculation Number: 1458240

Advisor : Prof. Dr. Jürgen Jung

Co-Advisor : Dr. Rainer Schlör

DECLARATION (ERKLÄRUNG)

I hereby assure that I wrote the present work independently and that I did not use any other sources than those given in the bibliography.

All passages that are taken verbatim or correspondingly from published or not yet published sources are marked as such.

The drawings or images in this work were created by myself or provided with a corresponding source reference.

This work has not been submitted to any other examination authority in the same or a similar form.

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Frankfurt a.M., 16. April, 2026

Hendrik Gruber

ABSTRACT

Lorem ipsum ...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI	Artificial Intelligence
APOC	Awesome Procedures on Cypher
AR	Action Research
EA	Enterprise Architect / Architecture
EAM	Enterprise Architecture Management
GenAI	Generative Artificial Intelligence
IT	Information Technology
MCP	Model Context Protocol
PAR	Participatory Action Research
RAG	Retrieval-Augmented Generation
UI	User Interface

USAGE OF GENERATIVE AI

OpenAI GPT-5 and GPT-5.2 (OpenAI, 2025) were used in order to find sources and summarize them. OpenAI GPT-5 and GPT-5.2 (OpenAI, 2025) were used in order to build both transpilers and the node.js backend server.

Part I

THESIS

1 INTRODUCTION

1.1 MOTIVATION AND THESIS QUESTION

In the field of define that we are building a RAG system which will be referred to as a chatbot. it is not an agent, as it is not executing anything.

1.2 RESEARCH METHOD

Action Research (AR) was applied in order to gain scientific value out of the developed prototype. The advantage of this research method is that it is very supportive of the development process for information systems. According to Baskerville (1999), all types of action research have the following four characteristics in common: An orientation towards developing, a focus on a specific problem, an iterative process, and a collaboration amongst participants. This is applicable to the work at hand because a prototype is being developed for a specific problem. The development cycle is conducted iteratively and collaboratively with different stakeholders. More details on this are described in chapter **??** [2]

2 TERMINOLOGY AND TECHNOLOGY

This chapter goes in detail on the terminology and technology that will be relevant for the reader to have a foundational understanding of the rest of this thesis. Later chapters will build upon these concepts and pieces of technology.

2.1 TERMINOLOGY

2.1.1 *Enterprise Architecture Management*

Enterprise Architecture Management (EAM) can be summarized as being the bridge between the business and IT departments of an enterprise. The goal is to implement information technology that is aligned with the business needs of the company. This is in contrast to the IT department implementing information technology for the sake of implementing information technology, which people in IT are often fond of doing [1]. An unwanted situation would then be when the IT department falls into a siloed way of thinking where they are decoupled from the rest of the company. EAM helps to ensure that the implemented information technology is achieving the right things, namely supporting the business capabilities and processes. [11, pg. 2-3]

Enterprise Architecture can benefit a company in various ways.

Diese Quelle hierfür vielleicht verwenden: [10] Die Quelle hat vor allem eine Menge Fragen aus der echten welt, was EAs sich im alltag fragen. diese liste kann als referenz dienen für meinen copilot. Diese Quelle hilft da vielleicht auch nochmal um in die Tiefe zu gehen. [3]

2.1.2 *Enterprise Architect*

A common challenge for an EA is dealing with the heterogenous nature of an application landscape [11, pg. 6]

Erwähnen, dass eine Herausforderung des EAs es ist, dass die vorzunehmenden Änderungen zwar von einem high-level POV einfach aussehen, aber in den details viele herausforderungen stecken. zB stakeholder management (jede Applikation hat einen eigenen verantwortlichen, viele schnittstellen der applikationen, etc.)

2.1.3 *Business Capabilities*

2.1.4 *Architecture Diagrams*

2.1.4.1 *Application Landscape*

2.1.4.2 *EAM Modeling Tool - Archi*

2.2 TECHNOLOGY

2.2.1 *Large Language Models*

LLMs are capable of supporting in language-related tasks where text needs to be generated, translated, summarized, analysed, or questions answered [9].

[17] describes what llms are and why they are not good with domain specific information and how that causes them to hallucinate.

2.2.2 *Graph Database*

2.2.3 *Retrieval Augmented Generation*

2.2.4 *neo4j and the Cypher Query Language*

2.2.5 *Model Context Protocol*

Also explain (maybe in the SOTA) how it is able to understand the schema of the graphdatabase (by calling get-schema when initializing the MCP it knows how my graph database is setup).

2.2.6 *XML and Transpilers*

As will be shown later in section xyz, the working prototype uses a domain-specific, source-to-source transpiler to turn the archimate-exported XML data into Cypher queries to insert the EA data into the graph database.

This source has a bit of good information on why XML is good as a structure language[16] "Simply speaking, eXtensible Markup Language (XML) is a data architecture connecting meta-data and data. The architecture's defining feature is the hierar-chical network of nodes. Every node in the XML structure is connected somehow to any other node; also, being a hierarchy, every node is either subordinate or su-perordinate to another node, as shown in the tree structure in Figure 1. Further, XML "provides a standard syntax for the mark-up of data and documents" (Watt 2002:1). The syntax along with the hierarchical network structure make XML doc-uments exhaustively searchable and therefore useful for linguistic research."

More detailed information on the transpiler is described in section xyz.

2.2.7 *APOC and XPATH*

Describe how apoc + xpath work and what this has to do with the neo4j cyphers. this might be better in the methodology section as a quick paragraph, but it has to be described somewhere.

