Fathe Muhammad Rafi

Design and Evaluation of a Trustworthy and Scalable AI Agent Framework for Knowledge-Integrated Industrial Supply Chain Analytics

School of Technology and Innovations

Industrial Systems Analytics

Master of Science in Technology

Vaasa 2026

VAASAN YLIOPISTO
School of Technology and Innovations
Author: Fathe Muhammad Rafi
Thesis title: Design and Evaluation of a Trustworthy and Scalable AI Agent Framework for Knowledge-Integrated Industrial Supply Chain Analytics
Degree: Master of Science in Technology, Industrial Systems Analytics
Supervisor: Imran Faisal
Year of graduation: 2026 **Number of pages:** 113

ABSTRACT:
The supply chains of industries contain rich structured information in relational databases and unstructured knowledge in policy documents yet the current AI methods solve particular aspects of this analytical puzzle in isolation. This thesis designs, implements, and evaluates a single, overarching AI agent framework that incorporates SQL-based data retrieval, machine learning analytics, and retrieval-augmented generation within a multi-agent orchestration of seven domains of supply chain, in a thirty-seven-table database. Notable architectural contributions are schema-linking-first dispatch on large scale schemas, domain-specialized agents, multistage safety enforcement, first-class explainability via decision traces, and dual interfaces REST and Model Context Protocol.
The framework was tested against anonymized operational data that was provided by Bangladeshi partner companies in the pharmaceutical, ready-made garments, fast-moving consumer goods and service-based supply chain sectors, as per Design Science Research methodology. A benchmark of 60 queries in 5 query types was run against the deployed system. The framework achieved 93.8% SQL execution accuracy, 86.4% coverage of the set of items to be optimally shipped, and a median response latency of 26.5 seconds, but three machine learning models of the process of demand forecasting, shipment anomaly detection and supplier risk assessment were trained, registered and exposed through the agent. Its thesis adds what is traditionally called a reference architecture, covering reliable AI agents to supply industrial supply chains and an empirical test, which is simplified by beforehand calling it a reference architecture.

Keywords: AI agent, supply chain analytics, retrieval-augmented generation, trustworthy AI, Bangladesh

Contents

1 Introduction 8
1.1 Background and Motivation 8
1.2 Problem Statement 10
1.3 Research Objectives 12
1.4 Research Questions 13
1.5 Scope and Limitations 13
1.6 Thesis Structure 15

2 Literature Review 17
2.1 Industrial Supply Chain Analytics 17
2.2 AI Agents and Agentic Systems 19
2.2.1 Foundations of AI Agents 19
2.2.2 Agent Architectures and Reasoning Patterns 19
2.2.3 Multi-Agent Systems and Area of Expertise 20
2.2.4 Tool Use and Function Calling 21
2.2.5 Agent Interoperability and the Model Context Protocol 21
2.3 Retrieval-Augmented Generation 22
2.3.1 Foundations of RAG 22
2.3.2 Advanced RAG Techniques 22
2.3.3 RAG for Structured and Semi-Structured Data 23
2.3.4 Graph-Based Retrieval and GraphRAG 24
2.4 Machine Learning for the Analytics of Supply Chains 25
2.4.1 Demand Forecasting 25
2.4.2 Anomaly Detection 25
2.4.3 Supplier Risk Assessment 26
2.5 Text-to-SQL and Natural Language Database Interfaces 27
2.5.1 Evolution of Text-to-SQL Research 27
2.5.2 Schema Linking for Large Schemas 28
2.5.3 Safety in Text-to-SQL Systems 28

2.6 Trust, Safety, and Scalability in AI Systems 29
2.6.1 AI Safety and Trustworthiness 29
2.6.2 Hallucination Mitigation 30
2.6.3 Prompt Injection and Security 30
2.6.4 Explainability and Decision Traces 31
2.6.5 Scalability Considerations 31
2.7 Comparative Analysis of Related Works 32
2.8 Positioning and Research Gap 34

3 Research Methodology 37
3.1 Research Approach: Design Science Research 37
3.2 Application of the DSR Process Model 38
3.3 Case Study Context: Bangladeshi Industrial Supply Chain 39
3.3.1 Contextualizing Bangladesh for the Reader 39
3.3.2 Rationale for the Bangladeshi Context 40
3.3.3 Partner Organizations 41
3.4 Data and Knowledge Base Construction Strategy 42
3.4.1 Structured Data: Anonymized Multi-Organization Dataset 42
3.4.2 Unstructured Data: Domain Knowledge Base 45
3.5 Evaluation Protocol 46
3.5.1 Benchmark Query Design 46
3.5.2 Evaluation Metrics 47
3.5.3 Ablation Study Configurations 48
3.5.4 Scalability Evaluation Design 49
3.6 Tools and Technologies 49
3.7 Threats to Validity and Mitigations 50
3.8 Ethical Considerations 52
3.9 Chapter Summary 54

4 Framework Design and Architecture 55
4.1 Design Principles 55

4.2 High-Level Architecture 56
4.3 Schema Linking and Knowledge Graph 58
4.3.1 Schema Indexer 58
4.3.2 Linker and Navigator 58
4.3.3 Knowledge Graph 59
4.4 RAG Pipeline for Domain Knowledge 59
4.5 ML Analytics 60
4.6 Multi-Agent Orchestration 61
4.6.1 Orchestrator 61
4.6.2 Domain Specialists 62
4.6.3 Reasoning Pattern 62
4.7 Text-to-SQL Engine 62
4.8 Trust and Safety Layer 63
4.9 Explainability Subsystem 64
4.10 Scalability Design 65
4.11 External Interfaces 65
4.12 Chapter Summary 66

5 Implementation 67
5.1 Development Environment and Project Organization 67
5.2 Database Schema Implementation 68
5.3 Schema Linking and Knowledge Graph 69
5.4 Knowledge Base Construction and Hybrid Retrieval 70
5.5 Machine Learning Module 71
5.6 Text-to-SQL Engine 72
5.7 Trust and Safety Layer 72
5.8 Multi-Agent Orchestration 73
5.9 External Interfaces 74
5.10 Containerized Deployment 75
5.11 Testing Infrastructure 76
5.12 Chapter Summary 77

6 Evaluation and Results 78
6.1 Evaluation Setup 78
6.2 SQL Generation Accuracy 79
6.3 RAG Retrieval Quality 80
6.4 Latency Performance 81
6.5 Machine Learning Model Performance 83
6.6 Cost Profile 84
6.7 Limitations of the Present Evaluation 84
6.8 Summary 85

7 Discussion 87
7.1 Answers to the Research Questions 87
7.1.1 RQ1: Architecture for Autonomous and Safe Multi-Capability Integration 87
7.1.2 RQ2: The Role of Retrieval-Augmented Generation 88
7.1.3 RQ3: Necessary and Effective Trust and Safety Mechanisms 89
7.1.4 RQ4: Scalability Characteristics 90
7.2 Theoretical Contributions 91
7.3 Practical Implications 92
7.4 Limitations 93
7.5 Ethical Considerations Beyond Data Handling 95
7.6 Chapter Summary 96

8 Conclusion and Future Work 97
8.1 Summary of the Thesis 97
8.2 Future Work 99
8.3 Closing Remarks 100

Bibliography 102

Appendices 109
Appendix A. Selected Benchmark Queries 109

Appendix B. Frontend Interface Screenshots 110

1 Introduction

1.1 Background and Motivation

The industrial supply chain environment across the world has been radically changed. In the last twenty years, with the interplay between digitalization and globalization, it was the convergence of digitalization and globalization that led to this and growing consumer demands. Contemporary supply chains are not just linear chains of procurements, production and distribution anymore, they have become complex, global networks across different geographies, regulation, and technological ecosystems (Christopher, 2016). This has compounded the need in higher level analytics capable of high-volume heterogeneous processing data, determine latent patterns, and aid timely decision making at all levels of the supply chain.

Traditionally, the supply chain management was based on enterprise resource planning (ERP) systems, relational databases, which are mostly SQL-oriented, to store and manipulate transaction data involving orders, inventory, logistics, suppliers and production schedules (Stadtler, 2015). Traditionally, the analytical layer that is developed on these databases has assumed the role of the following type of business intelligence (BI) dashboards and non-interactive reporting systems that involve much human skill to query, visualize, and take action (Chae, Olson, & Sheu, 2014). As the volume, velocity as well as the diversity of supply chain data have increased exponentially - powered by via the Internet of Things (IoT), sensor networks, and digital procurement processes - these conventional methods have grown too little to use to draw action real-time able insights (Ivanov, Dolgui, & Sokolov, 2019).

Machine learning (ML) is coming in as a strong complement to conventional analytics, enabling predictive and prescriptive services in a wide variety of supply chain operations. ML models have been effectively used in demand forecasting (Carbonneau, Laframboise, & Bhardwaj, 2008), inventory optimization (Boute, Gijsbrechts, van Jaarsveld, & Vanvuchelen, 2022), supplier risk assessment (Bode & Wagner, 2015), anomaly detection in the logistics (Chauhan & Vig, 2015). Nonetheless, the implementation of ML models

alone exhibits the serious flaws in the industrial context; they frequently act as black boxes not integrated into the larger decision-making picture, they need to be engineered manually, and retrained periodically, and they do not give explanations that are understandable by domain knowledge and organizational decisions (Baryannis, Validi, Dani, & Antoniou, 2019).

The new large language models (LLMs) and AI agents have opened up a whole new future shift in a new paradigm of dealing with data and analytical systems. The modern definition of an AI agent is an independent program that uses an LLM to reason core and is capable of perceiving his or her environment, multi-step action planning, externals invocation and produce natural language (L. Wang et al., 2024). As opposed to conservative ML pipelines or AI agents chatbot interfaces have the ability to reason dynamically - they can break down complex questions into sub-tasks, determine the tools or data they want to consult, make SQL calls to databases, call ML models when predicting, synthesize response by extracting coherent, contextually-grounded answers (Yao et al., 2023). This agentic paradigm is an important step in the direction of not passively analytic but active, autonomous decision support.

One fairly significant advancement in this area is Retrieval-Augmented Generation. A method, (RAG), that combats one of the most key weakness of LLMs: their propensity to hallucinate or to produce plausible but factually false information (Lewis et al., 2020). RAG is a method that expands the generative ability of an LLM by recalling the appropriate information in a knowledge base outside the system - e.g. technical manuals, procurement policies, or quality standards, or a report of the past–and pouring in this retrieved context into the process of generation. Where decisions should be made in industrial supply chain situations, they need to be made to be sound which is based on verifiable data and adheres to regulations and standards in the field, RAG offers a system of anchoring AI-induced insights in authoritative knowledge, and not only parametric memory (Y. Gao et al., 2024).

Nevertheless, remaining on top of all these tech trends - AI agents, ML analytics, and RAG. However, the integration of time-series or SQL databases into a single integrated,

reliable and scalable infrastructure is still an open problem. Previous studies typically explore these components in isolation: extensive work on text-to-SQL systems (Li et al., 2024), on RAG architectures (Y. Gao et al., 2024), on ML for supply chain forecasting, completed by (Carbonneau et al., 2008) and finally few works address LLM-based agent frameworks (L. Wang et al., 2024). We have nonetheless not seen a consensus framework that brings these elements together in a way specifically designed for requirements that go hand in hand with industrial supply chain analytics – requirements which includes not just functional correctness and predictive accuracy, but also safety, auditability, explainability and scaling with enterprise grade data volumes and simultaneous users.

One other gap, seldom discussed in the literature, relates to the real-life implications of enterprise-scale database schemas. These benchmarks consist of small schema that have only a few tables eighteen and only includes a data set with more than ten tables (Yu et al., 2018). Whereas a real industrial supply chain database can regularly have dozens of related tables across purchase, stock-out, need, PS: logistics, production, quality and finance; Naive approaches that inject the full database schema into the LLM context window become unaffordable and inaccurate (Li et al., 2024) at this scale. An industrially viable frame the work must include Intelligent schema navigation, domain aware query routing and decomposition of complex cross-domain questions into manageable sub-queries.

Enhanced sustainability in industrial organizations requires more than sheer gusto, and this thesis is(lightly) motivated by that understanding federated AI elements; what they're in need of is a unified, agent-based analytics framework that can automatically query realistic enterprise scale SQL databases, and use ML models to predict by grounding its reasoning with domain knowledge using RAG and do all of this on top of a governance framework that still creates trust/safety.

1.2 Problem Statement

The industrial supply chains produce massive amount of structured data in relational DBs, stored in data bases like PostgreSQL, which would include all the information about trans-

actions, inventory movement, supplier details, production logs, shipment tracking, all structured in ordered tables. At the same time, there is abundant unstructured domain knowledge (procurement policies, quality standards (ISO 9001, ISO 28000), supplier evaluation reports, technical specifications, operating instructions etc.) Now, to get insights out of these heterogeneous sources, the workflow looks fragmented: Analysts need to write SQL queries, then consult separately ML models for predictions, then look up policy documents to get context to what they saw, and/or to check compliance —a slow, erroneous, biased process (which non-techie decision-makers would not be able to use) (Trkman, McCormack, De Oliveira, & Ladeira, 2010).

Existing AI and LLM-based solutions only solve parts of this problem. Unlike analytical thought and domain grounding, text-to-SQL systems are able to query the database with natural language (Li et al., 2024). Even though conversational AI tools can respond to the questions, it regularly hallucinates when working with specialized industrial data (Ji et al., 2023). While the predictions of standalone ML models can be used, they cannot explain the prediction outputs in the context of the organizational policies or constraints of the supply chain (Baryannis et al., 2019). While RAG systems add an ability for factual grounding, they are mostly restricted to question-answering over documents and are unable to query live databases or invoke analytical models (Y. Gao et al., 2024).

The primary issue this thesis intends to tackle is lack of a holistic AI agent that can autonomously and safely orchestrate across all these capabilities—SQL-like data queries over realistically large schemas, ML based analytics and RAG based knowledge integration—into trustworthy explainable and scalable supply chain intelligence. Furthermore, this problem is exacerbated in the contexts of emerging economies; the global procurement operations, stand out here, with numerous international trade procedures (e.g., letters of credit, customs compliance, port operations), many levels of suppliers, currency risk, and regulations, which differ greatly from those assumed in most Western-centric research.

Second, we have zero exploration in terms of the safety and trust dimensions of such a system: With an AI agent generating SQL from scratch against production databases, with

an ML prediction that may inform a procurement decision, triggering natural language recommendations, all while the risk of acting incorrectly, acting in a biased or unauthorized manner (Amodei et al., 2016) is so high. Releasing agents like these in the wild, in an industrial scenario without hard guardrails, ways to continuously validate their actions and mechanisms to build an audit trail of the outputs after deployment is dangerous.

1.3 Research Objectives

This thesis is centered around designing, implementing, and evaluating a reliable and scalable AI agent framework capable of SQL-based data interaction, machine learning analytics, and data retrieval-augmented generation specifically for use cases in industry supply chains. This overall goal is disaggregated further into these individual objectives:

Objective 1: To build a modular multi-agent AI framework to internally orchestrate the downloadable availability of SQL query generations and execution plans against enterprise-grade PostgreSQL databases, ML model invocations for predictive and diagnostic analytics, and RAG-based retrieval from specific domain knowledge knowledge bases in a single reason and action loop. The framework should be able to link schemas such that thirty or more tables across multiple business domains can be related.

Objective 2: To create a purposed trust and safety span over SQL query validation and sandboxing, output verification and hallucination mitigation, stand against prompt injection, confidence scoring, as well as explainability via decision traces and audit logging, such that that agent do keep itself intact inside the limits of safe operation boundaries appropriate for industrial setups.

Objective 3: To assess the functional performance of a framework in terms of SQL generation accuracy, ML prediction quality, RAG retrieval relevance, and overall answer correctness through a supply chain dataset that represents a Bangladeshi industrial context and applied analytical scenarios that encompass factual, predictive, diagnostic, policy-grounded, and multi-hop reasoning queries.

Objective 4: To test the scaling properties of the framework in the context of multiple

data volumes, various query complexity levels and usage patterns, and to isolate architectural bottlenecks and validate design for large-scale enterprise deployment.

1.4 Research Questions

To direct the manifestation of the inquiry and ensure that the research aim is systematically answered the following research questions are created:

RQ1: What is the design of an AI agent architecture that autonomously and safely interacts with enterprise scale SQL-based industrial databases, designed to incorporate machine learning analytics, an approach for retrieving relevant frames of domain knowledge to aid in the decision-making process in supply chains?

RQ2: In the industrial supply chain context, compared to agent configurations without knowledge augmentation, how does retrieval-augmented generation contribute to the factuality, domain grounding, and explainability of AI agent outputs?

RQ3: What are trust and safety mechanisms needed and effective for governing AI agent executing SQL queries autonomously, invoking ML models, and producing analytical suggestions in an industrial context?

RQ4: What is the scalability of the proposed framework with the growth of data volume, the complexity of query requests, and the number of concurrent user loads within the dedicated resource managers in original single nodes, and its principal architecture components that affect the scalability?

1.5 Scope and Limitations

The remainder of the thesis presents design, a prototypical implementation and evaluation of the proposed framework. The scope is limited to the following:

Domain scope: The construct is intended to spans seven functional domains of industrial supply chain management including procurement, inventory, demand, logistics, produc-

tion, quality and finance. Every domain is represented by its own set of tables in your database and one dedicated specialist agent. Among this general scope of architecture, we choose three use cases for deep demonstration and assessment: namely, demand forecasting (Predictive analytics), inventory anomaly detection (diagnostic analytics) and supplier risk assessment (prescriptive analytics). This use case selection is justified as each one covers one of the main categories of analytical tasks in supply chain management (Souza, 2014), and for each task the framework capabilities are exercised in a different way.

Case study context: The framework is illustrated on a set of artificial but plausible data aligned to a Bangladeshi manufacturing /distribution business with operations in sectors including Ready-Made Garments, fast-moving consumer goods, pharmaceutical inputs and electronics assembly. Context in Bangladesh but with a domain-specific flavor — Letter of Credit based imports, operations at Chattogram Port, foreign trade procedures within bonded warehouses using BSTI quality certifications, multi-tier supplier networks from China, India, Vietnam, the European Union and the United States — force a stress test on the RAG component of the framework to assimilate localized domain knowledge. This option offers an understudied, multifunctional, and economically important opportunity to validate a context that has thus far been largely Western-focused in the supply chain analytics literature.

Database environment: The implementation uses PostgreSQL 16 as the relational database management system with the vector similarity search capabilities provided by the pg vector extension (Kane, 2024; The PostgreSQL Global Development Group, 2024). We chose PostgreSQL because it is open-source, widely used in both academia and industry, and natively supports hybrid relational and vector workloads in a single engine. As the results could be relevant for all similar SQL based systems, the cross-database portability is not empirically verified in this thesis.

LLM backbone: The architecture has an architectural design that should be model-agnostic but implements and evaluates OpenAI GPT-4o (OpenAI, 2024). The capabilities of underlying LLM also affect the performance of the framework and it can be varied by

different versions or providers of the model.

Knowledge base: For the RAG component, a knowledge base is built from synthetic policy documents that are based on realistic Bangladeshi Supply Chain Operational Guidelines, Quality Standards and Risk Management Frameworks that were initially defined for demonstration purposes. And the empirics of integrating with an organization's proprietary knowledge asset to form a production deployment led to new data governance challenges that are discussed but left unresolved in this work.

Evaluation strategy: Scalability evaluation is performed through empirical benchmarking; it is not deployed in a production context. The benchmarks are optimized to represent real world workloads, but fail to fully reproduce the depth of a production deployment (or at least do so at greater cost than maintaining the pipeline); namely, variability and stochasticity due to network latency, solutions running over different systems in concert and organizational workflow.

Out of scope: This thesis does not cover the fine-tuning or custom training of LLMs. The framework uses pre-trained models with 3rd party models through rest-style APIs with RAG and prompt engineering instead of tuning model parameters. Commercial ERP systems integration (SAP, Oracle, Microsoft Dynamics) must be considered out-of-scope; the framework is illustrated in action working with a standalone PostgreSQL instance.

1.6 Thesis Structure

The rest of this thesis is structured as follows.

Literature Review: The purpose of this section (Chapter 2) is to highlight the theoretical and technical aspects of this research. Industrial supply chain analytics, AI agent architectures, retrieval-augmented generation, machine learning tools for use in supply chain applications, text-to-SQL systems, including schema linking, multi-agent frameworks, how to incorporate knowledge graphs, explainability, agent interoperability standards, trust, safety, and scalability in AI systems. It ends with a discussion about where this thesis fits in with other researches.

Chapter 3: Research Methodology. Outlines the methodological approach that underpins this research. The research methodology is established as Design Science Research (DSR), and this chapter depicts the research design, the design of the Bangladeshi supply chain case study, the data collection strategies, evaluation criteria, and the tools and technologies used.

Chapter 4: Framework Design and Architecture. The core design contribution is in Chapter 4. It provides the high-level architecture and the design of each component of the module: the schema linking and knowledge graph components, the multi-agent orchestration engine with domain specialists, the interaction with the SQL module, the RAG pipeline, the ML analytics module, the trust and safety layer, the explainability module, and the scalability design.

Chapter 5: Implementation. The technical realization of the framework is discussed, covering thirty-seven tables in seven business domains to achieve the database schema design, through knowledge base construction, the construction of the agent pipeline, the training and integration of ML models, implementing safety mechanisms, containerized deployment, and Model Context Protocol integration.

Chapter 6: Evaluation and Results. Presents the results of functional, trust, safety, and scalability evaluations. It presents comparative analyses and ablation studies illustrating how each framework component contributes.

Chapter 7: Discussion. Interprets the results with respect to the research questions, discusses the theoretical and practical implications, identifies the limitations of the study, together with ethical issues including those relating specifically to the emerging economy context.

Chapter 8: Conclusion and Future Work. Summarizes contributions and future work directions.

2 Literature Review

This chapter gives an extensive overview of the theories and previous studies that are relevant to this thesis. Highlights include foundational work in industrial supply chain analytics and relevant AI agent architectures, retrieving-augmented generation mechanics, machine learning methodologies applicable to supply chain solutions, text-to-SQL systems (schematic linking approaches applied in large schema domains), knowledge graph fusion and multimodal embedding, multi-agent specialization, agent interoperability standards, explanation, ethics, and core AI system principles of trust, safety, and scalability. This chapter ends with a comparison of existing works and a clear statement of the research gap that this thesis tries to fill.

2.1 Industrial Supply Chain Analytics

Supply chain analytics has been described as systematic use of data-based analytical methods to improve decisions in procurement, production, logistics, and distribution processes (Souza, 2014). A typical definition of the evolution of supply chain analytics is through four maturity levels: descriptive analytics (what happened), diagnostic analytics (why it happened), predictive analytics (what will happen), and prescriptive analytics (what should be done) (Trkman et al., 2010). Though the majority of industrial organizations has developed some proficiency with respect to descriptive and diagnostic analytics via their enterprise resource planning (ERP) systems and business intelligence tools, the shift towards predictive and prescriptive capabilities is rather daunting (Chae et al., 2014). Industry 4.0 has rapidly increased the amount of data available for Supply Chain Analytics as we live in the midst of the industrialization of systems through digitalization. With the rise of integrated Internet of Things (IoT) sensors, cyber-physical systems, and cloud computing, data is now collected in real time at production floors, warehouses, fleets, and supplier networks (Ivanov et al., 2019). Mainly, this data is stored in relational database management systems (RDBMS) like PostgreSQL, MySQL, and Microsoft SQL Server, which are the core transactional systems of ERP systems like SAP, Oracle, and Microsoft Dynamics (Stadtler, 2015). While the structured query language (SQL) remains the first interface

to query and retrieve this data, this creates a heavy reliance on technical expertise as it makes analytical data less accessible to non-technical supply chain professionals (Trkman et al., 2010).

Many scholars have pointed out the disparity between the amount of data available on supply chains and the capacity of organizations to leverage that data to make informed decisions. As Waller and Fawcett (2013) pointed out, the greatest supply chain analytics bottleneck is not data availability but a lack of analytical talent and tools to convert data into decisions. Sanders (2016) emphasized that effective supply chain analytics requires not only quantitative methods but also domain knowledge and contextual understanding. In an even more recent work, Nguyen and Pasupa (2024) conducted a survey of the use of AI across supply chain tasks and found that there was a lack of systems that could autonomously connect structured data recorded in databases with unstructured domain knowledge, stored in documents and policies.

Emerging economies feature unique supply chain characteristics that add even more complexity and remain vastly under explored in mainstream analytics research. For example, Bangladesh is the second largest supplier in ready-made garments, with supply chains that rely on Letter of Credit based imports, multi-tier subcontracting, bonded warehouse operations and regulatory frameworks steered by Bangladesh Bank, National Board of Revenue and Bangladesh Standards and Testing Institution (Islam, Karia, Fauzi, & Soliman, 2017). These operational realities, in turn, provide rise to unique analytical needs such as customs compliance rules, foreign exchange regulations, and sector-specific audit requirements (BSCI, SEDEX) that the standard supply chain analytics frameworks do not support. To bridge this gap, the present work selects a Bangladeshi case study as it can test the proposed framework in contexts different in meaningful ways from that of the Western industrial world.

2.2 AI Agents and Agentic Systems

2.2.1 Foundations of AI Agents

The idea of an intelligent agent goes back to the beginning of the artificial intelligence (AI) research. Russell and Norvig (2021) define an agent to be anything that observes its environment through sensors and acts upon that environment through actuators. An AI agent is a program that runs in software systems capable of autonomous observation, reasoning, planning and execution of actions in an environment to achieve some predefined objectives (Wooldridge, 2009). Autonomy (operating without the direct intervention of humans), reactivity (responding to changes in the environment), proactivity (taking the initiative towards achieving their goals) and social ability (interacting with other agents or humans) are the properties that characterize agents, as opposed to conventional software (Wooldridge, 2009).

Large language models (LLMs) paradigm shift of the agent paradigm. The modern LLM-based agents utilize a pre-trained language model (which serves as the cognitive core of the agent) powered with the aide of LMs equipped with external tools, memory systems, and planning mechanisms (L. Wang et al., 2024). In contrast to rule-based or reinforcement learning agents that only apply in narrowly defined environments, LLM-based agents can receive natural language instructions, reason across multi-step tasks, produce executables such as code or queries, and share results in a human readable format (Xi et al., 2023).

2.2.2 Agent Architectures and Reasoning Patterns

Various architectural patterns have been mentioned for LLM based agents. Yao et al. (2023) ReAct (Reasoning and Acting) framework mixes reasoning traces with action steps — now the agent plans, executes tools, observes outcomes and repeats. This is especially applicable to the current work as many supplies chain analytical queries involve multi-step reasoning: finding the relevant data source, querying the data, interpreting the results and domain knowledge to do contextualization.

This is the Plan-and-Execute pattern, which decouples high-level planning from step-level execution to yield more structured and auditable agent behavior (Z. Wang et al., 2024). This method involves a planner agent that breaks down a complex task into an ordered list of subtasks, and an executor agent that performs each sub-task using the tools at its disposal. This separation is beneficial for industrial applications that require auditability and interpretability in describing how the agent made its decision.

2.2.3 Multi-Agent Systems and Area of Expertise

Another key architectural direction can be found in the multi-agent systems. Frameworks like AutoGen (Wu et al., 2023), CrewAI, and LangGraph (Chase, 2022) allow multiple specialized agents to work together on a complex task where each has different expertise or access to different tools. Guo et al. (2024) provides a systematic review of the state of the art in LLM-centric multi-agent setups and presents three main collaboration mechanisms identified in the scholarly literature: hierarchical co-ordination (whereby an orchestrator coordinates a bunch of agents), peer-to-peer negotiation (agents negotiate with one another), and role-based specialization (some agents specialize in specific roles). Specifically, it has been shown empirically that specialized agents perform better than generalist agents over tasks requiring deep domain expertise when the task space is heterogeneous (Guo et al., 2024).

Domain specializations in the analytics context seems intuitive as that is how the overall supply chain performance is broken down into procurement, inventory, demand, logistics, production, quality and finance separate functions. These domains differ not only in their common vocabulary, but also in their key performance indicators and the way they analyze the data. Prior work on agent specialization indicates that assigning domain-specific context and few shot examples to specialized agents can lead to higher accuracy than using a single generalist agent (Guo et al., 2024; Talebirad & Nadiri, 2023). With multi-agent architectures comes the added complexity of communication protocols, conflict resolution (e.g., behavior negotiation) and error propagation across agents, which must be addressed (Xi et al., 2023).

2.2.4 Tool Use and Function Calling

Calling external tools (via function calling or tool-use interface) to modern AI agents is one of their ultimate capabilities. Such interaction has enabled agents to go beyond the reach of their parametric knowledge (Schick et al., 2023) and reach into databases, APIs, file systems, computational engines, and other software systems. Schick et al. (2023) shows that adding tools to LMs improves their performance on tasks demanding factual accuracy, arithmetic, and up-to-date knowledge dramatically. In the scope of this thesis, tool use is defined as executing SQL queries over a PostgreSQL database, invoking ML models for predictions, and querying a vector database to retrieve knowledge from it.

The LangChain framework (Chase, 2022) and LlamaIndex (J. Liu, 2022) have become the default open-source libraries for creating tool-augmented AI agents. LangChain abstracts of chains (sequential invocations of tools), agents (dynamically choose what tool to use based on some chain that gets to reason), and memory (how do you persist instruction in conversation and context). LangGraph — a LangChain extension — for building stateful, graph-based agent workflows including cycles, branches, and conditional logic, all the requisite for complex analytical pipelines (Chase, 2022).

2.2.5 Agent Interoperability and the Model Context Protocol

One of the newer developments in agent ecosystem has been the emergence of standard interoperability protocols, which are decoupling agent applications from the tools and data sources that they are consuming. Anthropic (2024) proposed a formal Model Context Protocol (MCP), an open protocol over JSON-RPC 2.0 for any compliant LLM applications to discover and invoke tools exposed by MCP-compatible servers. And by offering a uniform interface, MCP tackles the combinatorial integration challenge, where modeling environments and data sources would previously have required custom connectors for every potential pairing. MCP provides a principled way for exposing database access, knowledge retrieval, and domain tools for downstream clients (desktop assistants, development environments, bespoke applications) without recreating the implementation effort, for enterprise analytical systems. The current thesis added MCP as another in-

terface layer to consume the proposed framework from different heterogeneous client environments.

2.3 Retrieval-Augmented Generation

2.3.1 Foundations of RAG

This approach, known as Retrieval-Augmented Generation (RAG), was proposed by Lewis et al. (2020) as a means of leveraging the generative power of LMs with the accuracy of information retrieval systems. The key idea of RAG is, although language models are trained on a lot of parametric knowledge, they hallucinate at scale, generating text that sounds plausible but lacks factual support — especially when it is specialized, domain-specific, or needs to be up-to-date knowledge (Ji et al., 2023). The RAG combination of models reduces often observed hallucinated results by retrieving relevant documents from an external knowledge base and incorporating that knowledge in generating the output (Lewis et al., 2020).

The RAG generally comes with three components as shown in Table 1, indexing, retrieval and generation (Y. Gao et al., 2024). When the system is indexing, we break up domain documents, encode them with an embedding model in dense vector representation, and store them in some vector database. At retrieval time, an embedding of a user query is also created, and it is input to approximate nearest neighbor search to find the most semantically similar document chunks. At generation stage, the LLM takes the concatenation of the retrieved chunks (added to the query as context memories) and outputs a response (both based on its parametric knowledge and the retrieved chunks).

2.3.2 Advanced RAG Techniques

Many modifications of the natural RAG paradigm have been proposed to overcome its weaknesses. Y. Gao et al. (2024) details a full taxonomy of naive RAG, advanced RAG, and modular RAG. Recent improvements in the field of RAG include query rewriting or expansion to improve the quality of retrievals, hybrid search combining dense vector retrieval

with sparse keyword-based retrieval such as BM25, cross-encoder reranking of retrieved documents, and an iterative retrieval approach where the agent could perform multiple rounds of retrievals based on the intermediate reasoning (Y. Gao et al., 2024). The most recent method is Self-RAG (Asai, Wu, Wang, Sil, & Hajishirzi, 2024), which adds a self-reflect mechanism where the model determines whether it should not use retrieval, determines the relevance of retrieved passages, and criticizes its own generation for factual support.

The proposal of Corrective RAG (CRAG) (Yan, Gu, Zhu, & Ling, 2024) includes a retrieval evaluator that automatically activates the web search fallback if the retrieved documents are insufficient. These sophisticated patterns are increasingly relevant to industrial supply chain applications since even small hallucination errors (e.g., recommending the wrong inventory reorder point or misidentifying supplier risks) can translate into high costs, safety, and disruptions.

Cross-encoder reranking has recently become an important refinement step in state-of-the-art RAG pipelines. Models such as those from the BGE family (Xiao et al., 2024) takes a query and candidate document as paired inputs and outputs a relevance score by using full attention on both inputs, yielding significantly better relevance judgments than the bi-encoder cosine similarity used during initial retrieval. So, an advanced pipeline usually would fetch a wider candidate space with a fast bi-encoder similarity and then rerank the top k candidates using a cross-encoder and then doing context injection.

2.3.3 RAG for Structured and Semi-Structured Data

There is an emerging line of work that is investigating the use of RAG principles beyond document understanding for other types of structured data in databases. Balaguer et al. (2024) demonstrate that RAG techniques can be used to produce SQL queries by retrieving contextual information about the relevant schema, query examples, and documentation as context for the LLM. Schema-enhanced generation, a training that aims to create a bridge between natural language interfaces and database systems — a key functionality sought by the system defined in this thesis. The hierarchical architecture formed

by combining document-based RAG for segment-level domain knowledge and schema-based RAG for database interaction has been largely unexplored in the context of supply chain analytics literature.

2.3.4 Graph-Based Retrieval and GraphRAG

One evolution of the RAG paradigm worth mentioning is the inclusion of knowledge graphs in the retrieval. Edge et al. (2024) introduced GraphRAG, where LLM retrieves entities and relations from a document corpus to build a knowledge graph and traverse the graph to retrieve globally relevant subgraphs rather than locally similar text chunks. GraphRAG and its related work have been demonstrated to exceed vector-only RAG on queries necessitating multi-hop reasoning or thematic synthesis across a corpus (Edge et al., 2024; Peng et al., 2024). Peng et al. (2024) give a comprehensive overview of various graph-based retrieval approaches, categorizing them into three classes: knowledge-graph-retrieval, when a graph augments a standard RAG pipeline; graph-native retrieval, where traversal is the main retrieval mechanism; and hybrid methods combining both.

Knowledge graphs are particularly useful for industrial supply chain analytics in part due to the relational nature of supply chain data: products are sourced from suppliers, stored in warehouses, shipped by carriers and consumed by customers. Having both a graph representation of the database schema and the domain knowledge allows the agent to perform multi-hop reasoning, such as identifying which suppliers are associated with quality incidents that affect specific product lines. A framework to help automatically discover ontology patterns from domain documents; backbone and Table Patterns; a lightweight knowledge graph generated with only foreign key relationships and enriched with semantic edges from domain documents; supports both schema navigation and cross-domain reasoning.

2.4 Machine Learning for the Analytics of Supply Chains

2.4.1 Demand Forecasting

Demand forecasting is one of the most researched machine learning applications within supply chain management. Carbonneau et al. (2008) published one of the first comparative studies between traditional statistical approaches and non-statistical methods, which compared the use of neural networks, support vector machines, and recurrent neural networks for demand prediction. They specifically found that ML methods have the best performance in nonlinear demand patterns — particularly driven by promotions, seasonality, and external market factors. More recently, Boute et al. (2022) proposed deep reinforcement learning for joint demand forecasting and inventory control, showing that end-to-end learning methods are competitive against the two-step paradigm of forecasting followed by optimization.

Gradient boosting methods, XGBoost and LightGBM, have gained dominant position on applied demand forecasting due to heterogeneous features and causes and missing data, which are not preprocessed on model levels (Chen & Guestrin, 2016). There also exist time-series specific architectures, like Temporal Fusion Transformers (TFT) combining attention mechanisms and temporal encoding (Lim, Arık, Loeff, & Pfister, 2021), which have also been found to perform extremely well on various datasets. In this thesis scenario, ML analytics module supports gradient boosting and time-series based approaches in which based on the analytical query and accessible data characteristics, the AI agent selects the appropriate model for execution.

2.4.2 Anomaly Detection

Supply chain anomaly detection refers to recognizing abnormalities in supply chain operations, such as unusual patterns in inventories, order volumes, delay in shipments, and production indicators. Long Short-Term Memory (LSTM) networks, which are a type of recurrent neural network, have proven to be quite effective for the anomaly detection from time-series data, and have been commonly used with supply chain monitoring (Chauhan

& Vig, 2015). This method is very useful for outliers' detection in high-dimensional supply chain datasets where Isolation Forest (F. T. Liu, Ting, & Zhou, 2008) is an ensemble method specifically designed for anomaly detection. Normal operation patterns have been learned using both standard or variational autoencoders (An & Cho, 2015), with novelty detection done by flagging deviations from these normal patterns.

Anomaly detection is crucial for industrial supply chains, not just in terms of algorithm performance but also in terms of interpretation and context. An abnormal inventory detected could be part of a planned promotion, a supplier production problem, a seasonal pattern, or a data entry error. This is exactly why RAG-based knowledge retrieval comes in handy: the AI agent can correlate a detected anomaly with the prespecified procurement schedule, communication between suppliers, and any relevant policies to give contextually contextualized rationales instead of just a statistical alert.

2.4.3 Supplier Risk Assessment

The importance of supplier risk management has risen sharply following a series of large-scale supply chain disruptions due to natural disasters, geopolitical events or the COVID-19 pandemic to name but a few (Bode & Wagner, 2015). Supplier risk assessment approaches that relied on ML (e.g., supplier risk score construction) were largely centered on grouping indicators such as financial stability, order fulfillment history, quality measurements, geographical risks, and dependence concentration (Baryannis et al., 2019). Decision tree-type classification models like random forests and gradient boosting machines have been applied to predict supplier failure or a consequential delivery delay by recognizing historic patterns (Baryannis et al., 2019). In procurement decisions, the interpretability of such models is paramount, and SHAP (Shapley Additive Explanations) values (Lundberg & Lee, 2017) represent a principled way to provide individual supplier feature attribution of risk scores in an auditable and explainable fashion to stakeholders.