3 CURRENT STATE OF THE ART

Briefly explain why a literature analysis is important. Define the scope (what fields you looked at, which databases, what keywords). Define the research method and how you narrowed it down from x sources to y sources.i

3.1 ENTERPRISE ARCHITECTURE MANAGEMENT

theories, digital twin efforts, EA tool landscapeStandards or frameworks (e.g., TOGAF, ArchiMate, IATA ONE Record, LeanIX). Theoretical foundations (auch auf prozessmanagment eingehen, wie der aktuelle Prozess aussieht, wenn die Landschaft geändert werden soll) Current tools and methods Research prototypes in EA

Authors Jung and Fraunholz 2021 [11] lay foundational work from which many EAM concepts can be derived.

3.2 LARGE LANGUAGE MODELS, CONVERSATIONAL AGENTS, AND RETRIEVAL-AUGEMENTED GENERATION

strengths, hallucination issues, graph-RAG enhancements Theoretical foundations Current tools and methods

This paper covers how ai tools are more scalable than manual expertise analysis of things. The source is highly relevant. Look at the summary in notebookLM. 05.10.25 [8]

This 2025 paper has ideas on how changing knowledge-graphs (e.g. through updates) can be handled [12]. It looks at temporal data and how to handle it. This might be relevant since addressing how a changing application landscape can be handled will probably be a challenge.

This paper gives an overview on how to control the dialog sequence and also notes 4 types of dialog options for chatbots in the related works section: [13].

This paper [20] covers how a chatbot can support in task-planning and output generation. Might be helpful in understanding how my chatbot can tell the EA how to conduct changes in the application landscape.

This paper [5] states how proactive dialogue systems work and can be improved. It goes into detail on 3 types of dialogue systems: clarification, target-guided, and non-collaborative dialogues. All 3 of these have a certain relevance for the EAM Chatbot.

RAG: Geh darauf ein, was es für unterschiedliche Chunking methoden gibt, wie man ein Buch runterbringt, und was das alles für vor und nachteile hat. auch welche tools es gibt (neo4j) ist wichtig.

3.3 COMPARABLE PROJECTS AND PROTOTYPES

Proof-of-concepts, research prototypes, industry whitepapers, GitHub projects.

Tools like ChatEA, LeanIX AI features, or Microsoft Copilot integrations in architecture/governance.

A prototypical graph-based RAG approach for text-summarization has been created by Microsoft: y[6]. The accompanying paper is here: [7]

Dragon1 needs to be detailed here!

3.4 EVALUATIONS AND LIMITATIONS

Studies analyzing strengths/weaknesses of RAG, embedding quality, hallucination mitigation.

Papers about user interaction with EA tools, chatbot evaluation frameworks, usability challenges

This paper [19] gives a standardized method and framework for evaluating coversational AI agents.

This paper [14] proposes a benchmark for open-ended multi-turn conversational agents. I think this paper focuses more on evaluating agents and comparing their results, but maybe i can copy their evaluation methods and benchmarks?

4

METHODOLOGY

This chapter describes how the chatbot “Masutā” (pronounced “mah-soo-taa” - a phonetic adaptation of the English term "master" into Japanese katakana) was developed. It provides further insight into the applied research method and the process through which the prototype was created. This gives the reader the understanding of how the research and implementation was conducted before discussing the implementation details in chapter 7.

4.1 ACTION RESEARCH DESIGN

Action Research (AR) is a research method which is highly applicable when developing an information system such as the one presented in this paper. The advantage of AR is that a large focus can be laid on the development of a system while still achieving an academic benefit. This allows for a very explorative approach to developing an information system.

Cyclical phases are central to the concept of Action Research. Baskerville [2] describes Action Research as an iterative process consisting of five steps within a single cycle. Other sources, such as Cornish et al. [4], propose variations with fewer (usually three) phases within a cycle; however these models also boil down to the same concepts. Across the literature, Action Research cycles follow the same structure: planning what should be done in the new cycle, taking action, and evaluating the outcome of the completed cycle before moving on to the next one [2, 4].

The paper at hand applied a cycle using the following five steps according to Baskerville [2]: diagnosing, action planning, action taking, evaluating, and specifying learning. The reason for choosing five cycles instead of three is the benefit of describing the development process in more detail. This five-step-cycle including the preliminary and subsequent steps are summarized in figure **to do create a drawing of the cycles**.

Each cycle lasted between three and four weeks. The sources used do not mention how long a single cycle should last. However, a cycle of two to three weeks were deemed as reasonable for the development of Masutā because status updates were held at the end of each cycle and with the given amount of time for the cycle, there was enough progress to discuss in these meetings. Participatory Action Research (PAR) goes a step further in creating a more collaborative environment between the researcher and client participants. Instead of leaving the theorizing up to the researcher, new information and ideas are thought up together with the client participants, giving both parties an active role. This is beneficial because the client participants often have

both theoretical and practical knowledge of the subject matter being worked on. [2]

The following subsections explain the research design of the applied action research. For the exhaustive action research protocol, refer to chapter A of the appendix.

4.2 ACTION RESEARCH SETUP

The domain of EAM was focused on within this research. In particular, the study addressed application landscapes, business capability maps, as well as the relationships between these two architectural artifacts. These artifacts are commonly used to support documenting an enterprise's landscapes are used to align an enterprise's IT with its strategic business objectives.

Due to their structural complexity with heterogenous data sources and multiple stakeholders involved, these artifacts are often large and difficult to interpret. Maintaining an overview can be especially challenging for junior level enterprise architects. This challenge motivates the exploration of AI-supported solutions that enable conversational interaction with the architecture, rather than manually navigating the complex diagrams.

The research was conducted in close collaboration with the academic supervisor and a co-advisor. The academic supervisor gave academic guidance and supported the structuring of the research process to ensure academic relevance. The co-advisor acted as the domain expert and the critical partner, contributing practical insights to ground the research in real-world relevance. The author of this thesis assumed the role of the researcher, implementing the action research cycles, validating findings, and planning subsequent steps in coordination with the other stakeholders.

At the outset of the research, the general problem space was clear, but the potential solution was only vaguely defined. The co-advisor initialized the research with a vague vision of a centralized system containing all enterprise architecture information, which can be interacted with via natural language. The motivation for this was to reduce the effort required to interpret the enterprise architecture artifacts.

However, in the early stages of the project, not only were the technical details of the potential solution unclear, but also the feasibility of such a system. Early on it was mutually agreed upon that LLMs and AI would play a key role in realizing this, even though the data structures, mechanisms, storage options, and interaction patterns were still open. Consequently, the initial problem statement was intentionally formulated at a high level to provide a suitable starting point for iterative exploration. Through the iterative development cycles, this high level problem statement was refined into a concrete problem definition and technical solution.

4.3 ACTION RESEARCH CYLCES

The above mentioned vague details of the implementation became clear during the development cycles, leading to a final architecture with a clear structure. Each cycle began with a meeting between all three stakeholders. The end of one cycle was the beginning of the next one and was concluded in the same meeting. Table 4.1 summarizes these cycles.

4.3.1 *Cycle 1*

Diagnosis: The initial problem statement lacked concrete technical formulation and the feasibility of natural language interaction with enterprise architecture data was unclear.

The co-advisor outlined typical enterprise architecture workflows and the tools used to document and interact with architectural artifacts in practice. An initial idea was propsed of an AI-based black-box system capable of ingesting architecture data and deriving its own internal representations. The feasibility as well as academic applicability of such a black-box system were critically questioned, particularly because of the limited transparency of the innern mechanisms and how to test this. As an alternative, the academic supervisor proposed an explicit knowledge-graph approach in which a knowledge-graph is built which can be used to supplement generated answers.

The key distinction between the black-box and white-box approaches is the transparency. While the black-box approach autonomously creates an internal representation of the information, the white-box approach requires the manual design and implementation of an explicit technical architecture.

These discussions were necessary in order to scope the solution space. Early solution ideas included a chat that would support enterprise architects in exploring and improving application landscapes, for example by identifying inconsistencies or incomplete application landscapes.

Action Planning: The first cycle aimed to develop a proof of concept that enables conversational access to a knowledge-graph consisting of enterprise architecture knowledge grounded in textbook-based domain information. It was decided that a single-agent architecture will be used, as multi-agent architectures were considered unnecessarily complex for an initial proof of concept.

At this stage, the research methodology had not yet been explicitly defined as action research, and it was initially assumed that the resulting prototype would be evaluated through an expert-interview.

Action Taken: An initial knowledge-graph was constructed based on content from the textbook *Masterclass Enterprise Architecture Management* [11]. The textual content was iteratively preprocessed and transformed into graph representations, with successive refinements applied to improve the mapping of domain concepts into nodes and relationships. After integrating the full textbook into the knowledge-graph, additional prototypical data was

incorporated to enable querying the knowledge-graph of concrete architectural data.

Evaluation: The state of the proof of concept after the first cycle was demonstrated to both advisors and evaluated qualitatively through open discussions. Both advisors positively assessed the feasibility of the approach, and the co-advisor confirmed that an explicit knowledge-graph-based solution will be a viable direction for further development.

Learning: Compared to the beginning of the cycle, in which the feasibility of a centralized knowledge-graph was uncertain, the first cycle demonstrated that an explicit, white-box approach represents a practical path forward. The cycle also revealed key challenges related to the transformation of heterogeneous data into graph structures and the effective querying of such representations. Finally, it became evident that further iterations would be required to systematically address these challenges with an exploratory development process.

4.3.2 *Cycle 2*

Diagnosis: a next idea would be to feed in more textbooks. idea would be to compare my results to an existing copilot such as the Microsoft Copilot Studio. An idea came up that it might make sense to have 1 database for the textbook information and 1 database for the enterprise architecture data. we agreed to use one database in hopes of achieving better results as nodes between the textbook and the enterprise data may be connected together in the single database, achieving better results. This is also the phase, where we started discussing potential data sets. the problem here being that the co-advisor is not allowed to simply export his company data and give it to me. we also started discussing how to evaluate my system. we discussed potential experiments and evaluation methods - including testing the knowledge graph directly. we agreed that i would use the data from the assignment of our uni's course "Enterprise Architecture Management" which i completed in the previous semester. this dataset contains an application landscape and a business capability map for a fictitious company named "SpeedParcel".

Action Planning:
Action Taken:
Evaluation:
Learning:

4.3.3 *Cycle 3*

Diagnosis:
Action Planning:
Action Taken:
Evaluation:
Learning:

4.3.4 *Cycle 4*

Diagnosis:
Action Planning:
Action Taken:
Evaluation:
Learning:

The main challenge of this cycle was refactoring the cyphers that get called agains the database. the reason being that the previous method for generating text-to-cypher was easy because it was only calling the speedparcel database and the schema of this was entirely known. however, during this sprint we got new data exported from architecture diagrams in archi. this data is well structured but the schema is unknown. this means that the text-to-cypher has to be able to query the graph database agnostic of any schema in it. meaning, the text-to-cypher has to be completely refactored in order to be able to reliable query the dataset with unknown contents.

The source from Wan i 5.2.2 [18] explains how he created a 3-step-preprocessing in order to query the database agnostically. what i did is not 1-to-1 the same thing, but i borrowed the ideas. the main change being changing the CALL db.schema.visualization() from before to the APOC call apoc.meta.schema() which apparently returns more sensible information. that combined with the SHOW INDEXES call give a better result (i assume - i'm writing this before testing just to get my ideas out of my head lol have fun rewriting this. i wrote this in Bremen on 28.12 xoxo)

Table 4.1: Overview of Action Research Cycles

Cycle	Diagnosis	Action Planning	Action Taken	Evaluation	Learning / Outcome
Cycle 1	Todo	Todo	Todo	Todo	Todo
Cycle 2	Todo	Todo	Todo	Todo	Todo
Cycle 3	Todo	Todo	Todo	Todo	Todo
Cycle 4	Todo	Todo	Todo	Todo	Todo

4.4 FINAL MEETING

Todo: den Termin vor Ort beschreiben Purpose of the meeting, who participated, what was validated, what kind of feedback was collected.