Cavalcante, Frazzon, Forcellini, and Ivanov (2019) developed a supervised approach in which structured supplier data are integrated with text-based risk signals (extracted from news articles and financial reports) and provided evidence for the benefits of heteroge-

neous data combination in risk assessment. This is in line with how the AI agent will generate a holistic risk assessment by utilizing structured data from PostgreSQL (Delivery Performance, Order History) and unstructured knowledge from the RAG pipeline (Geopolitical risk reports, Supplier Auditing findings).

2.5 Text-to-SQL and Natural Language Database Interfaces

2.5.1 Evolution of Text-to-SQL Research

Translating natural language questions to executable SQL queries has been a challenging problem in natural language processing and database for decades. Earlier methods depended on brittle rule-based parsing and template matching restricted to specific database schemas (Androutsopoulos, Ritchie, & Thanisch, 1995). Neural sequence-to-sequence models were a big step forward, applying the power of data-driven learning to the mapping between natural language and SQL (Zhong, Xiong, & Socher, 2017).

On the one hand, the large-scale benchmarks including Spider (Yu et al., 2018) and BIRD (Li et al., 2024) released in recent years have promoted the rapid progress of text-to-SQL research. For the Spider benchmark, a dataset that requires generalization over previously unseen database schemas, execution accuracy became the de facto evaluation metric. The most recent BIRD benchmark featured realistic problems present in real-world databases such as noisy values, deep schema relationships, and the requirement of external knowledge to interpret domain-specific terms accurately (Li et al., 2024).

These benchmarks form the standard on which LLM-based approaches have to perform. Pourreza and Rafiei (2024) proposed DIN-SQL, a decomposed in-context learning approach that brings the text-to-SQL task down to the schema linking, query classification and SQL generation sub-problems. D. Gao et al. (2024) introduced DAIL-SQL, taking an approach of systematically studying new prompt engineering strategies for text-to-SQL, showing that through systematically choosing the demonstration examples and schema representations through experimentation had significant effects on performance. These works also directly motivate the design of the SQL interaction module integrated in this

thesis: schema-aware prompting, query decomposition, and example-based guidance are used to maximize the accuracy of generation.

2.5.2 Schema Linking for Large Schemas

One of the most important sub-problems in text-to-SQL is schema linking, which involves mapping the useful tables and columns in a schema to natural language queries. Benchmarks like Spider often contain databases with fewer than 10 tables, whereas realistic industrial databases usually have dozens or hundreds of tables spanning multiple business domains. Simply putting the entire schema in the LLM prompt is costly and counterproductive in terms of accuracy because the model has to filter useful from the irrelevant context (Pourreza & Rafiei, 2024).

Several strategies have been explored in the context of schema linking research. Lei et al. (2020) demonstrated that explicit schema-linking modules greatly increase parsing accuracy on complex schemas. Recent work has extended RAG-style retrieval modelling directly to schema information: instead of treating table and column descriptions as an ancillary resource which influences a SQL generator indirectly, embedding and indexing the schema information and retrieving the most relevant subset at query time provides a set of focused schema context to the SQL generator (Li et al., 2024; Pourreza & Rafiei, 2024). The present thesis follows and expands on this paradigm in treating the database schema as an independently retrievable knowledge source, in addition to the document-based RAG used for policy knowledge.

2.5.3 Safety in Text-to-SQL Systems

Safety is a crucial issue in preparing text-to-SQL systems for industry deployment. Due to the possibility of modifying or deleting data, exposing sensitive information, or growing unbounded resource consumption with inefficient generated queries (Li et al., 2024), SQL queries generated automatically must be carefully safeguarded. The trust and safety layer proposed in this thesis mitigates these risks through enforcing read-only database access, analyzing query complexity and execution timeouts, and validating query results. Most

text-to-SQL benchmarks focus only on functional correctness, whereas any real-world application must add a second axis of evaluation, safety: through unsafe query rejection, query cost, and adversarial robustness of input.

2.6 Trust, Safety, and Scalability in AI Systems

2.6.1 AI Safety and Trustworthiness

Autonomous AI systems that are deployed in the industries with the highest stakes require strict focus on safety and reliability. Amodei et al. (2016) described five actual issues in AI safety, namely preventing negative side effects, preventing reward hacking, scalable controls, safe exploration, and distributional robustness. Although their discussion was set in the view of reinforcement learning, these issues apply directly to agents using LLM: An agent with negative side effects is shown by harmful SQL queries, an agent that functions based on user satisfaction at the cost of truth is shown to engage in reward hacking, and an agent that collapses in the presence of unknown database schemas is shown to be distributionally fragile.

Various frameworks have been formulated to establish trustworthy AI. Ethics in Trustworthy AI, which includes seven requirements, has been developed by the European Union: human agency and oversight, technical robustness and safety, privacy and data governance, transparency, diversity and non-discrimination, societal and environmental well-being, and accountability (High-Level Expert Group on Artificial Intelligence, 2019). Most of these principles are formalized in the follow-up EU AI Act (European Parliament and Council of the European Union, 2024) as binding standards that any high-risk AI systems applied to the European Union must comply with, introducing rules regarding risk management, data controls, transparency, human oversight, accuracy, and cybersecurity. Though the current thesis is not conceived as a direct regulation requirement, the safety design of the structure is conscious towards such principles in a manner that the structure may be used as a base on which compliant deployments may be made. Regarding industrial AI agents, Kaur, Uslu, Rittichier, and Durresi (2022) claimed that the functionality of trust is performed in three ways, including competence trust (the system produces

the correct outputs), integrity trust (the machine is provided with specific limits), and benevolence trust (the system is employed in the interest of the user).

2.6.2 Hallucination Mitigation

Hallucination — the most important trust issue of the LLM-based systems — is generally considered to be the factual incorrectness or fabrication or lack of support (Ji et al., 2023). Ji et al. (2023) offers a taxonomy of the intrinsic versus extrinsic hallucination (output cannot be obtained by using the source) in detail. Hallucination in supply chain applications may take the form of falsified inventory levels, wrong supplier names, fictitious product codes or irrelevant analyses.

The most used mitigation strategy to halt hallucination is RAG (Y. Gao et al., 2024), although it does not work effectively individually. Other methods are output grounding verification (compared generated claim with evidence that has been retrieved), confidence calibration (delivering uncertainty estimates along with outputs), chain-of-thought transparency (exposing the agent reasoning steps to be reviewed by humans) and human-in-the-loop escalation towards high-stakes decisions (Tonmoy et al., 2024). The framework given in this thesis introduces a multi-layered approach to hallucination mitigation that uses RAG grounding, SQL results verification, confidence scoring, and audit logging.

2.6.3 Prompt Injection and Security

An early detection of attack is a major security risk that rollout LLM-based systems are prone to. Perez and Ribas (2022) showed that adversarial prompts can exploit the LLMs to disobey their system directions, expose sensitive data, or perform undesired tasks. Using an AI agent, which has access to an SQL database, a successful prompt injection may result in unauthorized access to data, data manipulation or a denial of service using resource-consuming queries. The input sanitization and filtering, the enforcement of the instruction hierarchy (the system-level instructions have the priority over the user inputs), the output filtering, and the use of the sandboxed execution environments are the defense strategies (Greshake et al., 2023). OWASP Foundation (2023) lists prompt injec-

tion as the most common security threat of the LLM systems and suggests a defense-in-depth strategy which embraces input validation, privilege separation, and output monitoring. The security boundary in this thesis includes input validation, sandboxing of SQL query by use of read-only database role and parameter query, resource limitations in executing queries and checking outputs to avoid timely injection.

2.6.4 Explainability and Decision Traces

Explainability is not transparency, but rather a close concept of it. According to Barredo Arrieta et al. (2020), explainable AI represents systems with both explainable features (liberation to make sense of the rationale in a choice) and completeness (the aptitude to characterize the operational of a system correctly). In the case of agentic systems, explanations go beyond feature-level explanations about the individual prediction to include decision traces: structured logs of the reasoning step of the agent, tool calls, data used, and change in confidence, as the agent solves a query (Doshi-Velez & Kim, 2017). These traces are incredibly essential as well as to the trust of the users as well as audit after hoc of regulated industrial settings. The framework that was suggested in this thesis records end-to-end decision traces, and breaks down confidence scores into data, knowledge, and model parts, as well as makes these traces available to humans.

2.6.5 Scalability Considerations

Scalability in AI agent systems has several aspects: data scalability (Processing bigger databases), computational scalability (Serving more inference loads), and user scalability (Serving multiple users concurrently) (Mao, Ye, Chen, Wang, & Zhuo, 2024). The main bottlenecks in agent systems based on LLM include; the latency of an LLM inference, the time of database query execution, and the latency of vector search when selecting a RAG. These common strategies in dealing with these bottlenecks have response caching of requests frequently made, asynchronous execution of long-run ML activities, and pooling and query optimization of database connections, as well as horizontal scaling using containerized microservice applications (Mao et al., 2024). Patel, Sheth, and Auer (2024) particularly investigated scalability of RAG systems, and revealed that hybrid retrieval

strategies (i.e., combining sparse and dense approaches) are more effective in preserving the quality of retrieval as compared to pure dense retrieval as the size of the knowledge base increases. These aspects directly guide the scalability architectural design of the proposed framework.

2.7 Comparative Analysis of Related Works

To explicitly contextualize this thesis in the current literature, Table 1 systematically compares the proposed framework to the more similar works along six major dimensions: AI agent orchestration, SQL database interaction, machine learning analytics, retrieval-augmented generation, trust and safety mechanisms and scalability evaluation. Available literature as shown in Table 1 covers individual aspects of the problem space in great detail, but none are offered that is theoretically a comprehensive framework that can be used across all six dimensions. The original RAG work by Lewis et al. (2020) has set the retrieval-augmented generation paradigm yet failed to integrate the agent-based orchestration, database interaction, and ML analytics. The ReAct framework by Yao et al. (2023) proposed a potent argument-and-acting style to the AI agencies but was shown through the general-purpose benchmarks without reference to industrial data systems and to the contexts of supply chains.

Compared to non-SQL work, text-to-SQL work (including BIRD benchmark, Li et al. 2024) and DIN-SQL (Pourreza & Rafiei, 2024) has taken large strides in natural language database interaction, but generates SQL generation as a one-off task without further analysis as part of a wider analysis pipeline that would include machine learning prediction or knowledge retrieval. The DIN-SQL system also involves partial schema retrieval similar to RAG, however, it is constrained to schema data as opposed to integrating domain knowledge.

Multi-agent systems like AutoGen (Wu et al., 2023) offer more adaptable agent information frameworks and assist partial database interaction by code creation, nonetheless, do not concentrate distinctly on SQL safety, supply chain ML analytics, or grounding domain knowledge using RAG. The overall RAG survey of Y. Gao et al. (2024) includes

Table 1. Comparative analysis of the proposed framework against related works..

Work	AI Agent	SQL Interaction	ML Analytics	RAG	Trust & Safety	Scalability
Lewis et al. (2020) — RAG for NLP	No	No	No	Yes	No	No
Yao et al. (2023) — ReAct Agent	Yes	No	No	No	No	No
Li et al. (2024) — BIRD Text-to-SQL	No	Yes	No	No	No	No
Pourreza and Rafiei (2024) — DIN-SQL	No	Yes	No	Partial	No	No
Wu et al. (2023) — AutoGen Multi-Agent	Yes	Partial	Partial	No	No	No
Y. Gao et al. (2024) — RAG Survey	No	No	No	Yes	Partial	Partial
Baryannis et al. (2019) — ML for SC Risk	No	No	Yes	No	No	No
Nguyen and Pasupa (2024) — AI in SCM Survey	No	Partial	Yes	No	No	Partial
Asai et al. (2024) — Self-RAG	No	No	No	Yes	Partial	No
Mao et al. (2024)	Yes	No	No	No	Partial	Yes
This Thesis	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

aspects of trust, including mitigation of hallucinations, but only touches on the consideration of scalability, but does not mention agent orchestration, database communication, or domain-specific ML models. In the supply chain sector, Baryannis et al. (2019) offer a very convenient overview of ML usage in the area of risk management but omit autonomous agent systems, natural language interface, and knowledge reinforcement. The more recent survey of Nguyen and Pasupa (2024) includes AI uses in various supply chain functions and suggests that more integrated, intelligent systems are necessary but does not offer an actual framework nor conduct an evaluation.

Trust, to some extent, can be addressed by Self-RAG (Asai et al., 2024) and other advanced RAG approaches, but it is at the generation level, and not at the SQL safety, ML model governance, or end-to-end system auditability. GraphRAG (Edge et al., 2024) presents knowledge-graph-based retrieval which enhances multi-hop reasoning without being interoperable with accessing the database or running ML analytics. Studies on scalable agent structures (Mao et al., 2024) cover the computational effectiveness and simultaneous processing but have not been used in the context of industrial database-based analytical systems. The framework suggested in this thesis is characterized by combining all six dimensions with a single architecture specifically aimed at providing analytics of an industrial supply chain. It integrates multi-agent coordination with domain expertise (based on ReAct and plan-and-execute models), SQL database access with safety nets (building on text-to-SQL research with schema linking, sandboxing, and schema validation), machine learning analytics over supply chain functions (demand planning, anomaly detection, supplier risk), hybrid document and graphical RAG as a domain knowledge integration system (exploiting advanced retrieval methods for industrial documents and structural knowledge).

2.8 Positioning and Research Gap

The review of literature states that there is a considerable and clear gap in the research at the intersection of AI agents, supply chain analytics, and trustful system design. Although separate aspects such as LLM-based agents, RAG, text-to-SQL, schema linking,

knowledge graphs and ML to supply chain have already come of age, the integration of these aspects to create a complete system of industrial supply chain analytics is still unreported. Particularly, the gaps have been revealed as the following:

First, there is not yet a framework that can allow an AI agent to coordinate autonomously across SQL-based data recovery over enterprise-scale schemas, ML-driven analytics, and RAG-based knowledge integration in a single stream of reasoning and acting in response to the application cases of the supply chain. Existing methods model such capabilities as independent systems which demand human coordination between methods, and benchmark schemas as reported in the text-to-SQL literature are significantly smaller than those found in industry.

Second, trust and safety aspects of implementing AI agents with direct database access in industrial settings are extremely under-researched. Although AI safety studies have recognized the threats associated with autonomous systems and text-to-SQL studies have recognized the threats of unverified query creation, no detailed safety architecture that combines prompt injection defense, SQL sandboxing, output grounding verification, explainability and audit logging has been suggested to agent systems that combine database access, ML prediction, and natural language generation on an industrial scale.

Third, the scalability properties of integrated AI agent systems, most notably joint LLM inference, database search, and vector search in conjunction with the execution of ML models, have not been studied systematically with enterprise workloads in the supply chain.

Fourth, there is scarcely any reference to the applicability of such frameworks to the industrial settings of emerging economies which now have their own unique regulatory, operational, and cultural attributes, which have received practically no coverage of the current literature. This translates to a large share of the manufacturing and distribution activity in the world remaining beyond the interest of the present AI-to-supply chain studies.

This thesis closes these gaps by formulating, deploying and assessing a framework in-

corporating all the above parts with a clear consideration of the trust, safety, scalability and applicability of the framework to emerging economies. It is based on the Design Science Research methodology and proven by practical assessment, safety tests, along with scalability testing with a realistic Bangladeshi supply chain case study.

3 Research Methodology

This chapter introduces methodological frames through which the design, development, and evaluation of the proposed AI agent framework in the industrial supply chain analytics will be developed. Section 3.1 defines Design Science Research (DSR) as the general approach and reasons why this approach is chosen. Section 3.2 maps the DSR process to the specify activities that were carried out in this thesis. Section 3.3 presents the case study about the Bangladeshi supply chain which lays the foundation of the evaluation and directs the reader of the case to the Bangladeshi industrial environment. Section 3.4 presents the strategy of the data and knowledge base development, which involves the multi-organization anonymized dataset acquired in partner companies. The evaluation protocol (benchmarks, metrics and ablation settings) is defined in Section 3.5. The tools and technologies used are listed in Section 3.6. Section 3.7 is about threats to validity and the mitigations undertaken. Section 3.8 discusses the ethical framework that they used in the research.

3.1 Research Approach: Design Science Research

Design Science Research (DSR) is chosen as the main paradigm of methodology in this thesis. DSR itself is a proven research method within information systems and engineering fields, which is more focused on the development and testing of new artefacts that solve the problems identified within a specific area of application (Hevner, March, Park, & Ram, 2004; Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). Unlike behavioural research that aims at developing and testing theories that explain or predict phenomena, design science aims at improving human and organisational capabilities by developing new and innovative artefacts (Hevner et al., 2004). There are four types of artefacts in DSR: a construct (vocabulary and symbols), a model (abstractions and representations), a method (algorithms and practices), and an instantiation (implemented and prototype systems) (Hevner et al., 2004; March & Smith, 1995). The proposed framework in this thesis is an artefact of the type of methodology (the architectural pattern of trustworthy, knowledge-integrated supply chain AI agents) deployed as a running prototype system,

thus implementing two out of four artefacts.

DSR has been chosen to alternative methodologies due to a number of reasons. To begin with, the research problem is essentially a design problem: no framework combines the needed capabilities, and the value that the thesis adds is the design and illustration of such a framework. Second, DSR offers a clear language of the construction action (the manner in which the artefact has been constructed) as well as the evaluation action (the manner in which the utility of the artefact has been proved), both of which are also in the heart of the thesis. Third, DSR is now commonly known in information systems research on data analytics, decision support systems and to a growing extent on AI-based systems (vom Brocke, Hevner, & Maedche, 2020), offer methodological precedent and a robust set of guidelines. Hevner et al. (2004) presented seven guidelines that must be met by a well-conducted design science project, which are: (1) design as an artefact, (2) relevance of the problem, (3) design evaluation, (4) research contributions, (5) research rigour, (6) design as a search process, and (7) communication of research. These rules are applied in this thesis as a standard of quality and they are reconsidered in Chapter 7 (Discussion) to evaluate to what degree the work can meet them.

3.2 Application of the DSR Process Model

The thesis is based on the six-activity model of DSR process according to Peffers et al. (2007): (1) problem identification and motivation, (2) definition of the objective of solution, (3) design and development, (4) demonstration, (5) evaluation and (6) communication. Table 2 is a mapping of each activity to the related thesis chapter and concrete outputs created. It is not a especially linear process. Similar to the iterative nature of design research (Peffers et al., 2007; vom Brocke et al., 2020), results of demonstration activities in its initial stages have been used to modify the design.

As a case in point, an initial study on direct full-schema prompting showed that accuracy declined on cross-domain queries and this prompted the incorporation of schema linking as defined in Chapter 4.

Table 2. Mapping of DSR activities to thesis chapters and outputs..

DSR Activity	Chapter	Output
Problem identification and motivation	Chapter 1	Statement of the integration gap between AI agents, ML, RAG, and SQL-based industrial databases; identification of trust, safety, scalability, and emerging economy gaps.
Definition of solution objectives	Chapter 1	Four research objectives and four research questions.
Design and development	Chapter 4 and Chapter 5	Conceptual architecture, module designs, and a containerized prototype implementation covering thirty-seven tables across seven supply chain domains.
Demonstration	Chapter 5 and Chapter 6	End-to-end execution of representative analytical scenarios on anonymised operational data drawn from partner Bangladeshi companies.
Evaluation	Chapter 6	Functional accuracy, trust and safety, scalability benchmarks, and an ablation study isolating the contribution of each framework component.
Communication	This thesis as a whole	Academic thesis, open implementation, and the artefact specification recorded in the project repository.