As described in this chapter, it becomes clear why action research was an invaluable methodology. From unclear beginnings containing only a vague vision for a final prototype, each development cycle contributed to the final architecture being clear and goal oriented. Each phase helped to examine what was possible from a technical standpoint as well as how to move forward. This supported the explorative nature of the project.

5 IMPLEMENTATION

The methodology chapter describes how and why we did things. This chapter describes what we built, including architecture, data processing, sequence diagrams, retrieval, UI and interaction flow.

Do not evaluate here. Only describe what is done.

5.1 DATA USED

5.2 FINISHED PROTOTYPE

Explain here, what the finished prototype is (including architecture diagram, sequence diagram, etc.). or should this be an entirely separate chapter? describing this somewhere here makes sense though before moving on to the experiments done with the prototype.

Mention that this is a "ephemeral conversation memory" and why that is. We will probably need some kind of source on this.

Also mention that we agreed to test my system using exported XML files from Archimate. So other tools like

5.2.1 *Finished Architecture*

Explain in detail here what each component of the finished architecture is and how it all fits together.

5.2.1.1 *XML Transpiler*

Explain in detail here how the XML transpiler takes an XML file as the input and transpiles it into Cypher. it is model-to-model and thus touches on the subjects compiler construction, model-driven engineering, graph databases, and enterprise architecture tooling.

Show how fine-tuning the system prompt can have an effect on the results. E.g. if in the context it says to answer within 1-2 sentences or to answer in 4-6 sentences. show examples of how small things in the prompt can have a large impact.

5.2.2 *Generating Text-to-Cypher Independent of the Database Schema*

Highlight this as the main challenge of the thesis!

This source describes how and why agnostic cyphers can and should be generated. [18] also, check their sources and use those as well. It also mentions that a challenge when giving the LLM the necessary context is that it

gets overloaded with information and that the context length may get exceeded (page 5).

This source [15] talks about how natural language text can be used to **create** the knowledge graph. Def use this when explaining how i created my speedparcel imports.

The whole system prompting this is called "in-context learning (ICL)". search for sources that support this.

6 EXPERIMENTS

What we did with the finished prototype and how we tested it. Prompts we used, cases we built, edge cases, etc.

show some test cases here. break down an answer like "i want to remove the application StatManPlus. What do i have to look out for as an enterprise architect?". the returned answer goes into a lot of depth as this is a level 3 question. break down the result of the query and how the result pulls information about the application but also pulls information from the Lehrbuch database.

```
MATCH (a:Application {name: 'StatManPlus'})
OPTIONAL MATCH (a)-[:SUPPORTS]->(cap:Capability)
WITH a, collect(DISTINCT cap) AS caps
UNWIND (CASE WHEN size(caps) > 0 THEN caps ELSE [NULL] END) AS cap
OPTIONAL MATCH (cap)-[:HAS_CHILD*0..]->(desc:Capability)
WITH a, caps, collect(DISTINCT desc) AS cap_impacted
OPTIONAL MATCH (a)-[:DATA_FLOW]->(outApp:Application)
WITH a, caps, cap_impacted, collect(DISTINCT outApp.name) AS outgoing_apps
OPTIONAL MATCH (inApp:Application)-[:DATA_FLOW]->(a)
WITH a, caps, cap_impacted, outgoing_apps, collect(DISTINCT inApp.name) AS
    incoming_apps
OPTIONAL MATCH (ch:Chunk)
WHERE toLower(coalesce(ch.text,'')) CONTAINS toLower(a.name)
   OR toLower(coalesce(ch.table_summary,'')) CONTAINS toLower(a.name)
   OR toLower(coalesce(ch.title,'')) CONTAINS toLower(a.name)
   OR toLower(coalesce(ch.definition,'')) CONTAINS toLower(a.name)
WITH a, caps, cap_impacted, outgoing_apps, incoming_apps,
     collect(DISTINCT {key: ch.key, title: ch.title, snippet: ch.
    table_summary, text: ch.text}) AS app_chunks
OPTIONAL MATCH (guid:Chunk)
WHERE toLower(coalesce(guid.text,'')) CONTAINS 'decommission'
   OR toLower(coalesce(guid.text,'')) CONTAINS 'retire'
   OR toLower(coalesce(guid.text,'')) CONTAINS 'sunset'
   OR toLower(coalesce(guid.title,'')) CONTAINS 'decommission'
   OR toLower(coalesce(guid.title,'')) CONTAINS 'retire'
   OR toLower(coalesce(guid.title,'')) CONTAINS 'sunset'
   OR toLower(coalesce(guid.definition,'')) CONTAINS 'decommission'
   OR toLower(coalesce(guid.definition,'')) CONTAINS 'retire'
   OR toLower(coalesce(guid.definition,'')) CONTAINS 'sunset'
   OR toLower(coalesce(guid.table_summary,'')) CONTAINS 'decommission'
   OR toLower(coalesce(guid.table_summary,'')) CONTAINS 'retire'
   OR toLower(coalesce(guid.table_summary,'')) CONTAINS 'sunset'
WITH a, caps, cap_impacted, outgoing_apps, incoming_apps, app_chunks,
     collect(DISTINCT {key: guid.key, title: guid.title, snippet: guid.
    table_summary, text: guid.text}) AS guidance_chunks
RETURN a.name AS application,
       [c IN caps WHERE c IS NOT NULL | c.id] AS supported_capability_ids,
       [g IN cap_impacted WHERE g IS NOT NULL | g.id] AS
    impacted_capability_ids,
       outgoing_apps AS outgoing_data_flows_to_apps,
```

```
        incoming_apps AS incoming_data_flows_from_apps,
        app_chunks AS chunks_referencing_application,
        guidance_chunks AS chunks_with_decommission_guidance
```

Listing 6.1: Cypher query to retrieve application impact and decommission guidance

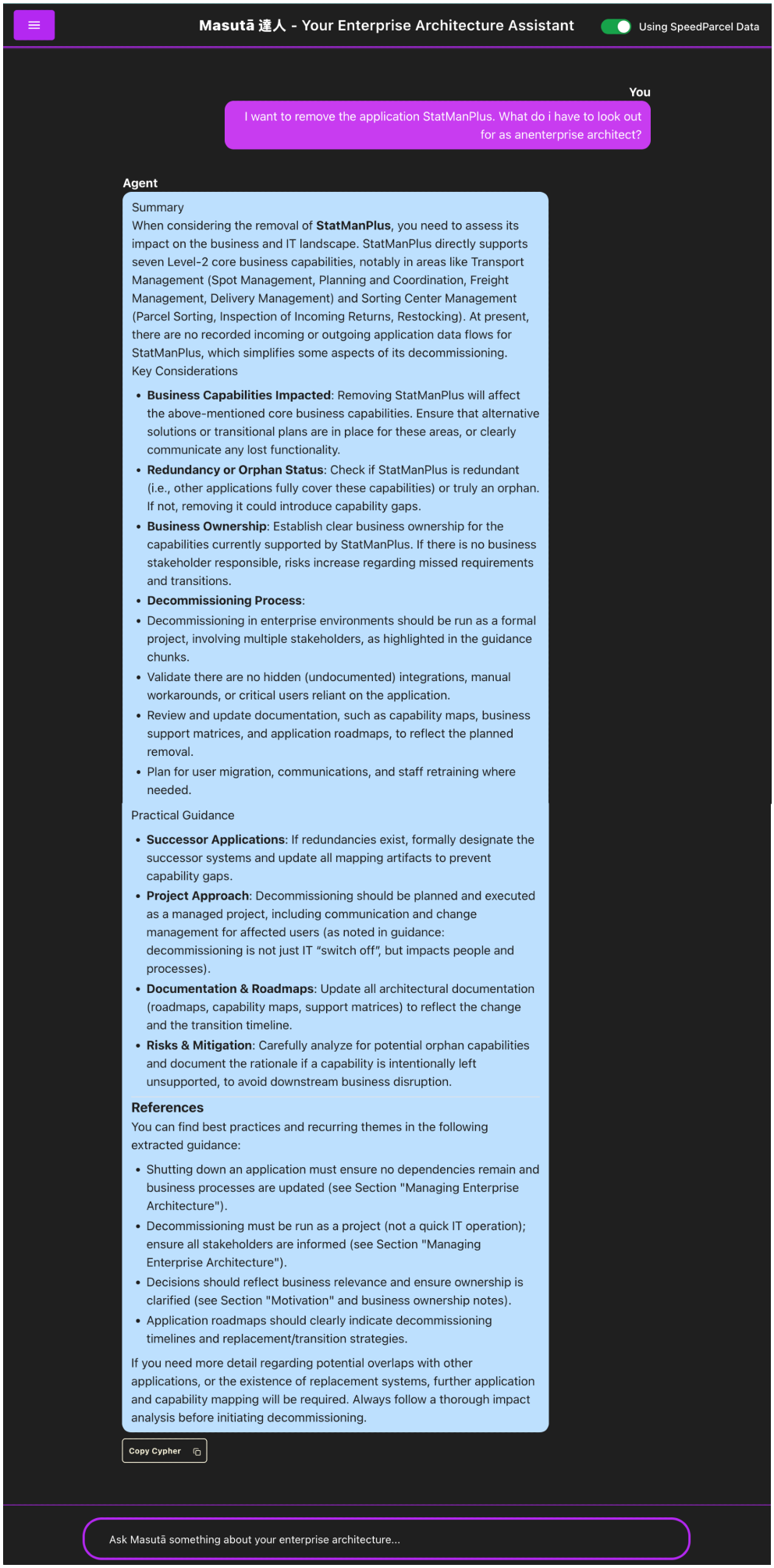

Figure 6.1: Raw response returned from the Cypher in listing 6.1. Notice that the

application	supported_capability_ids	impacted_capability_ids	outg	incon	chunks_referencing_application	chunks_with_decommission_guidance