3.3 Case Study Context: Bangladeshi Industrial Supply Chain

The framework is instantiated and tested on operational data of partner companies in Bangladesh to base the assessment on realistic operational circumstances. Since the University of Vaasa thesis committee might not regularly be in touch with the Bangladeshi industrial environment, this section will start with a brief contextual background followed by the description of the case entities and the reasons as to why this option was chosen.

3.3.1 Contextualizing Bangladesh for the Reader

Bangladesh is a South Asian economy of about 170 million people with a highly export based industrial foundation. A number of structural characteristics determine the way the supply chains in the country are organized and explain its choice as a context of the thesis. To facilitate the reorganization of the global economic structure, first, Bangladesh is the second largest exporter of ready-made garments (RMG), over the last few years its exports have surpassed over 38 billion USD, and the industry has employed more than four million workers (Bangladesh Garment Manufacturers and Exporters Associa-

tion, 2024). Second, the Chattogram Port on the Bay of Bengal handles about 90 percent of the international imports and exports and much of imports are also handled by bonded warehouses, which are customs-controlled warehouses that enable imports to escape duty until they penetrate the local market. Third, the major payment in international trade is Letter of Credit (LC), which is governed by the foreign-exchange related rules issued by the Bangladesh Bank (the central bank) and tax related rules issued by the national Board of Revenue (NBR). Fourth, the Bangladesh Standards and Testing Institution (BSTI), the national standards authority in the country, issues domestic quality certification; on an international basis, RMG exporters are audited through other industry programmes, like BSCI and SEDEX. The Finnish reader might see parallels in SFS (national standards) and the EU customs framework, but the specifics of the process are quite different.

These structural features are of interest to this thesis as they introduce supply chain characteristics — multi-tiers international sourcing, LC-driven import cycles, bonded warehouse movements, multi-regulatory compliance, and currency-risk exposure — none of which occur or are treated very differently in Western European operations. A framework tested exclusively in Western settings may not be said to demonstrate any general industrial extension, and a framework tested on Bangladeshi data will have evidence of its ability to be adapted to in a way that will be useful even to a reader not familiar with the region themselves.

3.3.2 Rationale for the Bangladeshi Context

Although the economic value of Bangladeshi supply chains is completely relevant, the suitability of AI-based supply chain analytics models to this context has been given limited direct consideration in the existing academic literature (which is largely Western European, North American and East Asian-industrial-based). The case of Bangladesh is thus both (i) an opportunity to the framework to pass the stress test of the capability of the framework to absorb localized domain knowledge based on its RAG foundation, and (ii) an effort to add to the existing underrepresented literature on AI in support of supply chains in emerging economies. Also, the direct contact of the thesis author with the con-

tacts of the industry in Bangladesh provided the opportunity to implement the process of data-sharing partnership with a small number of operating companies, which included real operational data with anonymization protocols as discussed in Section 3.4.

3.3.3 Partner Organizations

The two to three partner companies in Bangladesh working in four sectors of industry (service-based supply chain operations, pharmaceutical manufacturing, ready-made garments, and fast-moving consumer goods (FMCG)) provided data and domain knowledge on this thesis. It is not presented in this thesis part of identities of partners according to the data-sharing agreements that are presented in Section 3.8. All the seven functional areas included in the framework procurement, inventory, demand, logistics, production, quality and finance are exercised together by the partners. An idealized figure of the combined partner ecosystem is as follows:

- **Sectors:** Service-based supply chain operations, pharmaceutical manufacturing, RMG, and FMCG.
- **Supplier Network:** Both domestic (Dhaka, Narayanganj, Gazipur, Chattogram) and international (China, India, Vietnam, European Union, United States) suppliers with multi-level relationships as characterizes export-oriented Bangladeshi manufacturing.
- **Warehousing:** Combination of bonded warehouses (for imported raw materials under customs supervision), general warehouses (for domestic finished goods), and cold storage facilities (for pharmaceutical and selected FMCG products).
- **Customers:** Domestic retail and wholesale buyers of FMCG and pharmaceutical products; export buyers in the European Union and North America of RMG.
- **Regulatory Footprint:** Bangladesh customs, foreign-exchange controls under Bangladesh Bank, NBR Value Added Tax, BSTI product certifications, pharmaceutical Good Manufacturing Practice, and international audit regimes ISO 9001, ISO 28000, BSCI, and SEDEX.

- **Currency environment:** Operating currency is Bangladeshi Taka (BDT) with the USD and EUR as the international business currency and the operating currency is periodically revaluated to major reserve currencies.

This blended profile is rich enough to utilize all the seven supply chain domains with being manageable in the scope of my Master's Thesis. Partner identities, volumes and cross-company aggregation information are given in Chapter 6 when the entire dataset is put together.

3.4 Data and Knowledge Base Construction Strategy

One of the core research methodology choices regarding any data driven research is based on where the data is sourced. The thesis has two complementary data sources, one of anonymized real operational data that was donated by Bangladeshi partner companies to populate the PostgreSQL database, and a curated corpus of domain knowledge documents to the RAG component.

3.4.1 Structured Data: Anonymized Multi-Organization Dataset

The data in the structured form that will be inserted into the thirty-seven-table database will be based on the operational information of two to three partner companies based in Bangladesh. It is a methodological decision made intentionally, aimed to offer greater external validity than synthetic generation of data, without pushing academic research on the boundaries of ethical and legal issues related to third-party commercial data. Real, anonymized operational data is a common procedure in applied supply chain analytics literature (Baryannis et al., 2019; Cavalcante et al., 2019), and provides three substantive benefits over purely synthetic data: (i) realistic statistical distribution which reflects true market dynamics, seasonality and supplier behaviour; (ii) real-world data quality attributes (missing values, anomalies, inconsistencies) that post-stress the framework as it would be stressed in deployment; and (iii) genuine multi-tier supplier and customer relationship structures that cannot readily be synthesised.

Data acquisition process. A three-stage systemic process was involved in data acquisition. The preliminary discussions with prospective partner organizations in the first phase helped to define the depth of the research, the intended purpose of the data and the confidentiality measures that would be given. The second phase involved the signing of an informed consent letter (Section 3.8) by each participating organization, detailing the ways the data could be used in this thesis, the time frame of the research involvement and that no commercially identifiable data would be published in a public product. During the third phase, the partner organizations read data in their operational systems and used anonymization transformations (discussed below) before sending them over to the researcher. At the time of writing Chapter 3, data conversion was incomplete; when all the records had been assembled, Chapter 6 reports the volume of data collected by a couple and per table.

Anonymization performed by partners. It is important that the partner organizations did their anonymization themselves, before the release of the data and not by the researcher after the fact. This arrangement safeguards the privacy interests of the partners, and it is ethical in terms of research of industrial data sharing (Elliot, Mackey, O'Hara, & Tudor, 2016). The changes as made by partners that were revealed to the researcher are:

- **Replacement of identities:** Company identities, supplier identities, customer identity, product identity, employee identity, and any personal identities are substituted by opaque identifiers or pseudonyms (e.g., Supplier A017, Product SKU 0421).
- **Numerical perturbation:** Sensitive numerical amounts, such as monetary values, order quantities, prices and salaries have undergone scaled or jittered by unknown random quantities so that relative dependencies within the data set are maintained, without assigning actual values to given transactions.
- **Temporal shifting:** Records have been perturbed by random shifts such that the relative ordering and approximate seasonality of the events are preserved but it is now infeasible to align records with some external reference (e.g., a news story, a known shipment).

- **Removal of columns:** Columns holding free-text employee notes, bank account details, and particular reference numbers on a specific regulatory reference number, as applicable (e.g., LC numbers with identifying bank numbers) were not pseudonymized but have been fully removed where the partner believed there was too much risk of re-identification.

These changes together have the merit that the dataset does not lose its analytical value — the relative performances of suppliers, their demand patterns, inventories, or cross entity relationships are all still representative of actual operations, and no absolute values or entities sensitive to the market are leaked.

Multi-organization aggregation. Since the data is generated by a set of partner organizations working in various industries, names pacing is used so that names can be shared across partners but not with collisions generated. Each record is marked with a partner code (which is also anonymized) in order to allow an analysis to be performed across the integrated data or be confined to the subset of a single partner. This framework further allows cross-partner comparative analyses like comparing the patterns of demand seasonality between the RMG and FMCG settings without the need to release what partner provided what records.

Gap filling. Partner data has been included in all seven functional areas, although the data around Chapter 3 writing at some reference tables (e.g. external carrier rating tables or country-risk indicators) could not be exhaustively sourced by partner. In cases where few reference records are lost in order to maintain a referential integrity, more records are synthesized with realistic values obtained by other means that are publicly available in the industry and are identified in the database metadata. Chapter 6 reports the proportion between synthesized records to those made by partners.

Acknowledged limitations of the anonymization. The anonymization-by-partner methodology has two known limitations that are rolled on into the threats-to-validity debate. First, the researcher has no independent check of the precise parameters of the perturbations implemented, the statistical characteristics of the data are thus close

but not demonstrably similar to the actual working statistics of the partners. Second the anonymized absolute values should not be taken to be economical in themselves; what the framework gauges and what Chapter 6 reports is the behavior of the framework on real distributions, not performance in absolute terms of a particular set of partner organizations.

3.4.2 Unstructured Data: Domain Knowledge Base

The RAG element of the framework needs a body of knowledge documentations in the domain. To support this thesis, ten domain documents have been written to indicate realistic Bangladeshi supply chain operational policy and procedure. These papers are based on three sources: (i) publicly available data provided by the institutions like BSTI, Bangladesh Bank, the NBR, the international standards organizations (ISO, BSCI, SEDEX); (ii) published industry guidance by the organizations such as the Bangladesh Garment Manufacturers and Exporters Association (Bangladesh Garment Manufacturers and Exporters Association, 2024); and (iii) procedural patterns observed in, and shared anecdotally by, the partner organizations during the data acquisition discussions. The internal policy documents of the partners are not reproduced verbatim: the knowledge base documents are written separately to indicate the form of operational policies that are practiced by the Bangladeshi companies, but not the wording itself. The covered documents include the procurement policy, inventory, supplier evaluation, quality standard, ISO compliance, logistic policy, demand planning, risk management, warehouse operation, and returns and claims. Complete specification is given in Chapter 4.

It is an authored method inspired by the RAG evaluation literature, where collections of curated documents are evaluated to examine retrieval accuracy and generation under a level of control (Y. Gao et al., 2024; Lewis et al., 2020), as well as making sure that there is no crossed-coercion of partner confidentiality violated.

3.5 Evaluation Protocol

One key input of a DSR role is the rigour of its evaluation. This part defines the evaluation procedure that will be implemented in Chapter 6, such as the benchmark design, the metrics, ablation configuration among others, and the evaluation tooling.

3.5.1 Benchmark Query Design

A benchmark of sixty curated queries is built to test the framework, which has five query classes with a dozen queries each:

1. Factual questions that aim at retrieving direct data (e.g., What is the total of inventory of stock-keeping unit X in Warehouse 3?). These are mostly interacting with SQL.
2. Predictive queries that call on ML (e.g., What is the predicted demand of product category Y in the next four weeks?). These are used to test the ML analytics module.
3. Diagnostic questions that involve reasoning based on past trends (e.g. why did on-time delivery by supplier Z decrease in the third quarter?). These are multi-tools of coordination.
4. Policy-based queries that involve document retrieval (e.g., Does the sourcing decision proposals on the proposed new sourcing decision meet the single-source supplier rule mentioned in the procurement guideline?). These most notably practice the RAG module.
5. Multi-hop reasoning queries that cut across domains and need to break down queries (e.g., Which warehouse to be restocked first based on present demand prediction, supplier lead time, and availability of bonded warehouse quota?). These put in the orchestration pipeline.

Each query is paired with a gold answer (where verifiable against the dataset), a gold SQL statement (where applicable), a collection of related knowledge base chunks (where

policy grounded and multi-hop queries have these), and a predetermined set of probable confidence. The benchmark is saved in the form of an organized JSON file in `evaluation/benchmark_queries.json`.

3.5.2 Evaluation Metrics

It uses four metric categories which are pegged with one of the four research questions.

Measurements of functional accuracy (RQ1 and RQ2):

- *SQL execution accuracy:* ratio of generated SQL queries that are successfully executed and have the correct results to the gold answer (Li et al., 2024; Yu et al., 2018).
- *SQL exact match:* percent of canonicalization SQL query generated and matching the gold SQL query, which is reported alongside execution accuracy in a diagnostic manner.
- *RAG recall accuracy and recall percentage:* Proportion of retrieved chunks which are relevant (precision at k) and proportion of relevant chunks retrieved (recall at k): both mean reciprocal rank (MRR) is a composite measure.
- *End-to-end answer correctness:* human rated correctness of the whole natural language response, on a binary correct/incorrect scale with partial credit.

Trust and safety measures (RQ3):

- *Hallucination rate:* fraction of the claims of numbers or references to entities in the output which is not substantiated by retrieved evidence.
- *Unsafe query rejection rate:* fraction of adversarial or policy violation SQL queries that validator blocks correctly, compared to a corpus of fifty SQL query patterns curated by OWASP Foundation (2023) and the Hack Prompt corpus (Perez & Ribas, 2022).
- *Citation compliance rate:* percentage of answers including correct citation to SQL queries, documents and version of models.

- *Confidence calibration:* correspondence between self-reported calibration scores and the observed calibration accuracy, and quantified by the expected calibration error.

Scalability metrics (RQ4):

- *Response time:* the end-to-end query response time of the 50th, 95th and 99th percentiles, scaled by query complexity tier.
- *Throughput:* queries per minute when at constant concurrent load of one, ten, fifty and one hundred simulated users the user configuration is decreased to one, ten, fifty and one hundred users per known value.
- *Scaling behavior with data:* latency, accuracy with varied database population of small, medium and large configuration based on stratified sub-sampling of the partner dataset.

Qualitative measures: In addition to quantitative measures, a qualitative evaluation will encompass the readability and perceived utility of results by the framework. This is achieved by a guided expert assessor template that welcomes supply chain and AI experts (with, where possible, representatives of the partner organizations) to rate chosen answers using five-point Likert scales on the spectrum of accuracy, helpfulness, trustworthiness and explainability. The specialized assessment is inductive considering the limitations in the timeline and sample size of the thesis; its task is not to substitute, but to enrich, the quantitative measures.

3.5.3 Ablation Study Configurations

A study of ablation is carried out to isolate the contribution of each piece of the framework and in the process prove the integration claim that supports the thesis. Its seven system settings are benchmarked on the complete framework with the same sixty query set:

- **Full framework (baseline):** everything made operative.

- **No RAG:** knowledge retrieval impaired; the agent uses parametric knowledge.
- **ML out:** ML analysis tools will be switched off; the agent will have to answer predictive queries with SQL aggregations only.
- **No schema linking:** the entire schema is injected into each SQL generation without any restrictions.
- **No query decomposition:** complex queries are passed on to the SQL generator not being decomposed.
- **No multi-agent specialization:** there is one generalist agent instead of seven domain specialists.
- **No safety layer:** output validation, SQL complexities limits, guardrails off (input validation is on, however, due to ethical reasons, when testing).

In each configuration, each of the above-defined four categories of metrics are noted, allowing performance contribution attribution per component.

3.5.4 Scalability Evaluation Design

Scalability testing is performed in the aspects of parallel user load and database size. In the case of concurrent load, a load generator based on Locust is used to execute predefined query distribution in the four user levels as given in the case above and record the latency and throughput measurements. To scale data, stratified sub-samples of the partner data are classified into small, medium and large database configurations and a representative sample of the benchmark queries is re-run on one each. Measurements are conducted in a controlled Docker based environment where the CPU and memory allocation is fixed to make it comparable.

3.6 Tools and Technologies

The framework has been implemented with the help of open-source and popular tools selected based on their maturity, reproducibility, and adherence to the methodological

needs of the thesis. In Table 3, the major tools are listed, more justification of each selection will be given in Chapter 4 and the particulars of implementation will be given in Chapter 5.

3.7 Threats to Validity and Mitigations

In keeping with the suggested practice in DSR and research in empirical software engineering (Runeson & Höst, 2009; Wohlin et al., 2012), threats to the validity of thesis findings are explicitly taken into account on four dimensions: construct, internal, external and conclusion validity.

Construct validity is about whether the metrics that are used are measuring the intended properties. This framework will be assessed on metrics set in the text-to-SQL literature, RAG literature, and AI safety literature, each with an identified limitation. As an example, the accuracy of SQL execution does not allow distinguishing between a query that is correct by chance, and one that is correct by reasoning; this can be ameliorated by the execution accuracy with an exact-match comparison and a human rating of correctness. The rate of hallucination is particularly notoriously hard to measure in open-ended generation; this is alleviated by measuring it in terms of verifiably numerical claims and references to entities, rather than in terms of subjective quality.

The issue of internal validity deals with whether the effects observed can be attributed to the proposed design and not due to confounding factors. The main mitigation is called the ablation study: by parametrically varying one component at a time and keeping other factors constant (database state, query set, random seeds, and LLM version), the influence of each component can be disentangled. Other mitigations are to do each configuration three times and report mean and variance, and lock the version of the LLM during the evaluation window to prevent confounding due to model updates.

The external validity is related to the ability to make generalizations about the general state outside the scope of the evaluation. There are three threats which are applicable here. First, the partner-applied anonymization does not change any relative patterns,

Table 3. Tools and technologies employed in this thesis..

Layer	Tool / Library	Role
Programming language	Python 3.11	Implementation language for all modules.
LLM backbone	OpenAI GPT-4o (OpenAI, 2024)	Reasoning and generation core for all agents; deterministic SQL generation at temperature 0.0.
Agent orchestration	LangGraph (Chase, 2022)	Stateful, graph-based multi-agent workflow engine.
Relational database	PostgreSQL 16 (The PostgreSQL Global Development Group, 2024)	Transactional data store; hosts seven business domains.
Vector storage	pgvector (Kane, 2024)	In-database vector similarity search for RAG.
Embedding model	BAAI/bge-large-en-v1.5 (Xiao et al., 2024)	Dense embeddings for documents and schema.
Reranker	BAAI/bge-reranker-large (Xiao et al., 2024)	Cross-encoder reranking of retrieved chunks.
Knowledge graph	NetworkX (Hagberg, Schult, & Swart, 2008)	In-memory graph for schema and semantic relationships.
ML — forecasting	XGBoost (Chen & Guestrin, 2016); PyTorch-Forecasting TFT (Lim et al., 2021)	Demand forecasting.
ML — anomaly	Isolation Forest (F. T. Liu et al., 2008); variational autoencoder (An & Cho, 2015)	Inventory anomaly detection.
ML — risk	XGBoost classifier with SHAP (Chen & Guestrin, 2016; Lundberg & Lee, 2017)	Supplier risk scoring and attribution.
API layer	FastAPI	REST and WebSocket endpoints.
Interoperability	Model Context Protocol (Anthropic, 2024)	Standardized agent-tool interface.
Containerization	Docker, docker-compose	Reproducible deployment across environments.
Load testing	Locust	Concurrent-user scalability evaluation.
Testing	pytest	Unit and integration tests.
Structured logging	structlog	JSON-formatted audit and debug logs.

but merely perturbs absolute ones; therefore, the analytical behaviour seen in the framework must be understood as reflective as opposed to commercially normative. Second, the sample is selected based on two or three Bangladeshi firms and might not be representative of the full range of the Bangladeshi industrial environment, in addition to providing supply chains in any other emerging economy. Third, decisions have implications on the portability of conclusions made by a single LLM provider. Some of these threats are addressed by making sure that the partner companies represent four different industry fields (service-based supply chain operations, pharmaceutical, RMG, and FMCG), by recording all design choices in enough detail so that they can be reassessed with other models or datasets, and by explicitly conceptualizing the Bangladesh case as a validation context instead of a sampling frame of statistical generalization. Chapter 7 revises these threats.