"StatManPlus"	["cap_l2_51_spot_management", "cap_l2_52_planning_and_coordination", "cap_l2_53_freight_management", "cap_l2_54_delivery_management", "cap_l2_58_parcel_sorting", "cap_l2_59_inspection_of_incoming_returns", "cap_l2_60_restocking"]	["cap_l2_51_spot_management", "cap_l2_52_planning_and_coordination", "cap_l2_53_freight_management", "cap_l2_54_delivery_management", "cap_l2_58_parcel_sorting", "cap_l2_59_inspection_of_incoming_returns", "cap_l2_60_restocking"]	[]	[]	[{
					snippet:null,
					text:"Capability "Spot Management" (Level 2 in band "Core Capabilities", under L1 capability "Transport Management") is supported by: StatManPlus (flag "x").",
					title:null,
					key:"c7e0b12a9b94ca889c7e13d9e4cea9e504e5e0d2_sec_transport_management_chunk_spot_management"
					}, {
					snippet:null,
					text:"Capability "Planning and Coordination" (Level 2 in band "Core Capabilities", under L1 capability "Transport Management") is supported by: StatManPlus (flag "x"), TrackDB (flag "x").",
					title:null,
					key:"c7e0b12a9b94ca889c7e13d9e4cea9e504e5e0d2_sec_transport_management_chunk_planning_and_coordination"
					}, {
					snippet:null,
					text:"Capability "Freight Management" (Level 2 in band "Core Capabilities", under L1 capability "Transport Management") is supported by: StatManPlus (flag "x").",
					title:null,
					key:"c7e0b12a9b94ca889c7e13d9e4cea9e504e5e0d2_sec_transport_management_chunk_freight_management"
					}, {
						[{
						snippet:null,
						text:"This last difference is visible in a lot of companies today. There are software systems available and used by business people, but nobody on the business side feels responsible for them. Responsibility would include maintenance (i.e. conducting projects to perform changes), support and funding (e.g. for licenses). It, furthermore, refers to making decisions on the future of individual software applications, like its decommissioning if it does no longer provide relevant value. Making decisions based on business relevance implies an assessment from the business perspective. In fact, many systems are just handed over to IT so that they keep them running but there is no business ownership. IT people then need to drive changes driven by business requirements. Ideally, these should be driven by business stakeholders. We still have to convince people in business and IT to understand the relevance of each IT system and then make sure the IT system has business relevance. We further need clear ownership within the business to take the responsibility for decisions on this IT system.",
						title:null,
						key:"27be2852-75f0-44bc-a0d8-343ee3ed7ce9_10"

Figure 6.2: Raw response returned from the Cypher in listing 6.1. Notice that the response contains both information about the applications themselves as well as textbook information on enterprise architecture management (right-most column).

7

IMPLEMENTATION

The methodology chapter describes how and why we did things. This chapter describes what we built, including architecture, data processing, sequence diagrams, retrieval, UI and interaction flow.

Do not evaluate here. Only describe what is done.

8

DISCUSSION

This chapter discusses what the answers mean. interpretation of the results, comparison with the literature, limitations, implications for EAs in practice.

9 CONCLUSION AND FUTURE WORK

Todo

9.1 CONCLUSION

9.2 FUTURE WORK

Some ideas for future work: how to improve the challenges mentioned. what other areas this could find utility in. how could access management be handled? e.g. if the database contains customer-information that not every user should be able to see, how can that be differentiated? real-world scenarios that could use my prototype and try out a pilot phase?

BIBLIOGRAPHY

[1] Shoaib Ahmed, Nazim Taskin, David J Pauleen, Jane Parker, et al. "Motivating information technology professionals: The case of New Zealand." In: *Australasian Journal of Information Systems* 21 (2017).

[2] Richard L Baskerville. "Investigating information systems with action research." In: *Communications of the association for information systems* 2.1 (1999), p. 19.

[3] Sandra Castro and Jürgen Jung. "Towards Measuring Success of Enterprise Architecture Decisions: Survey among Practitioners and Outline of a Framework." In: *Enterprise Architecture Professional Journal* (2021), pp. 1–22.

[4] Flora Cornish, Nancy Breton, Ulises Moreno-Tabarez, Jenna Delgado, Mohi Rua, Ama de Graft Aikins, and Darrin Hodgetts. "Participatory action research." In: *Nature Reviews Methods Primers* 3.1 (2023), p. 34.

[5] Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. *Prompting and Evaluating Large Language Models for Proactive Dialogues: Clarification, Target-guided, and Non-collaboration*. 2023. arXiv: 2305.13626 [cs.CL]. URL: https://arxiv.org/abs/2305.13626.

[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. "From Local to Global: A Graph RAG Approach to Query-Focused Summarization." 2024. URL: https://www.microsoft.com/en-us/research/publication/from-local-to-global-a-graph-rag-approach-to-query-focused-summarization/.

[7] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. 2025. arXiv: 2404.16130 [cs.CL]. URL: https://arxiv.org/abs/2404.16130.

[8] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. "A survey on llm-as-a-judge." In: *arXiv preprint arXiv:2411.15594* (2024).

[9] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. "Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects." In: *Authorea preprints* 1.3 (2023), pp. 1–26.

[10] Jürgen Jung. "Purpose of enterprise architecture management: investigating tangible benefits in the german logistics industry." In: *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE. 2019, pp. 25–31.

[11] Jürgen Jung and Bardo Fraunholz. *Masterclass Enterprise Architecture Management*. Springer, 2021.

[12] Dong Li, Yichen Niu, Ying Ai, Xiang Zou, Biqing Qi, and Jianxing Liu. *T-GRAG: A Dynamic GraphRAG Framework for Resolving Temporal Conflicts and Redundancy in Knowledge Retrieval*. 2025. arXiv: 2508.01680 [cs.AI]. URL: https://arxiv.org/abs/2508.01680.

[13] Zhigen Li et al. *ChatSOP: An SOP-Guided MCTS Planning Framework for Controllable LLM Dialogue Agents*. 2025. arXiv: 2407.03884 [cs.CL]. URL: https://arxiv.org/abs/2407.03884.

[14] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. "Agentbench: Evaluating llms as agents." In: *arXiv preprint arXiv:2308.03688* (2023).

[15] Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F Enguix, and Kusum Lata. "Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text." In: *International semantic web conference*. Springer. 2023, pp. 247–265.

[16] Christoph Rühlemann and Matt Gee. "Conversation Analysis and the XML method." In: *Gesprächsforschung* (2017).

[17] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. "Large language models encode clinical knowledge." In: *Nature* 620.7972 (2023), pp. 172–180.

[18] Yuwei Wan, Zheyuan Chen, Ying Liu, Chong Chen, and Michael Packianather. "Prompting large language models based on semantic schema for text-to-Cypher transformation towards domain Q&A." In: *Decision Support Systems* (2025), p. 114553.