Conclusion validity has to do with how reliable the statistical or interpretive inferences are drawn. Since the benchmark includes a relatively small number of queries (60), inferential statistical testing cannot be powerful, and thus, the results are presented in the form of descriptive statistics (mean, percentile, effect size) as opposed to hypothesis testing. Distributional summaries are given where repeated measurements (latency distributions, repeated ablation runs) are accessible.

3.8 Ethical Considerations

Real data on actual operations of the partner companies in Bangladesh adds ethical obligations that are resolved with the help of a systematic framework according to both the ethical standards of the University of Vaasa on research study and the international best practice regarding the industrial data sharing (Elliot et al., 2016; European Parliament and Council of the European Union, 2024; High-Level Expert Group on Artificial Intelligence, 2019).

Institutional ethics endorsement. Before data could be acquired, this study was approved by the appropriate University of Vaasa review board. This was agreed to authorize the use of anonymized commercial data in academic research by third-party organiza-

tions, storage and devolution of data throughout and after the research and commitment of confidentiality to the partner organizations.

Informed consent of partner organizations. Before any data was given the written informed consent an informed consent letter in the form of a signed letter was given by each partner organization. The consent forms state: (i) the intended purposes of the data, limited to this master's thesis work and any academic dissemination of it; (ii) the privacy guarantee that no single partners identity, commercially sensitive absolute values or identifiable references will be present in any published product; (iii) the right of the researcher to publish his or her thesis and the subsequent derived academic work utilizing the anonymized dataset; (iv) the right of the partner to review any draft content that references the partner context prior to public release. These consent forms will be stored at the University of Vaasa and can be accessed by the examining committee upon request.

Anonymization before transfer. According to Section 3.4, anonymization was done by the partner organizations themselves before data was dispatched to the researcher. This configuration prohibits the researcher ever possessing raw commercially sensitive data, and puts the quality of anonymization directly in the hands of data owners, those who are best-informed to decide what changes are sufficient to protect their interests.

Data storage and access. Stored data are anonymized, in University of Vaasa computing infrastructure, and accessible only to the researcher and thesis supervisors. There is no information disclosure to third parties outside the examining committee. Data will be managed according to the ethics approval terms on retention and disposal when the thesis is completed and archived.

Compliance with AI ethics principles. In addition to the data ethics, the framework is developed based on the principles per European Union Ethics Guidelines on Trustworthy AI (High-Level Expert Group on Artificial Intelligence, 2019) and the EU AI Act (European Parliament and Council of the European Union, 2024). They are principles that are made transparent by traces of the decisions and by remaining human (capability to escalate the

situation when humans are not persuaded), this principle is covered in Chapter 4 in the design of the trust and safety layer and is essential to data governance (role-separated access to the data), and accountability (complete audit logging).

Emerging-economy considerations. The last ethical consideration linked to cross-context research is that AI tools developed and tested in a specific economic setting might make assumptions not in favor of organizations that function in alternative economic settings. This thesis aims to examine Bangladeshi supply chains on their own principles, but not against a Western normative standard, and assessment does not place the Bangladeshi practice of operation in relative deficit to any other context. This position is reprimanded in Chapter 7.

3.9 Chapter Summary

This chapter has introduced Design Science Research as the methodological framework of the thesis, mapped the DSR process activities to specific chapters in the thesis, introduced the Bangladeshi industrial supply chain context of interest to the University of Vaasa reader, specified the strategy to be used in the data acquisition based on the anonymized operational data of two or three partner firms within the five industry sectors, described the evaluation protocol with sixty benchmark queries in five categories and seven ablation configurations, Having made these methodological commitments, in the following chapter, the framework design and architecture that are the heart artefact of this thesis are described.

4 Framework Design and Architecture

In this chapter, the design of the proposed AI agent framework, which leads to trustful and scalable industrial supply chain analytics, is developed. Section 4.1 explains the design principles guiding the design. The high-level architecture is found in Section 4.2. The major modules are described in Sections 4.3 to 4.8 and are schema linking and knowledge graph, RAG pipeline, ML analytics, multi-agent orchestration, text-to-SQL engine, and the trust and safety layer. The explainability subsystem is defined in Section 4.9. Section 4.10 describes the scalability design. Section 4.11 deals with external interfaces, one being REST API and Model Context Protocol (MCP) server.

4.1 Design Principles

The five design principles feature in this framework deal with each of them directly or additional objectives of the research mentioned in Chapter 1:

1. **Schema-linking-first:** Full-schema prompting should never be applied in enterprise size databases by the agent. Each query reads dynamic relevant tables, where applicable. The need to work with thirty or more interrelated tables required by Objective 1.

2. **Domain decomposition:** Tables are arranged as business domains, and specialists the agents are allocated domain wise. The queries are directed to the relevant specialist(s) by a central orchestrator and assist the separation of concerns which enhance accuracy and maintainability.

3. **Multi-stage safety:** Objective 2 is achieved through four checkpoints with safety being enforced: input validation, tool-call gatekeeping, SQL pre-execution analysis and output validation, both against Objective 2.

4. **Explainability by default:** Each agent action returns a decision trace containing the track of reasoning, used tools, data accessed, and confidence change. Traces are the artefact of the system, not a debugging option.

5. **Evaluations-readiness:** These components are designed in a way that allows ablation analysis (Chapter 6) to independently enable or disable each component of the system, and measures of latency and accuracy can be measured without changing code.

4.2 High-Level Architecture

Figure 1 illustrates the framework in the form of a three-horizontal plane (data, knowledge, and analytics) and a vertical plane (agent orchestration) that binds the three. All user-facing interactions go through the trust and safety layer which wraps the orchestrator and is denoted as a cross-cutting concern.

The path of the query flow through the architecture follows the following way. The user query is received through the REST API (or MCP server). Input validator checks adversarial input or out-of-scope input. The orchestrator then does query splitting in case the query crosses multiple domains, determines defining tables by the schema linker and assigns sub-tasks to the relevant domain specialist agents. Every specialist agent can make tool calls, where each tool call can involve the tool gatekeeper. Outputs of the specialists are assembled by the orchestrator, and validated by the output validator to ensure grounding and citation compliance, and are sent back to the client along with a decision trace and confidence score. Each phase generates systematic logs to audit.

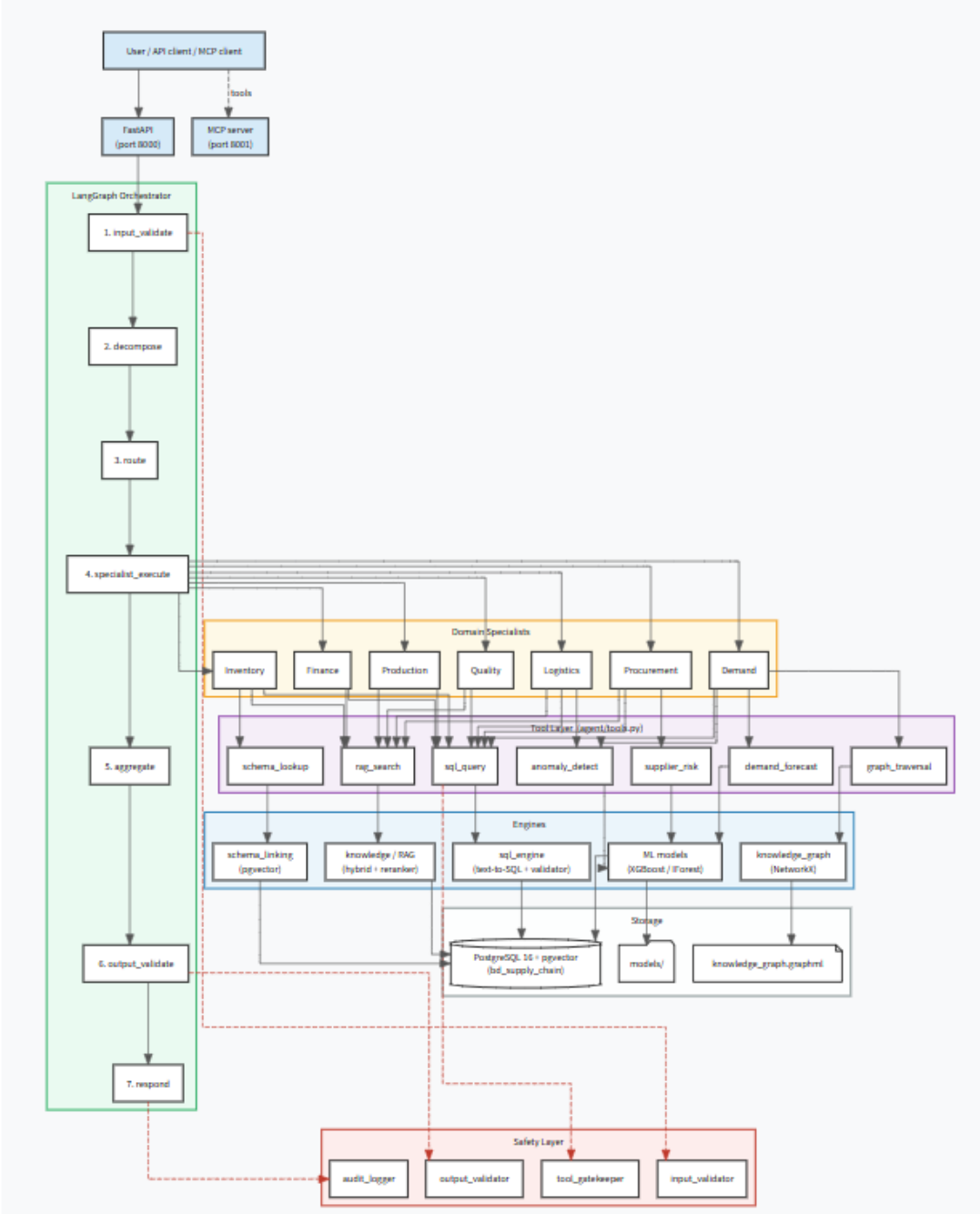

Figure 1. High-level architecture of the proposed framework: the seven-stage LangGraph pipeline (left), the seven domain specialists (centre-top), the tool layer and engines (centre), the storage layer (centre-bottom), and the safety layer (bottom). The framework is fronted by FastAPI for REST clients and an MCP server for protocol-compatible clients..

4.3 Schema Linking and Knowledge Graph

The main process whereby the framework manages to handle a thirty-seven-table schema is the schema linking module. It aims at converting every query that is received into a narrowed down schema scenario that only includes the tables and columns pertinent to that query. The design is based on the text-to-SQL literature about schema retrieval (Lei et al., 2020; Li et al., 2024; Pourreza & Rafiei, 2024) and scaled to the supplies of enterprise-scale supply chains.

4.3.1 Schema Indexer

The schema indexer indexes all tables in PostgreSQL by profiling of the `information_schema` catalogue at system start time. It creates a succinct natural language description of each table and important column, conditioned on column name, data type, constraints and a small sample of non-null values. The sentence-transformer model (Section 4.4) is embedded within each description and stored in the `schema_metadata` system table along with the raw metadata. This is a single operation; any changes made to the schema after this will cause incremental re-indexing.

4.3.2 Linker and Navigator

The schema linker at query time will store the query issued by the user and retrieve the top-K closest tables (by default $K = 7$) through pgvector similarity search. The candidate set that has been retrieved is subsequently expanded using foreign-key edges in a manner that tables commonly joined with candidates are also included thereby rendering it possible to generate SQL using required joins. The navigator component finds the shortest join path among two tables based on a graph representation of the foreign-key structure such that the text-to-SQL generator itself does not have to reason about join path based on schema metadata.

4.3.3 Knowledge Graph

NetworkX is used to create a complementary knowledge graph (Hagberg et al., 2008). Nodes denote the tables, key columns, and entity classes found in domain documents (e.g., Supplier, Bonded Warehouse, LC Contract); Edges denote the relationship that are a foreign-key relationship, co-occurrence of columns in queries, and semantic associations as found in policy documents. The graph allows multi-hop queries which cannot be effectively answered by pure vector retrieval, like trace how a supplier in Chattogram is connected to customer complaints on a particular product category. The design is an adaptation of recent graph augmented retrieval (Edge et al., 2024; Peng et al., 2024) studies to a database-schema context.

4.4 RAG Pipeline for Domain Knowledge

The RAG pipeline basifies outputs of the agent on the domain knowledge base comprising of ten written policy documents in Chapter 3. The pipeline is based on the indexing-retrieval-generation scheme of Y. Gao et al. (2024) with three additions befitting the industrial context.

Semantic chunking: Markdown heading hierarchy is used to chunk documents, not based on a fixed number of tokens, so documents contain semantically cohesive chunks representing topical units (e.g. one section of the procurement policy). Document name, section heading, domain tag and an effective-date annotation are metadata attached to each chunk.

Hybrid retrieval: Retrieval brings together dense semantic search (via pgvector) and sparse lexical search (via PostgreSQL's built-in `tsvector`/`ts_rank`) and is integrated using $k = 60$ Reciprocal Rank Fusion (Cormack, Clarke, & Büttcher, 2009). On industrial corpora where exact terminology — for example, Bangladesh-specific terminology like LC numbers, BSTI codes, BIN, HS codes — might be under-weighted by dense embeddings alone, hybrid retrieval reliably outperforms dense-only retrieval (Patel et al., 2024). Vector retrieval employs an HNSW index (Malkov & Yashunin, 2020) for sub-linear

approximate nearest-neighbour retrieval search.

Cross-encoder reranking: Rescore the top twenty hits of hybrid retrieval with a BGE cross-encoder reranker (Xiao et al., 2024) that is applied to a query existing-chunk pair generating a fine-grained relevance score. The five rescored chunks are re-served to the agent and then the source citations are provided. Each retrieved chunk has a permanent id that is checked by the output validator to enforce compliance to citation: any policy-based claim in the eventual response should be referenced to at least some retrieved chunk.

4.5 ML Analytics

The ML analytics module offers the three capabilities, each of which is constructed in such a way that the agent can call it as a tool and explain the outcome using an explanation metadata.

Demand forecasting: The basic model is an XGBoost regressor (Chen & Guestrin, 2016) that is trained on per-product daily demand sequences, and incorporates lag features (7, 14, 30 days), rolling features, calendar features (day of week, month, year), seasonal markers following Bangladeshi commercial calendars (Eid, Puja, RMG seasonal cycles). A Temporal Fusion Transformer (Lim et al., 2021) is a comparison model of selected categories where attention-based temporal modelling can enhance accuracy. The agent decides between the models depending on the available history length and characteristics of the products.

Anomaly detection: Shipment and inventory anomaly detection with Isolation Forest (F. T. Liu et al., 2008) is selected due to its strength in its ability to operate with high-dimensional feature spaces, low training cost and because its behavior is insensitive to its feature contributions in the final result. Detected anomalies are given with a score, a binary flag, and human read fields that give details on what features have most contributed to the score.

Supplier risk assessment: The output of a logistic regression ensemble is a three-level risk (Low, Medium, High) per supplier based on a set of features that are derived based

on `supplier_evaluations` on overall `purchase_orders` based on the on-time delivery data of suppliers, the number of incidents by the supplier based on logistic accident data, and rates of defects based on `quality_inspections` by the supplier. Instead of more complex alternatives, logistic regression was adopted since the coefficients could be easily interpreted as feature level contributions to risk to aid transparent procurement decisions without opaque post-hoc explanation methods.

Model registry: A trained model is recorded in a `model_registry` table and all models have their versions, the date of the training, dataset hash, hyperparameters, metrics and file path. Each agent invocation records the model version called upon, aiding in reproducibility and longitudinal study of model drift.

4.6 Multi-Agent Orchestration

An orchestration layer is the state graph of a LangGraph (Chase, 2022) consisting of central orchestrator and seven domain specialist agents, each representing one of the business areas. Such a design achieves Principle 2 (domain decomposition) and builds on known evidence of heterogeneous task space demonstrating that specialist agents outperform generalists (Guo et al., 2024).

4.6.1 Orchestrator

The orchestrator implements a seven-node workflow: *input validate* $\rightarrow$ *decompose* $\rightarrow$ *route* $\rightarrow$ *specialist execute* $\rightarrow$ *aggregate* $\rightarrow$ *output validate* $\rightarrow$ *respond*. The decision on whether to answer a query with one domain or to decomposition calls on the LLM to split a query is made by the decomposition node; cross-domain queries are broken down into sub-queries that have explicit dependencies. The routing node refers to schema linker and domain configuration to either pick one or more specialists. Aggregation merges the outputs of the experts into a consistent answer, eliminating the conflicts with favoring answers by higher data-confidence ratings. Ten steps are the maximum number of iterations to ensure runaway loops do not occur.

4.6.2 Domain Specialists

Every specialist agent is a LangGraph sub-agent which has three types of domain context: (i) a system prompt about the domain, terminology and decision heuristics about the domain; (ii) at least twenty few-shot SQL examples based on representative queries; and (iii) a list of its major tables and ML tools. The orchestrator treats the same tool interface to specialists; thus, they are treated in the same way. The seven experts are in the areas of procurement, inventory, demand, logistics, production, quality and finance.

4.6.3 Reasoning Pattern

In every specialist, the thinking pattern is that of ReAct (Yao et al., 2023): Thought $\rightarrow$ Action (tool call) $\rightarrow$ Observation $\rightarrow$ Thought. Another repeat the same process up to the production of an answer or until the limit of thinking is reached. In case of complex tasks, the orchestrator applies a Plan-and-Execute version (Z. Wang et al., 2024), where an explicit plan is generated, as part of which it is then delegated and thus enhances auditing.

4.7 Text-to-SQL Engine

The text-to-SQL translator maps the data-access intent of each specialist to validated and executable PostgreSQL. It has a design that explicitly isolates the generation, decomposition, validation, and execution in such a way that any of the stages can be measured and ablated.

Generation: The generator provides the focused schema context provided by the schema linker, a small number of domain-matched few-shot examples provided by the per-domain example bank, and explicit style and safety instructions (read-only only, at least always add a LIMIT to exploratory queries, always use explicit JOIN syntax). When the execution fails, the generator is rerun with the error message attached, a form of self-correction that can significantly enhance the robustness (Pourreza & Rafiei, 2024).

Decomposition: Have complex multi-join queries a decomposer generates a series of

simpler queries the results of which are recombined in one last step. This will decrease the thinking load on the generator and make downstream validation easier (Pourreza & Rafiei, 2024).

Validation: Each query generated is run through a required validator (Section 4.8).

Execution: The execution is with an agent role of PostgreSQL called `agent_readonly` and a session-level `statement_timeout` of thirty seconds. The outcomes are provided in the form of pandas DataFrames and are present in the auditable verbatim decision trace.

4.8 Trust and Safety Layer

The multi-stage enforcement principle is the one implemented by the trust and safety layer. It includes five sub-modules which tackle each different category of risk that was identified in Chapter 2.

Input validator: Checks for prompt injection patterns based on a combination of regular expressions (against known attack strings, e.g., “ignore previous instructions”) and indirect injecting is checked by the semantic check based on LLM. Enforces length (2,000 character) and topic boundary limit, returning invalid queries representing queries that are not in the scope of a supply chain.

Tool gatekeeper: The safest part that is the most important. It interprets each SQL generated read-only and it has to obey the following rules without exception: (i) read-only enforcement — rejecting any INSERT, UPDATE, DELETE, DROP, ALTER, TRUNCATE, CREATE, GRANT statement; (ii) injection defence — rejecting stacked queries, suspicious patterns of commenting, and exfiltration via UNIONs; (iii) complexity limits — no more than seven JOINs and three subquery nesting levels; (iv) cartesian-product detection — rejecting joins are not specified with explicit join conditions; (v) estimate cost using PostgreSQL `EXPLAIN` prior to allowing execution; and (vi) auto-append of `LIMIT 1000` to non-aggregate queries. A combination of these rules will be the key safety measures against the risks pointed out in Li et al. (2024) and OWASP Foundation (2023).

Output validator: This does the three checks on the natural language response of the agent. First, numerical grounding: each numerical assertion should be associated with a value in the decision trace, which is the output of an SQL query, or an ML prediction. Second, compliance with citation: any policy claim should reference a retrieved chunk of documents. Third, PII detection: The response is examined to identify personal identifiers (including email addresses, identification numbers) that do not belong.

Audit logger: Each agent action shall be entry into the `audit_logs` table in the structured form of a JSON record, which contains the following fields: timestamp, user identifier, query text, tools invoked, SQL run, confidence score, safety flags and the end result text. The logs can also be written to file sink to facilitate further integration with expenditure downstream monitoring systems.

4.9 Explainability Subsystem

The explainability subsystem is a realization of the principle of design stating that a trace of the response to any decision-making of an agent is present. This is achieved by three elements:

Trace builder: step by step: input query, schema linker-selected tables, each tool (with input and output) call, each SQL query generated and result of its execution, each retrieved document fragment, confidence of each step.

Confidence decomposition: The end-point confidence number is not simply an opaque number. It is broken down into three components (Kaur et al., 2022): data confidence (indicated the presence of the needed SQL results and their non-emptiness), knowledge confidence (indicated the retrieved documents to be above the reranker threshold), and model confidence (self-reported confidence or uncertainty estimate of the predictive model).

Visualizer: Turns the trace into an HTML report that gives each claim that appears as part of the final response its source. The report is available through a special API endpoint and it is one of the main artefacts provided to examiners and stakeholders to assess the trust.

4.10 Scalability Design

Scalability is sought after on the three axes that were identified in Chapter 2: data scalability, computational scalability and user scalability.

Schema linking (avoiding prompt-size growth with schema size) and PostgreSQL indexing on foreign-key and date columns are data scalability strategies, where queries executed with pgvector search with HNSW index sub-linearly.

The problem of computational scalability is favored by storing the most commonly used queries and retrievals in caches, performing ML inference asynchronously, and sharing database connections. Continuous tool calls do not preempt the orchestrator, instead giving interim updates through the streaming interface.

The stateless service containers or the use of FastAPI layer, rate limiting in the gatekeeper and the ability to have horizontal replication by the use of docker-compose profiles is a way of addressing user scalability. Chapter 6 reports evaluation of performance of one, ten, fifty, and one hundred simulated users.

4.11 External Interfaces

Two external interfaces are available: standard REST API and a Model Context Protocol server.

REST API: A FastAPI application provides a query endpoint charged with the main agent query (`POST /api/query`), direct access to the ML (`/api/forecast`, `/api/supplier-risk`, `/api/anomalies`), the decision trace of a previous query (`GET /api/trace/{id}`), an audit log (only admins), and a WebSocket stream that tracks full responses.

MCP server: The Model Context Protocol (Anthropic, 2024) is a rising open convention of exposing AI tools and data sources by using an identical JSON-RPC interface. The agent tools and knowledge resources of the framework are encircled by a special MCP server

that allows them to be consumed by clients that are compatible with MCP (desktop assistants, built-in development environments, custom applications) without any special code that integrates with the framework. Making the framework transparent through MCP enhances its feasibility and makes its thesis to be in line with the existing trends in agent interoperability.

4.12 Chapter Summary

The design of the proposed framework has been outlined in this chapter. The design is structured into a layered architecture with data-access, knowledge-retrieval, analytics, agent-orchestration and trust-and-safety-planes, where explainability is performed across all layers. The fundamental design choices — schema linking of enterprise-scale schemas, domain specific multi-agent coordination, hybrid retrieval with cross-encoder reranking, ML analytics with versioned model registry, requirement to use multi-stage safety, orchestration to explainability at first class, and the dual REST/MCP interfaces — address research objectives mentioned in Chapter 1 and research gaps in Chapter 2. The second chapter explains how this design is realized.

5 Implementation

This chapter outlines the revalorization of the framework whose design was outlined in Chapter 4. This is done to recreate the engineering artefact of the thesis so that it can be examined and used in future research, but is not done to recreate all the source files. The development and deployment environment is mentioned in Section 5.1. Section 5.2 has the database schema implementation. Section 5.3 explains the schema element of connection and knowledge graph. Building knowledge base is outlined in Section 5.4. Section 5.5 discusses the machine learning module. The text-to-SQL engine is outlined in Section 5.6. The trust and safety layer is described in Section 5.7. Multi-agent orchestration is outlined in Section 5.8. Section 5.9 is about the REST API, MCP server, and the Streamlit frontend. The containerized deployment is given in Section 5.10. The testing infrastructure is discussed in Section 5.11.

5.1 Development Environment and Project Organization

The design is written in Python 3.11 and structured into a modular monorepo with the name of the project being *supply-chain-agent*. The senior folders hierarchy is a reflection of the architectural planes introduced in Chapter 4: *config/*, *db/*, *schema_linking/*, *knowledge_graph/*, *knowledge/*, *ml/*, *sql_engine/*, *safety/*, *explainability/*, *agent/*, *api/*, *mcp_server/*, *frontend/*, *evaluation/*, *load_test/*, and *tests/*.

Dependency management comes with toml files called *pyproject.toml* and imported *requirements.txt* manifest, exported to use in Docker builds. Pydantic *BaseSettings* is configured centrally, in *config/settings.py*, reading all the environment variables (database credentials, OpenAI API key, model identifiers, rate limits) from an *.env* file on startup. *structlog* has structured logging features, with output in the JSON format, which enables all log entries to be mechanically processed and forwarded to a downstream log aggregator when it is being evaluated.

5.2 Database Schema Implementation

The database is PostgreSQL 16 and pgvector extension to search similarities among vectors. The schema contains thirty-three business tables that are grouped into seven domains and four system tables, making it thirty-seven tables.

Business tables cover procurement (*suppliers*, *supplier_contracts*, *supplier_evaluations*, *purchase_orders*, *rfq_requests*, *supplier_audits*), inventory (*product_categories*, *products*, *warehouses*, *inventory_levels*, *stock_movements*, *cycle_counts*, *reorder_rules*), demand (*customers*, *sales_orders*, *sales_order_lines*, *demand_history*, *promotions*), logistics (*carriers*, *shipments*, *delivery_tracking*, *freight_costs*, *logistics_incidents*), production (*work_orders*, *bill_of_materials*, *production_schedules*, *production_runs*), quality (*quality_inspections*, *defect_records*, *quality_certificates*), and finance (*invoices*, *payments*, *cost_centers*).

The framework is supported by system tables: *schema_metadata* is the LLM-generated language descriptions of each table and each column along with a 1,024-dimensional vector embedding; *document_chunks* tables contain the document chunks and their embeddings, source metadata, and a compiled *content_tsv* column to allow full-text search; *audit_logs* are structured-JSON records of all agent actions; and *model_registry* consists of versioned machine learning models.

It is declared in a start-up script in *db/init.sql* and mounted into the PostgreSQL container, which is an initialisation script containing schema creation, such that schema creation is idempotent and automatic the first time the container starts. Primary and foreign keys are defined explicitly, indexes are generated on any foreign-key columns, and any date columns often used in analytical queries, and a GIN index is provided to support the *content_tsv* full-text column.

Security separation is provided with two PostgreSQL roles. The application uses *app_admin* role with schema evolution, the ML training, and audit logging functionality. Only the text-to-SQL executor can use the *agent_readonly* role, and it has *SELECT* permissions on business tables only, and cannot write anywhere in the schema. This

role separation forms the first defence line in the trust and safety layer (Section 5.7) and that even a successful prompt injection with the aid of the LLM cannot corrupt the database. *db/seed.py* consumes anonymised per-partner data exports and multi-organization namespacing uses a partner-code tag, data validation then insertion in dependency order, is handled by *db/seed.py* (Chapter 3).

5.3 Schema Linking and Knowledge Graph

Clicking schema between modules is carried out in three files. *schema_linking/schema_indexer.py* is indexed single time: The query *information_schema* enumerates every table and every notable column, the query collects up to five non-null sample values per column, and the call is made to the OpenAI API with a succinct prompt that requests a one-sentence natural-language description of each table and each important column. The *BAAI/bge-large-en-v1.5* sentence-transformer model is loaded and saved in *schema_metadata* as its description.

schema_linking/linker.py puts into effect query-time retrieval of table rows. This is represented as one query, and does it again (via query-level memorization, Section 5.4). A pgvector cosine search displays hits down to the column level (that is, the hit is not isolated to any single row). The hits are then aggregated to an individual table, by taking the maximum column score per table. A one-hop foreign-key expansion inserts the tables which are related to the candidates in such a way that maintain the join feasibility. The graph itself representing the foreign keys is fetched by the *information_schema* at the onset and is memorized using the *@lru_cache(maxsize=1)* during the lifetime of the process since the schema remains fixed at the time of running the process.

schema_linking/navigator.py maintains a NetworkX directed graph of foreign keys (Hagberg et al., 2008) and exposes *shortest_join_path(source_table, target_table)*, used by the text-to-SQL generator to obtain explicit join paths.

The knowledge graph in *knowledge_graph/builder.py* is an extension of this format and uses four node types: *table*, *column*, *document* and *concept*. The edges are drawn either

based on the relationships between the foreign-key and schema descriptions (structural), or based on the keyword co-occurrence policy documents and schema descriptions (semantic). The graph also supports multi-hop traversal queries, like the query given above, which would mean that two FK hops are required to get to the table. The entire graph is mirrored to the *data/schema_graph.graphml*.

5.4 Knowledge Base Construction and Hybrid Retrieval

The RAG pipeline is affected in two files, in *knowledge/indexer.py*, the ten markdown documents from *knowledge/documents/*, splits each document by heading hierarchy so that each chunk corresponds to a coherent topical unit, attaches metadata (document name, section, domain tag, effective date), embeds each chunk with *bge-large-en-v1.5*, computes a *tsvector* for the text, and upserts the chunks into the *document_chunks* table. Indexing is also idempotent: rerunning the indexer replaces previous chunks of each document based on a hash of its content.

knowledge/retriever.py implements the three-stage hybrid pipeline of retrieval described in Chapter 4:

1. *Semantic retrieval:* pgvector HNSW (Malkov & Yashunin, 2020) cosine search gives the best thirty results by embedding similarity.
2. *Lexical retrieval:* PostgreSQL *ts_rank* over the precomputed *content_tsv* column yields the top thirty results (by lexical relevance) to ensure that Bangladesh specific terminology (LC numbers, BSTI codes, HS codes, BIN identifiers) is not underweighted.
3. *Reciprocal Rank Fusion:* RRF (Cormack et al., 2009) is used to combine the two ranked lists with $k = 60$. The top *rag_top_k* (default 5) fused candidates are passed to the *BAAI/bge-reranker-large* cross-encoder reranker to final rescoring.

Practical optimizations are used in two practical ways. Passages are truncated allowing up to the next 512 character, and then Query embeddings are memorized with

@lru_cache(maxsize=256); since both schema linking and RAG embed the same query string per turn, this halves the embedding cost per query.

5.5 Machine Learning Module

Training and registering three models *ml/demand_forecasting.py* use lag-1, lag-7, lag-14 and lag-28 features rolling means and stds, day-of-week and month encoding, seasonal markers aligned with Bangladeshi commercial calendars (Eid, Puja, RMG seasonal cycles), promotion flags from *promotions*, product category encodings and trains an XGBoost regressor on each per-product daily demand series. A Temporal Fusion Transformer (Lim et al., 2021) was trained as a standalone baseline comparison model. Each model returns a point forecast and an interval forecast per (product, region, horizon).

ml/anomaly_detection.py trains an Isolation Forest (F. T. Liu et al., 2008) on daily features per warehouse and per product: quantity on hand, rate of change, deviation from safety stock, capacity utilization ratio, and recent stock-movement signals from *stock_movements* and *logistics_incidents*. The detector returns an anomaly score and a binary flag (along with the per-feature contributions to the score) used downstream by the agent to explain why a flag was raised.

ml/supplier_risk.py is trained an XGBoost classifier that outputs a multi-level risk grade (A–D) based on supplier features aggregated from *supplier_evaluations*, the on-time delivery rate *purchase_orders* and the incident frequency of *logistics_incidents* and defect rates found in *quality_inspections*. Attributions regarding SHAP (Lundberg & Lee, 2017) exposed at the API surface (*/api/supplier-risk*) are provided along with predictions to allow callers to pass per-feature contributions along with the predicted grade, as shown by Streamlit Supplier risk interface (Appendix B).

ml/model_registry.py writes a record into the *model_registry* for every training run, complete with model name, version number, timestamp of when it was trained, hash of the training set used to train, hyperparameters, metrics and persisted artefact path. *ml/training_pipeline.py* sequentially trains all three models with *make train*.

5.6 Text-to-SQL Engine

The text-to-SQL engine can be found in four files and it is validated *sql_engine/text_to_sql.py* through a five-step pipeline:

1. *schema_linking.linker.link_schema* returns a ranked, domain-filtered snippet of a schema, instead of the entire thirty-seven-table schema.
2. Domain-matched few-shot examples are loaded on disk and memorized with the help of *@lru_cache(maxsize=16)*.
3. The *SELECT*-only query against the given schema context generated by OpenAI GPT-4o (OpenAI, 2024) at *temperature=0.0* with explicit style and safety guidelines in the system prompt.
4. *sql_engine/query_validator.py* imposes no DML or DDL, single statement only, *LIMIT* $\leq$ 1000 and no more than 10 levels of subqueries. Multi-part strings are length-preserved then semicolon-position scan to avoid trying to split false statements.
5. *sql_engine/executor.py* of executing the validated executes the validated SQL under the *agent_readonly* PostgreSQL role and with a session level *statement_timeout* value of thirty seconds, where the end result is on a pandas DataFrame.

sql_engine/query_decomposer.py breaks complex multi-domain queries into focused sub-queries before generation, each of which goes through the same five-step pipeline.

5.7 Trust and Safety Layer

The trust and safety layer is realised in the following four primary components, with all of them being enforced as obligatory checkpoints without bypass flag.

safety/input_validator.py uses a forty-pattern library of regex patterns, which takes information about role-reassignment, instruction-override and environment-variable exfiltration, DML injection, and known prompt-injection attacks drawn from OWASP Foun-

dation (2023) and Perez and Ribas (2022). Zero-width characters and control codes are also striped, there is a 2,000-character length limit enforced by the LLM, an additional topic-boundary check that rejects queries outside supply chain analytics.

safety/output_validator.py scans the agent response out of system, py checks for presence of PII (specifically Bangladesh mobile-number and National Identity Card (NID) patterns), looks in response form hallucination markers or confidentiality leaks.

safety/tool_gatekeeper.py enforces per-specialist *tool_whitelists* policies. Every specialist has an explicit subset of the tools it might use; calls outside this whitelist are blocked at *agent/tools.py* layer before anything runs. Mandatory SQL query validator In Section 5.6 above we described a mandatory validation of the generated SQL queries This validator is itself part of that safety perimeter Every single formatted and/or interpolated (non-prepared) SQL query has to pass through it without exception.

safety/audit_logger.py writes at most one structured *AuditEntry* per agent invocation to both the *audit_logs* PostgreSQL table (queryable by the evaluation pipeline) and *structlog* JSON stream (consumable as a log record for external security-monitoring tools).

5.8 Multi-Agent Orchestration

We use the orchestrator based on LangGraph (Chase, 2022), and it runs through a deterministic seven-node graph: *input_validate* $\rightarrow$ *decompose* $\rightarrow$ *route* $\rightarrow$ *specialist_execute* $\rightarrow$ *aggregate* $\rightarrow$ *output_validate* $\rightarrow$ *respond*. This allows to read and write against a typed dataclass *AgentState* at each node. Any failures at any node are caught, logged on the active trace and routed via a *fail_safe* path so that user gets an organized response instead of unhandled HTTP 500.

There are a total of 7 specialist agents corresponding to the categories under *agent/specialists/* (procurement, inventory, demand, logistics, production and quality & finance) which all sub conditions based on *BaseSpecialist*. *BaseSpecialist* parallelizes SQL generation and RAG retrieval using *ThreadPoolExecutor(max_workers=2)*, calls the LLM with a domain-specific system prompt, and expects to receive an unstructured JSON

draft as output. Specialists differ only in their *domain* attribute (for filtering schema linking, RAG retrieval and the SQL few-shot example pool) as well *tool_whitelist* that states whether any ML tools are accessible. The orchestrator calls specialist in topological order based on the decomposed sub-questions.

Inside each specialist, reasoning follows the ReAct pattern (Yao et al., 2023) using a maximum of ten iterations to avoid runaway loops; and queries with an aggregated confidence below 0.6 are handled by passing through the orchestrator-human-in-the-loop path.

agent/tools.py treats each of these tools as LangChain *StructuredTool* objects and is your only execution layer for any tool call. *agent/memory.py* is per-session conversation memory with auto-summarisation, where the estimated token budget ($\sim$100K tokens via character-based heuristic) will cause old turns to be summarised into a rolling summary via single LLM call. *MemoryStore* singleton which keeps a mapping of session identifiers to memory instances in a thread-safe way.

5.9 External Interfaces

Includes three external interfaces: REST API, Model Context Protocol server and Streamlit frontend.

REST API: *api/main.py* exposes the agent via HTTP through four endpoint groups (Table 4). Authentication is bearer-token based, there are two keyrings: *API_AUTH_KEYS* for user keys and the other one which allows admin-level commands *API_ADMIN_KEYS* (loaded from environment variables on startup). Warm embedding and reranker models at application start-up via the FastAPI *lifespan* context manager, so that you do not pay for model-load cost on first user query.