[19] Anna Wolters, Arnold Arz von Straussenburg, and Dennis M. Riehle. "Evaluation Framework for Large Language Model-based Conversational Agents." In: July 2024.

[20] Wenshuo Zhai, Jinzhi Liao, Ziyang Chen, Bolun Su, and Xiang Zhao. "A Survey of Task Planning with Large Language Models." In: *Intelligent Computing* (2025). DOI: 10.34133/icomputing.0124.

Part II

APPENDIX

A

ACTION RESEARCH PROTOCOL

This chapter contains the full action research protocol.

A.1 PRECONDITIONS

Test: This reference should have a lowercase, small caps A if the option

A.2 CYCLE 1

Start: October 17th, 2025. End: October 31st, 2025

Diagnosis: The initial problem statement lacked concrete technical formulation and the feasibility of the natural language interaction with enterprise architecture data was unclear. The co-advisor described in detail what his daily work looks like and what tools are used at his company to document and interact with enterprise architecture data. He also showcased the existing solution Dragon1 (described in more detail in section todo).

The co-advisor had the idea of creating an AI, acting as a black-box system, that ingests the specific enterprise architecture data and creates its own internal knowledge graph. All as a black-box. The feasibility as well as academic applicability of this was questioned intensely because of the difficulty of testing such a black-box system when nothing is known about the innern mechanisms.

The academic supervisor recommended an approach using a knowledge graph and supplementing a language model with the domain specific information. He also recommended tools such as Microsoft's Copilot Studio. The academic supervisor also allowed the use of his textbook "Masterclass Enterprise Architecture Management" [11] to feed the knowledge graph with information specific to the role of an enterprise architect.

The difference between the black-box and white-box (knowledge graph) versions is that the black-box system would build its inner mechanisms itself in order to generate domain specific answers. In the white-box version the technical architecture would have to be built manually.

These steps were required in order to scope the potential solution and to think about what possible data sources could be.

First rough ideas were a chatbot with which an enterprise architect could make changes to the application landscape. This included being able to complete unfinished or incomplete application landscapes. For example, being able to interact with the chatbot to identify inconsistencies and weak spots in the landscape.

Action Planning: The goal of the first phase was clear: create a proof of concept. This meant setting up a basic LLM, feeding it with general enter-

prise architecture data and context so that it is trained to see the interaction through the eyes of an enterprise architect. On top of this a small UI to be able to interact with the proof of concept chatbot. The end-goal of this proof of concept chatbot was to be able to make recommendations on how to edit an application landscape while still covering all business capabilities. The goal until the next cycle would be to be able to ask a chatbot general questions about enterprise architecture management and receive answers based off of the textbook information fed into a knowledge graph.

It was agreed upon, that a single-agent architecture will be used, as multi-agent architectures come with more complexity and are not ideal for this use case.

At this stage, it was not clear that action research will be applied during development. At this stage it was assumed that the finished prototype will be passed through an expert interview.

Action Taken: Setup a neo4j database, created a graph representation of the first chapter of "Masterclass EAM". Created a React.js frontend which was directly connected to the graph database. Setup the first prompt context to query the database. Confirmed that the supplemented answers used the knowledge graph by comparing the textbook to the generated answers. Improved embeddings into the graph database (better nodes and relationships). Transforming the textbook into graph representations was done using a python parser which was created with the help of ChatGPT. Improvements on parsing the textbook into a graph representation. Read in the rest of the chapters from the textbook. Created a capability support matrix in excel and parsed it into the database and adapted the queries for this.

Evaluation: Both advisors were ecstatic to see the progress so far after giving a demo of the current state. The co-advisor agreed that building a knowledge-graph is the correct path forward after having seen the proof of concept in action.

Learning:

A.3 CYCLE 2

Start: October 31st, 2025. End: November 28th, 2025 **Diagnosis**: a next idea would be to feed in more textbooks. idea would be to compare my results to an existing copilot such as the Microsoft Copilot Studio. An idea came up that it might make sense to have 1 database for the textbook information and 1 database for the enterprise architecture data. we agreed to use one database in hopes of achieving better results as nodes between the textbook and the enterprise data may be connected together in the single database, achieving better results. This is also the phase, where we started discussing potential data sets. the problem here being that the co-advisor is not allowed to simply export his company data and give it to me. we also started discussing how to evaluate my system. we discussed potential experiments and evaluation methods - including testing the knowledge graph directly. we agreed that i would use the data from the assignment of our uni's course "Enterprise

Architecture Management" which i completed in the previous semester. this dataset contains an application landscape and a business capability map for a fictitious company named "SpeedParcel".

A.4 CYCLE 3

Start: November 28th, 2025. End: December 17th, 2025

A.5 CYCLE 4

Start: December 17th, 2025.End: January 23rd, 2026

A.6 FINAL MEETING

Meeting date: January 23rd, 2026. Location: Campus of the Frankfurt University of Applied Sciences.

B

INSTRUCTION MANUAL FOR MASUTĀ

Masutā is the name of the prototype enterprise architecture chatbot. Follow these instructions to download Masutā, set it up, and learn how to use it.

When executing the commands from the terminal, always execute them from the root folder of the project. The best results are achieved by executing each command individually (i.e. copy-paste the commands individually line-for-line instead of the entire block).

B.1 DOWNLOADING MASUTĀ

In order to get Masutā up and running, a few prerequisites will have to be fulfilled.

The first is to download the application code for Masutā from GitHub: https://github.com/HendrikDA/masuta-ea-chatbot-prototype

The second prerequisite is to download Docker and Docker Compose. You can find instructions on how to set this up on your machine under this link: https://docs.docker.com/compose/install/.