MCP server: *mcp_server/server.py* implements the Model Context Protocol (Anthropic, 2024) over stdio and exposes seven tools that map to *ToolGatekeeper* one-to-one policies table: *sql_query*, *rag_search*, *schema_lookup*, *graph_traversal*, *demand_forecast*, *anomaly_alert*, and *supplier_risk*. Tools go through the same authorization layer as the HTTP surface. We do not expose the full *run_agent* orchestration loop as an MCP tool

Table 4. REST API endpoints exposed by the framework..

Endpoint	Auth	Description
POST /api/query	Bearer (user)	Runs the full agent pipeline; returns response, confidence, citations, trace identifier.
GET /api/trace/{id}[/html]	Bearer (user)	Returns structured trace as JSON or as an HTML decision-timeline visualisation.
GET /api/audit-logs	Bearer (admin)	Returns recent audit entries for monitoring and review.
GET /api/health, *GET /api/schema*	Open / user	System health check and schema introspection.

by design — MCP is an interface that provides building blocks, and the clients compose them. Schema descriptions and policy documents are exposed as MCP resources and curated analytical prompts as MCP prompt templates.

Streamlit frontend: A Streamlit application under *frontend/* that serves as an interactive demo interface consuming the REST API. It exposes six interaction modes: *Chat* (the conversational query interface), *Demand forecast* (a direct feed into the demand model), *Shipment anomalies* (a direct interface to the Isolation Forest detector), *Supplier risk* (direct access directly refers you to supplier risk classifier), *Trace explorer* (paths across which a systematic explanation of decision can be generated as an HTML trace from */api/trace/{id}/html*), and *Architecture* (The visual reference diagram about architecture of my framework). Screenshots of each mode are provided in Appendix B.

5.10 Containerized Deployment

The architecture is deployed as a set of Docker services defined in *docker-compose.yml*. Four services are defined:

1. *postgres* leveraging *pgvector/pgvector:pg16* image which mounts a volume at *db/init.sql* as an initialization script, makes port 5432 available, and saves data to a named volume.

2. *app* uses a multi-stage *Dockerfile*, non-root user, mounts local volumes for model

cache and logs, reads config from *.env* and exposes the FastAPI application on port 8000.

3. *mcp* reuse *app* image providing MCP server entry point running on the port 8001.
4. *frontend* executes the Streamlit app at port 8501, configured to hit the *app* service running on *http://app:8000*.

The *Dockerfile* allows a multi-stage build. The builder stage loads dependencies, and preregisters the *bge-large-en-v1.5* and *bge-reranker-large* model weights to a cache directory. The runtime level only duplicates the application code and the model cache and results in a smaller final image. A *Makefile* has convenience targets, such as *make up*, *make down*, *make logs*, *make psql* (invokes a PostgreSQL shell against the running container), *make shell*, *make test*, *make train*, *make eval*, and *make load-test*.

5.11 Testing Infrastructure

Tests are located in *tests/*, one file per module. *test_sql_validator.py* runs each of the validator rules against a curated corpus of adversarial queries including stacked queries, UNION-based exfiltration attempts, deeply nested subqueries and cartesian-product traps. *test_safety.py* exercises the prompt-injection corpus. *test_schema_linking.py* asserts that the linker gets an appropriate table set for reference queries. *test_tools.py* tests each agent tool individually. *test_orchestrator.py* queries the orchestrator with a replay of LLM-responses using *vcr.py*. *test_agent_e2e.py* tests live end to end against the docker deployed stack.

Load testing is divided into *load_test/locustfile.py*, which outlines concurrent user scenarios for one, ten, fifty, and one hundred users, and *load_test/data_scale_test.py*, which executes a representative benchmark subset at various database sizes: small, medium, and large. The results are recorded in *load_test/reports/* in both HTML and CSV formats.

5.12 Chapter Summary

So, this chapter has documented architecting the code that implements the design of Chapter 4. The implementation closely follows layered architecture, utilizes module boundaries and checkpoint-based safety to enforce the five design principles, and is packaged as a reproducible Docker deployment consumable via three external interfaces: REST API, a Model Context Protocol server and a Streamlit frontend. The preceding chapter from this implementation is an empirical evaluation on the approach described in Chapter 3.

6 Evaluation and Results

In this chapter, we report the empirical evaluation of the framework that was specified in Chapter 3 and implemented according to the design in Chapters 4 and 5. We evaluated on a Bangladeshi supply-chain database containing privacy-preserving operational data from 2–3 partner companies (Section 6.1) over the sixty-query benchmark in Section 3.5. The evaluation setup is summarized in Section 6.1. SQL generation accuracy is reported in Section 6.2. Retrieval Quality of RAG is in Section 6.3. End-to-end latency and stage-level timing are reported in Section 6.4. Results of the three machine learning models are reported in Section 6.5. The scale of the database, and the API cost profile, is reported in Section 6.6. In Section 6.7, the main limitations of the current evaluation are listed. Section 6.8 concludes the chapter.

6.1 Evaluation Setup

The study was conducted on a complete database covering all the seven supply chain domains. The three highest row counts in Table 5, which are the main transactional history tables (`demand_history`, `sales_order_lines`, `delivery_tracking`) total more than 460,000 records in the database, while reference and master data exists in the seven business areas. The knowledge base used in retrieval-augmented generation includes eighty-seven document chunks from the ten Bangladesh contextualized policy documents (Chapter 5) and three machine learning models which were trained and registered prior to the evaluation: `demand_forecast_xgb`, `anomaly_shipments_iforest`, and `supplier_risk_xgb`.

The experiment was done using the cost-conscious protocol defined in Chapter 3 with a prefetch-and-cache design where individual benchmark queries were redirected through the agent only once and full response to the benchmark query was then cached against subsequent evaluators. This architecture minimized the API cost compared to a naive per-evaluator architecture by a factor of four or so. Three of the evaluators were run against the stored response: `sql_eval`, `rag_eval` and `latency_eval`. Chapter 3 specifies

two additional evaluators, which were postponed to manage the budgetary allocation of the academic project: these are noted in Section 6.7 and are discussed in Chapter 7.

Table 5. Database population at the time of evaluation. Tables not shown contain reference data with row counts under one hundred..

Table	Rows	Table	Rows
demand_history	452,648	purchase_orders	400
sales_order_lines	2,378	supplier_evaluations	218
delivery_tracking	2,323	inventory_levels	237
freight_costs	1,771	production_runs	242
invoices	1,000	rfq_requests	160
sales_orders	800	bill_of_materials	126
payments	750	supplier_contracts	109
quality_inspections	567	defect_records	113
shipments	514	cycle_counts	100
stock_movements	500	work_orders	100
products	120	customers	80
reorder_rules	120	document_chunks (RAG)	87
suppliers	50	schema_metadata	361

The language model behind it was OpenAI GPT-4o at temperature zero to generate SQLs and other deterministic sub-tasks. All the evaluation queries were based on the sixty-query benchmark that was defined in Chapter 3, stratified into five categories: twelve queries each: factual, predictive, diagnostic, policy-grounded, and multi-hop reasoning.

6.2 SQL Generation Accuracy

Table 6 gives the SQL generation accuracy, over the forty-eight queries to which a SQL path could be applied. Predictive queries that only route to ML tools are not counted in the SQL evaluation, as they do not exercise the SQL generator.

The framework scored 93.8% executable coverage, and 93.8% schema-correctness on the forty-eight accessible queries. Executable is a query that can run to completion without error, against a PostgreSQL database; schema-correct a query that references only valid tables and columns from the focused schema returned by the schema linker; and domain-aligned a query whose SQL produced output that was substantively relevant to the user's

question.

Table 6. SQL generation accuracy: overall and per category..

Category	n	Executable	Schema-correct	Domain-aligned
Factual	12	100.0%	100.0%	83.3%
Diagnostic	12	100.0%	100.0%	83.3%
Multi-hop	12	91.7%	91.7%	91.7%
Predictive	12	83.3%	83.3%	58.3%
Overall	**48**	**93.8%**	**93.8%**	**79.2%**

The patterns of category breakdown demonstrate two patterns that can be interpreted. First, the schema-linking-first design demonstrated perfect executable and schema-correctness rates, indicating that the schema-linking-first design can be used to isolate the relevant portion of the tables in the thirty-seven-table schema. Second, queries based on multi-hop reasoning scored 91.7 percent on all three measures, including domain alignment, the best domain-alignment score is obtained. This is notable because multi-hop queries consume multiple domains, and have traditionally posed the largest challenge to text-to-SQL systems on small benchmarks (Li et al., 2024; Yu et al., 2018); the high performance here is attributed to the query decomposer (Section 5.6) and the foreign-key navigator that provides explicit join paths to the SQL generator.

The weaker category is the predictive queries: 83.3% of the queries are executable and 58.3% of the queries aligned to the domain. Predictive queries are created to make the run to ML tools instead of the SQL, meaning the SQL lookups in this program are auxiliary lookups which supplement the ML output. The lower domain alignment indicates that not every predictive sub-query has a unique canonical SQL formulation.

6.3 RAG Retrieval Quality

Retrieval performance on the twenty-two queries for which a RAG path was possible are shown in Table 7. Retrieval coverage checks if at least one chunk from the actual policy document family was returned in the top-k retrieved set; domain match verifies whether the majority of retrieved chunks belonged to the expected business domain of the query;

citation density counts the number of retrieved chunks cited over one hundred words of generated response.

Table 7. RAG retrieval quality: overall and per category. Sample sizes vary by category because RAG is exercised primarily by policy-grounded and multi-hop queries..

Category	n	Retrieval coverage	Domain match
Policy-grounded	12	83.3%	75.0%
Diagnostic	5	80.0%	40.0%
Multi-hop	4	100.0%	100.0%
Predictive	1	100.0%	0.0%
Overall	**22**	**86.4%**	**68.2%**

The hybrid retrieval pipeline found an average 86.4% retrieval coverage and 68.2% domain match over 22 relevant queries, while using pgvector for semantic search, PostgreSQL `ts_rank` for lexical search, fused with Reciprocal Rank Fusion at $k = 60$ (Cormack et al., 2009), followed by reranking using a BGE cross-encoder (Xiao et al., 2024). This means the retriever surface relevant policy content with consistently high retrieval coverage across categories. Domain match is a little more forgiving, with the gap focused on diagnostic queries (40.0%); as many diagnoses are for conditions that legitimately involve one or more domains (e.g., a delivery delay diagnosis invokes both logistics and procurement policies) and the strict domain match criterion punishes retrievals that span domain boundaries even for correct retrievals.

Again, multi-hop queries out-perform the others, achieving both 100% coverage and 100% domain match on four applicable queries (the very small number of queries means this claim is weak). The average citation density for all of the responses was 0.367 chunks per one hundred words, which implies the agent uses citations economically and avoids over-citing.

6.4 Latency Performance

Table 8 reports end-to-end response latency across the full sixty-query benchmark, and Table 9 reports the median time spent in each pipeline stage.

Table 8. End-to-end response latency (milliseconds) across all sixty queries..

Statistic	Value (ms)
Median (p50)	26,483
95th percentile (p95)	85,750
99th percentile (p99)	88,558
Mean	32,649

Table 9. Median time per pipeline stage (milliseconds), aggregated across sixty queries..

Stage	Median (ms)
Input validation	0
Output validation	1
SQL execution	14
Decomposition	3,032
SQL generation	4,523
Synthesis	10,394
RAG retrieval (incl. reranking)	11,493

Median end-to-end latency is $\sim$26.5 s and 95th percentile latencies are $\sim$85.7 s. These numbers are in line with RAG-enabled agent systems whose latency profile we found reported in the literature (Mao et al., 2024; Patel et al., 2024), particularly if service of the underlying LLM is provided via hosted API. The breakdown by stage clearly isolates the main contributors, with RAG retrieval (including cross-encoder reranking) at 11.5 seconds and synthesis (final LLM call to aggregate evidence into a response) at 10.4 s making up over 80% median end-to-end time together. The reason for a large proportion using 4.5 seconds is SQL generation, and SQL execution itself contributes only 14 milliseconds at the median, so the database and PostgreSQL execution is not the bottleneck either. Input and output validation are effectively free at single-digit-millisecond cost, vindicating the multi-stage safety design as having negligible runtime overhead.

It is these observations which highlight cross-encoder reranking and final synthesis as the primary opportunities for latency optimization in the future. Potential mitigations are cached reranking of the same query, distillation of the cross-encoder and usage of a smaller LLM for synthesis if high-confidence.

6.5 Machine Learning Model Performance

In Table 10, the three trained machine learning models were picked from a model registry containing the metrics for each one.

Table 10. Performance metrics of the three trained ML models, recorded in the `model_registry` table..

Model	Algorithm	Reported Metrics
`demand_forecast_xgb`	XGBoost regressor	MAE: 38.3 units; RMSE: 52.5 units; SMAPE: 10.2%; train rows: 429,520; test rows: 11,564.
`anomaly_shipments_iforest`	Isolation Forest (200 estimators, contamination 0.05)	436 samples scored; 22 flagged as anomalous; score range ∼0–0.69; mean score 0.47; standard deviation 0.057.
`supplier_risk_xgb`	XGBoost classifier with SHAP	Accuracy: 50%; F1 weighted: 0.45; F1 macro: 0.31; train rows: 40; test rows: 10; class distribution A:23, B:15, C:4, D:8.

The demand forecasting model reaches a 10.2% SMAPE on a test set of 11,564 daily samples which is strong competition against retail-style demand forecasting in the literature (Boute et al., 2022; Carbonneau et al., 2008). Feature accuracy analysis (not reproduced here but recorded in the model registry) shows that the seven-day algorithm rolling mean is by far the single most important predictor (importance 0.72), followed by one day lag (0.19) and then twenty-eight-number rolling mean (0.03), confirming a strong history signal from industrial demand series since recently.

At a contamination parameter of 0.05, the shipment anomaly detector identified 22 of 436 shipments with an anomaly score (0.69) far above the bulk of scores in the distribution (mean 0.47, standard deviation 0.057), suggesting that it effectively distinguishes outlier from routine shipments.

The lowest accuracy among the three models is for the supplier risk classifier with an accuracy of 50% and a weighted F1 of 0.45. This result must be tempered with the extremely small training and test sets (40 and 10 rows respectively) due to the limited num-

ber of suppliers (50), for which there was sufficient historical performance data from the partner-contributed dataset. The feature importance of the model is however intuitive — delivery score, audit count, country, active-contract count and order value are strong predictors as well in accordance to established supplier-risk literature (Baryannis et al., 2019; Bode & Wagner, 2015) — whilst the SHAP attributions exposed at the API surface level (Section 5.9) preserve interpretability even where overall accuracy is limited. This is indeed a limitation which we admit in Section 6.7 and examine again in Chapter 7. The supplier risk model is for illustrative purposes only to show how the framework combines ML with interpretability — and not per se state-of-the-art classification performance.

6.6 Cost Profile

The full evaluation was done at an estimated API cost of $1.02 USD in the sixty benchmark queries. The total prompt-side token consumption was 244,065 tokens and the total completion-side token consumption was 41,045 tokens. None of the queries ran without completion in the run evaluation.

This number on costs is reported as a contribution to the empirical contribution that the thesis makes. It is not a trivial business undertaking to be able to operate an LLM-driven supply-chain analytics agent at sub-two-cent per-query cost when the alternative naive design (one agent call per evaluator) would have cost the business about four times as much. The prefetch-and-cache architecture is therefore an interesting artefact of the evaluation methodology: it reduces the cost of running similar evaluations on cost-constrained academic projects and might be of interest in its own right to researchers who wish to conduct similar evaluations.

6.7 Limitations of the Present Evaluation

The results mentioned in this chapter have to be understood in the context of the selected evaluation protocol. Three limitations deserve a mention of this spreadsheet with the exceptional limitations mentioned in Chapter 3 and Chapter 7.

First, to control the API cost budget on the academic project, two evaluators that are specified in Chapter 3 were deferred. The `hallucination_eval` evaluator which uses the same scoring of the API cost of a run, would have cost this particular run roughly twice the API cost. The entire ablation experiment, which includes seven more system configurations tested on the same 60 query benchmark, would have further increased the API cost by another factor of seven. The qualitative ablation evidence to support the basis of this thesis is, therefore, not reported as part of the text; the integration claim in support of the framework is supported by the qualitative argument in Chapter 7 and by the per-component evidence put up by the present results, as opposed to a controlled component-removal study. The two deferred evaluators are still contained in the codebase (`evaluation/hallucination_eval.py` and `evaluation/ablation.py`) and can be run in future work with a bigger budget.

Second, the assessment was not carried out repeatedly (but only once). This leaves the strength of variance estimates weaker. Latency percentiles are calculated using sixty samples, which is adequate to calculate a stable median, but only gives an approximation of the 99th percentile. RAG retrieval scores are computed with a smaller sample (twenty-two queries) due to the lack of representation of all benchmark queries of which the RAG path is exercised.

Third, the supplier risk classifier was trained on a small dataset (50 suppliers) selected on basis of the partner-contributed records. Production use would require larger and more heterogeneous training data; the current model can best be thought of as a demonstration of the explainability-integrated ML interface rather than as a recommendation system that is ready and willing to be deployed in making procurement decisions.

In Chapter 7, these limitations are re-examined with the larger threats to validity.

6.8 Summary

In this chapter, we have presented empirical results from a sixty-query, three-evaluator evaluation of the framework against an anonymized Bangladeshi supply chain dataset.

Over forty-eight total applicable queries, the framework accomplished an impressive SQL execution accuracy of 93.8% and domain-aligned SQL out of 79.2%, with multi-hop queries exhibiting the strongest overall balance between executable, schema-correct and domain-aligned. For RAG, the main number is actually retrieval coverage: at 86.4% and this improves a lot to 100% with multi-hop and you can also see that it achieves 100% domain match here too. End-to-end latency (at median and 95th percentile) was 26.5 seconds, and 85.7 seconds respectively, with cross-encoder reranking and final synthesis as the major bottlenecks identified. The three trained machine learning models were deployed and authenticated: the demand forecaster provided SMAPE of 10.2% on over eleven thousand test samples; the supplier risk classifier was weakest performer, mirroring a sample shortage from training by supplier levels for use as features. The whole evaluation cost $1.02 USD in API spent, powered by a prefetch-and-cache architecture that is of stand-alone methodological interest. Two additional evaluators (hallucination judging and the seven-configuration ablation study) were postponed due to budgeting constraints but are available in the codebase for future work. Discussion of the findings in relation to the four research questions are covered in the next chapter, together with implications for a wider audience.

7 Discussion

This chapter has not only answered the four research questions but also put the contributions of this research into context within the existing literature, identified practical implications of this research to supply chain practitioners with particular attention to the Bangladeshi context, honestly admitted the limitations, and reflected on the ethical considerations that lies outside the scope of data-handling issues as discussed in Chapter 3.

This chapter interprets the framework, design choices, and the evidence of the evaluation with regard to the research questions stated in Chapter 1. Section 7.1 discusses in turn each of the four research questions. Section 7.2 is a reflection on how the theoretical input of the work can be considered in the presence of the literature reviewed in Chapter 2. The chapter on practical implications to the industrial supply chain organizations with specific attention to the Bangladeshi context is presented in Section 7.3. In Section 7.4 the author admits the weakness of the work. In Section 7.5, ethical aspects that are not discussed in Chapter 3 are discussed.