An optional prerequisite is to download neo4j Desktop. This allows the user to inspect the graph database within an interactive desktop application. Neo4j Desktop can be found here: https://neo4j.com/download/. However, this is not mandatory and the databases may also be inspected via neo4j's web-application.

B.2 FIRST TIME SETUP

If Masutā is being run for the first time, then two containers need to be setup so that they are populated with the default data. From within the root folder of the project, execute the following commands

```
docker compose --profile speedparcel-restore run --rm speedparcel-restore
docker compose --profile textbook-restore  run --rm textbook-restore
```

Once these commands have run through successfully, both databases are populated with the default data and are ready to be used.

B.2.1 *Hard-Resetting Everything*

If at any time you would like to hard-reset the entire application, you may run the following commands:

```
docker compose down
rm -rf $HOME/neo4j/data
rm -rf $HOME/neo4j_empty/data
```

After resetting the application, you will have to restart with the first time setup.

B.3 RUNNING THE APPLICATION

Running Masutā is as simple as running the following command:

```
docker compose up
```

With this command, four containers are started.

- The SpeedParcel neo4j database filled with example data and the textbook
- The playground neo4j database filled with only the textbook
- The MCP-Backend which is the backend node.js which by default runs under http://localhost:4000
- The frontend which by default runs under http://localhost:3000/

If Docker does not automatically open the frontend in your browser, navigate to it under http://localhost:3000/. You are now ready to start using Masutā.

B.4 FEATURE OVERVIEW

This section details the features within Masutā.

B.4.1 *Input Field and Chat History*

At the bottom of the application is an input field where the user may prompt Masutā. Pressing enter sends the prompt. The user's input prompt then appears as a chat bubble bound to the right of the chat-area. The system then starts working on answering the prompt and displays a chat bubble bound to the left of the chat-area displaying "Thinking...". As soon as the application is done thinking, the response is displayed in the same chat bubble.

Masutā's response has a button appended to it which allows the user to copy the cypher which was used to generate the answer. The user may take this copied cypher and paste it into neo4j desktop to inspect the raw response that was used to generate the answer.

B.4.2 *Toggling the Database*

At the top right of the application is a toggle switch. This allows the user to toggle between the SpeedParcel database and the playground database.

B.4.3 *Resetting the Playground Database*

Under the menu at the top left, the user is presented with the option to reset the database. This option is only available if the playground database is selected via the toggle switch. After confirming the reset in the dialog that pops up, the playground is reset to its default state with only the textbook information. The SpeedParcel database cannot be reset via the UI.

B.4.4 *Importing Custom Data*

Under the menu at the top left, the user is presented with the option to import their own data. This option is only available if the playground database is selected via the toggle switch. Within the dialog that pops up, the user can add up to 10 files either by selecting them via the file system or by dragging-and-dropping them into the dialog. Only .xml files are accepted.

After clicking the import button, the files are uploaded. If the upload was successful, an alert is shown notifying the user of this. The uploaded data remains within the realms of the Docker containers and is not uploaded to any third-party sources. The data can then be queried via the playground database.

B.4.5 *Inspect Database*

Under the menu at the top left, the user is presented with the option to view the graph data in neo4j browser their own data. This option is available for both the SpeedParcel and playground database. Clicking this opens another tab under https://console-preview.neo4j.io/tools/query. Here, the user can connect to the selected database. The connection string is displayed in a dialog within Masutā. Here, the user can inspect the graph structure and paste the cyphers used for the generated responses.

B.4.6 *Chat Context and Building on Previous Answers*

Masutā saves the chat history so that the user is able to build upon previous prompts. The chat history is set so that the single previous user prompt and corresponding response is saved. This means that a further question can be asked about the previous prompt or its response. The chat history's context only goes back one prompt and response.

B.5 EXAMPLE QUESTIONS

Questions can be broken down into three categories:

1. Category 1: Textbook questions about enterprise architecture management
2. Category 2: Basic information about the enterprise architecture data

3. Category 3: Questions that span both category 1 and 2 and are more complex than questions in category 2. These require more in-depth information about the data as well as how to deal with it as an enterprise architect.

The following list of questions serve as examples that Masutā can handle. These simply serve as ideas to test the system - the user may input their own thought up questions.

1. Questions in Category 1
    - What are the schools of enterprise architecture management?
    - How does enterprise architecture relate to town planning?
2. Questions in Category 2
    - How many capabilities are supported?
    - How many applications are in use?
    - Which application supports the most capabilities?
    - If I remove application x, which capabilities will be affected?
3. Questions in Category 3
    - We plan on removing application x. What do we have to look out for when doing so?
    - We wish to implement a new application x which covers the capabilities x, y, and z. Which existing applications may become redundant?
    - Which capabilities currently rely on single-point-of-failure applications? How should we deal with this?
    - Which capabilities are over-supported by multiple applications?

This concludes the instruction manual for Masutā. Beyond the example questions, users are encouraged to prompt Masutā to explore the example data provided in SpeedParcel or to analyze their own proprietary data after importing it, in order to gain insights into their enterprise architecture.

C

MASUTĀ CODE SNIPPETS

This chapter showcases code snippets which are of special interest as they show how key functionalities of Masutā work. The code snippets do not showcase the full application and may not complete.

C.1 SYSTEM PROMPT AND RESULTS EXPLAINER

The following code snippets showcase what meta-prompts are passed to the LLM in order to generate the cyphers via the natural language text input and how the cypher results are transformed back into natural language.

Todo: Add the system prompts here.

D

MASUTĀ EXAMPLE QUESTIONS AND ANSWERS

This chapter contains screenshots of user prompts and the system's responses. It serves as a showcase of what the application looks like and how it responds to questions.

D.1 QUESTION XYZ

todo