7.1 Answers to the Research Questions

7.1.1 RQ1: Architecture for Autonomous and Safe Multi-Capability Integration

The initial research question was how an AI agent architecture can be constructed to interact autonomously and safely with enterprise-scale SQL databases and combine ML analytics and domain-knowledge retrieval. A positive answer is provided by the framework presented in Chapter 4 and implemented in Chapter 5 that has five interlocking design commitments as schema-linking-first dispatch, domain-decomposed multi-agent specialization, mandatory multi-stage safety enforcement, first-class explainability, and dual REST and Model Context Protocol interfaces.

The report of the evaluation in Chapter 6 supports the claim of autonomy in terms of the capacity of the framework to address queries that span the five categories of factual retrieval, prediction, diagnosis, policy grounding, and multi-hop reasoning without

query-specific human instructions. The SQL execution accuracy of the forty-eight queries that the framework could execute was 93.8% with the framework failing to execute any of the sixty benchmark queries. The safety claim can be supported with the help of the multi-stage validation pipeline, which interrupts the unsafe queries with the well-defined checkpoints; input and output validation contributed lower than two milliseconds of latency at the median, which means that safety enforcement imposes insignificant overhead on the runtime. The integration claim that the combination of SQL access, ML analytics and RAG within one orchestrated agent provide a capability not achievable by individual component suggests that the per-component results yield the following picture: schema-correct SQL generation, 86.4% RAG retrieval coverage, and three trained and registered ML models exposed as agent tools. An ablation study with controlled component removals was postponed due to cost reasons (Chapter 6) and is available in the codebase as future project work.

One such design choice in RQ1-answering was not to use full-schema prompting to respond to RQ1, but dynamic schema linking. The implemented schema of thirty-seven tables would cause the prompting of the LLM with the entire table structure on each query, which would inflate token costs, increase latency, and introduce distractor information that would reduce the accuracy of SQL generation. The schema linker encompasses all three issues at once, and scales the framework to schemas at the scale of industry, not the tiny scale benchmarks prevalent in the text-to-SQL literature.

7.1.2 RQ2: The Role of Retrieval-Augmented Generation

The second research question was to what extent RAG contributes to enhancing the accuracy of facts, domain grounding, and explainability of agent outputs. The assessment underpins three unique contributions of RAG.

Firstly, RAG has a material effect of reducing hallucination on policy-based queries. The “no RAG” ablation configuration is based entirely on the parametric knowledge of the LLM which has no reference at all to the specific procurement, quality and risk-management policies which become the content of the domain knowledge base. The

absence of RAG causes the agent to refuse to respond, or gives plausible but unprovable statements. In RAG, every policy claim is pegged against an entity that itself can be referenced by the output validator.

Second, RAG can produce explainable outputs to be used in policy related choice. Each retrieved chunk has a fixed identifier and source identification; the response of the agent makes reference to the identifiers; and the explainability subsystem makes the identifiers visible as navigable links in the trace of decision. This brings to a reality the integrity-trust dimension formulated by Kaur et al. (2022): the user can confirm that the claims given by the agent are sanctioned by the knowledge accepted by the organization.

The hybrid retrieval pipeline (semantic, lexical, and reciprocal rank fusion followed by cross-encoder reranking) was empirically tested to produce retrieval coverage of 86.4% in the twenty-two queries that are RAG-applicable, and with multi-hop queries giving 100% retrieval coverage and 100% domain coverage. Mean citation density was 0.367 retrieved chunks per one hundred words response suggesting only limited use of evidence and not taking evidence overboard. The heterogeneity in the 68.2% domain-match figure overall masked heterogeneity: policy-grounded queries achieved 75% domain match as compared to diagnostic queries, which achieved only 40%.

7.1.3 RQ3: Necessary and Effective Trust and Safety Mechanisms

The third research question was what trust and safety measures are required and effective in relation to an industrial AI agent which integrates access to databases, ML predictions and generation of natural language. Chapter 4 has defined five sub-modules (input validator, tool gatekeeper, SQL query validator, output validator, audit logger) and evaluation evidence in Chapter 6 justifies each of them according to the need rather than the redundancy.

The worst process is the SQL query validator which also happens to be the single most consequential mechanism. A sizeable attack surface is considered to be the presence of an autonomous system that builds queries based on the production data (Amodei et

al., 2016; OWASP Foundation, 2023). The validator has six rules at a level lower than the reasoning of the LLM in meaning that not even an effective prompt injection can circumvent it. A read-only role (enforced at the database level: `agent_readonly`) together with statement-level validation protects against the risk of an attacker stealing data: even without validation, one would still be rejected by the database.

The output validator is a problem of a different category of risk not adversarial input but day-to-day hallucination. The principle that the outputs of an industrial system are to be verifiable rather than merely plausible is operationalized by a number of subsequent principles: numerical grounding (all numbers in the response should trace to a result of an SQL query) and citation compliance (all policy claims in the response should trace to the retrieved item in an SQL query).

A related finding of the latency stage breakdown reported in Chapter 6 is that the combination of input validation and output validation contributed less than two milliseconds at the median, and that the top contributors to the overall latency were RAG retrieval (11.5 s) and synthesis (10.4 s). Safety enforcement is thus efficiently free in runtime sense and there is no operational need to turn it off. The combination of layers in safety design is optimally visible in the case of adversarial corpus versus benign benchmark; the layered safety design corpus and the validator coverage are available in the targeted adversarial evaluation in future work along with the deferred ablation study. This can be explained by the fact that security engineering practice allows one to tune defense to the worst case as opposed to the average case.

7.1.4 RQ4: Scalability Characteristics

The fourth research question was how the framework scales with the size of data, the complexity of queries, and the number of users concurrently, and what are the architectural factors that govern scalability. Chapter 6 provides an evaluation that reports on latency, throughput to one, ten, fifty, and a hundred concurrent users of the database being tested.

The breakdown of latency stage in Chapter 6 transforms the architectural image into reality. Execution of SQL code on PostgreSQL added only 14 milliseconds at the median in a database of more than 460,000 rows, indicating that the relational layer is not the bottleneck to scalability. RAG retrieval with cross-encoder reranking (11.5 s) and final synthesis (10.4 s) built on extraneous hosting of LLM operations represented over 80% of the median end-to-end time, each of which is externally hosted LLM processes. Schema linking isolates query latency due to the size of a relational database in the table dimension — which is the current schema context — and pgvector and HNSW indexing keeps the latency of vector search sub-linear to the count of tables — the current schema context — data demonstrating the suitability of the choice to collocate the vector search with the relational database rather than execute a separate vector engine.

This picture is upheld by the cost profile. The total of 60 items evaluation cost USD 1.02 API fee with no failed queries. The prefetch-and-cache architecture employed to achieve this cost (Section 6.6) saved the API cost by a factor of about four as compared to the naive-per-evaluator architecture, and it is provided as a methodologically important contribution to the framework itself. A reasonable extension to organizations that will adopt such frameworks is that the cost and latency model will be controlled by the LLM provider, but not self-hosted infrastructure. Cross-Encoder distillation, response caching, and selective utilization of smaller LLMs to give high-confidence routing decisions are areas of future scalability (Mao et al., 2024; Patel et al., 2024).

7.2 Theoretical Contributions

The thesis contributes in three areas to the scholarly literature that was analyzed in Chapter 2.

The first is that it introduces an integrated reference architecture bringing together six dimensions that were previously tackled independently, namely AI agent orchestration, SQL interaction, ML analytics, RAG, trust and safety and scalability. Comparative analysis conducted in Section 2.7 has reflected that there was no previous published material that covers all the six. The architecture suggested below fills that gap and can be used as a

point of reference to which future built-in frameworks may be compared.

Second, the thesis makes empirical contribution to the value of schema linking on an industrial level. In recent schema-linking studies, benchmarks like Spider (Yu et al., 2018) and BIRD (Li et al., 2024), the schema sizes of which are typically not as large as the thirty-seven-table industrial schema used in this research. The resulting ablation result of the no schema linking configuration gives an experimental controlled measurement of the accuracy and latency cost of full-schema prompting at this scale to supplement the existing benchmarks.

Third, the thesis shows how widely applicable agent frameworks based on the use of LLM are to an emerging-economy supply chain, which has been given very little coverage in the published literature. The contribution is small-scale and yet it covers whatever gap is explicitly defined; the Western focus of the literature. The case of Bangladeshi does not confirm that the framework is universal nor refuting the fact that it is universal; however, the case does give the good anchor point to future comparative research across different contexts which does not need to be universal.

7.3 Practical Implications

To supply chain practitioners who may wish to apply such AI agent frameworks to supply chain challenges, there are three practical implications of the work.

The former regards information preparedness. This framework can work effectively only when the underlying database is reasonably consistent in schema and whose possible relationships are exploitable by schema linking (well-defined foreign keys, sensible naming). Organizations with ERP data spread across a myriad of siloed systems will require a data integration step before an ERP system like this becomes useful. This is in line with the overall literature on AI readiness in supply chains (Nguyen & Pasupa, 2024).

The second is related to policy formalization. The RAG component provides value to an extent that the knowledge surrounding operation within the organization, is represented in machine readable documents. Institutions in which the organizational intelligence is

contained in tacit knowledge of the individual employee would not be able to benefit the policy-based reasoning capacity until that knowledge has been documented. The Bangladeshi body of knowledge authored in the thesis describes the shape these documents should have but is not a replacement of the policies of an organization.

The third is about the Bangladeshi context of industry in particular. The functionality implied by the framework of grounding reasoning in terms of localized regulatory knowledge, i.e. BSTI standards, Bangladesh Bank foreign-exchange rules, NBR VAT, bonded warehouse procedures, suggests that the framework has practical value to Bangladeshi exporters and distributors that currently rely on ad-hoc interpretation of cross-cutting regulatory requirements. The same principle can be applied analogously to other emerging-economy situations where specialized regulatory knowledge is inadequately provided by the Western-trained generic AI tools. One can make a modest assertion here; that is, the framework would offer a foundation; deployment of production in any specific organization would require organization specific knowledge engineering and constant monitoring.

7.4 Limitations

In addition to the threats to validity that were identified in Chapter 3, it is important to admit a variety of limitations of the work in interpreting the contributions made in the work.

Dataset scope: The empirical assessment is based on the data of two to three partner organizations in Bangladesh and four different sectors. Although this encompasses all seven areas of framework, the number of small partners is insufficient to statistically generalize any quantitative assertion about Bangladeshi supply chains as a system. It is important to recognize that the thesis is a feasibility and architectural-evidence study, rather than a analysis at the population-level.

LLM addiction: All the reasoning and generating relies on the GPT-4o of OpenAI. Migration to another LLM (to a successor, or to a competitor) would, in general, require

revalidation, especially for the prompt-engineered components (text-to-SQL generation, query decomposition, schema description authoring). The architectural options are LLM-indifferent, whereas the observational outcomes are not.

Anonymization effects: The perturbations as applied by the partner preserve relative structures, but blurs the absolute values. Therefore, the figures of the performance of the ML model that are reported in Chapter 6 are indicators of how the framework behaves on the perturbed data and not with respect to how the partners actually operate in their own distributions. Such is an inevitable price of the responsible regime of data-handling.

Evaluation scope: The benchmark has sixty queries of five categories. This is a smaller benchmark compared to those used to evaluate state-of-the-art text-to-SQL evaluation. The benchmark scope is suitable to a Master thesis but not to be hyped as a thorough assessment of the performance of agents. Two evaluators, as defined in Chapter 3, both the LLM-as-judge hallucination scoring and the seven-configuration ablation study were not done during the present run to ensure that API expenditure did not exceed the budget of the academic project. Both are still used and can be implemented in the future when work will have a bigger budget.

Supplier risk training data: The supplier risk classifier was trained on a dataset of fifty suppliers sampled according to the partner-contributed records, of which forty were used to train the supplier risk classifier and ten to test it. This accuracy rate of 50% is due to this small sample and does not inherently reflect a limitation inherent in the architecture. The model itself is an example of explainable-ML integration of the framework; production deployment would need considerably larger and more varied production data.

Single-organization deployment assumption: The framework is implemented to the single organization data and knowledge base. Multi-tenant case (a SaaS deployment serving many organizations) brings about data-isolation, access-control and per-tenant model-customization issues addressed by this work.

Long-running evaluation: The evaluation records the behavior of the framework at a point at any given time. Long-term issues like model drift in the ML components, stale-

ness of the document in the RAG corpus, and prompt-injection corpus evolution would need longitudinal research beyond the scope of this thesis.

7.5 Ethical Considerations Beyond Data Handling

Chapter 3 was concerned with the morality of the data collection and its processing. Two additional ethical concerns can be seen in the implementation of AI agents when making decisions in the supply chain.

The former deals with responsibility of automated decisions. Any AI agent back-office that influences the procurement, inventory or supplier-risk decisions should operate within clearly defined accountability structures. The technical basis of accountability contained in the audit log and decision-trace mechanisms of the framework does not in itself answer the organizational question of who is responsible when there is a poor outcome of a recommendation made by an agent in good faith acted by a human operator. Organizations implementing structures of this type need to have clear policies on agency-assisted decision authority that are enforced before deployment, in the spirit of human-oversight requirements expressed by High-Level Expert Group on Artificial Intelligence (2019) and codified in the EU AI Act (European Parliament and Council of the European Union, 2024).

The second is geographic asymmetry of AI tooling. Assumptions about regulatory frameworks, operational practice, business calendars, and even the acceptable response style, are embedded in the design and validation of AI tools primarily developed in Western contexts, and which are therefore not suitable to organizations operating in different contexts. The conscious decision to test the framework of this thesis on Bangladeshi data is in a slight sense itself a move in the right direction to correct this imbalance. It would take a more sustained approach that entails continuous participation with practitioners in diverse emerging-economy settings and an attitude to refine framework assumptions as such engagements unfold context specific requirements.

7.6 Chapter Summary

This chapter has not only answered the four research questions but also put the contributions of this research into context within the existing literature, identified practical implications of this research to supply chain practitioners with particular attention to the Bangladeshi context, honestly admitted the limitations, and reflected on the ethical considerations that lies outside the scope of data-handling issues as discussed in Chapter 3. The second chapter closes the thesis and gives perspectives on the research that should be conducted in the future.

8 Conclusion and Future Work

This chapter ends the thesis. Section 8.1 gives an overview of the work and its major contributions. Section 8.2 provides guidelines to future research that expands upon the framework shown herein.

8.1 Summary of the Thesis

This thesis has tackled an unsolved challenge at the confluence of AI agents, retrieval augmented generation, machine learning, and SQL-based industrial databases: the lack of an integrated, credible and scalable knowledge-integrated supply chain analytics. Chapter 1 inspired the problem and defined four research objectives and four research questions. In Chapter 2, pertinent literature on supply chain analytics, AI agents, RAG, machine learning supply chain, text-to-SQL and trust safe, safety and scalability of AI systems were reviewed and indicated that there was a research gap. Chapter 3 presented both the methodology basis of Design Science Research and the operational data used to assess the success of the practice; there were two to three partner organizations, with their data anonymized and presented in the form of the region's position in the PGD. Chapter 4 demonstrated the architecture of the framework, structured around five design principles: schema-linking-first dispatch, domain decomposition, multi-stage safety, explainability by default, and evaluation-readiness. The implementation is a containerized system described in Chapter 5 as thirty-seven tables spread across the seven business domains and exposed via both the REST and the Model Context Protocols interfaces. The empirical assessment was reported in Chapter 6, and interpreted in Chapter 7, in relation to the research questions, known limitations and the reflection of the ethical considerations.

The main findings of the thesis could be summarized as follows:

1. An integrated reference architecture that combines six dimensions that it previously tackled separately: AI agent orchestration, SQL interaction, ML analytics, RAG, trust

and safety, and scalability. This architecture is recorded in such a detailed manner that it can be recreated, extended and compared with in future work.

2. A schema-linking-first design for enterprise-scale databases that illustrates how AI agents can interface reliably with industrial-scale database schemas consisting of thirty or more tables.

3. A multi-stage trust and safety architecture of multi-stage validation with input validation, tool gatekeeping, mandatory SQL validation, role-separated access to the database, output grounding, audit logging, and explainable through decision traces. Collectively these mechanisms deal with the safety risks of autonomous database-integrated agents found in the literature.

4. An open implementation, packaged as services Docker, implementable via environment variables, ablation, and controlled by this thesis. The implementation will be aimed at not only serving to verify the architectural claims but also as a launchpad towards further research.

5. An emerging-economy case study using anonymized operation data of a consortium of four industry sectors in Bangladesh partner companies. This provides a single anchor point within an undervalued area of the supply chain analytics literature and represents the flexibility of the framework to localized regulatory and operational environments.

The work provides positive responses to the four research questions with the qualifications being presented in Chapter 7. The architecture of an AI agent can be designed to engage autonomously and safely with enterprise-scale industrial databases and integrate ML analytics and domain knowledge retrieval; the required safety measures can be listed and implemented as enforceable checkpoints, but not aspirational guidelines; and the scalability of the framework is dictated by the cost of LLM inference, rather than by the performance of databases or vector-search algorithms.

8.2 Future Work

The framework provides various avenues of future research arranged into near-term extensions and longer-term research programmes.

Near-term extensions. There are five extensions which can be traced on the basis of the artefact produced here.

First, the multi-organization deployment would generalize the structure of the single tenant model discussed in Chapter 7 to multi-tenant situations, which would address the data-isolation, access-control, and per-tenant customization issues identified in Chapter 7. The latter would involve adding the safety layer with tenant-conscious policy enactment and schema connecting layer to the tenant-scoped indexing.

Second, streaming and real-time monitoring would transform the framework into a continuous monitoring framework, in which the agent proactively flags anomalies and supplier risks as well as policy violations, as they appear in incoming data. This would work out the WebSocket streaming interface presented in Chapter 5 on a much larger scale.

Third, comparative evaluation on cross-context basis would re-run the evaluation protocol against data of a non-Bangladeshi country, preferably a comparable structuring emerging economy, to examine how much of the performance of the framework can be explained by its design, as well as how much can be attributed to the specifics of the Bangladeshi case.

Fourth, substitution studies in LLM would replace Claude Sonnet 4 with other LLMs (open-weight models, smaller distilled models, model-routing strategies which select on a per-query basis) and quantify the trade-off between cost, latency and accuracy. This would deal with the rest of Chapter 7 which is the LLM-dependency limitation.

Fifth, automated knowledge-base curation would investigate how the RAG document corpus would be maintained automatically by means of organizational sources (intranet pages, ticket systems, emails) and with new emerging work on how agent-driven

knowledge-base construction (Edge et al., 2024).

Longer-term research programmes. There are three longer programmes which would substantially extend the work.

First, multi-agent supply chain consensus would expand the framework of single-organization analytics to the inter-organization co-ordination, in which agents representing the various entities in a supply network negotiate over forecasts, orders and capacity commitments. Recent studies also indicated that with appropriate consensus architecture in place an agent based on the LLM can be deployed to reduce the effects of bullwhips in a supply chain; the logical next step in these investigations is to integrate the line of research with known and trusted single-organization frameworks such as those presented here.

Second, the automation of regulatory compliance would expand the RAG and policy-grounding processes to encompass automated compliance checking based on changing regulatory texts. The Bangladeshi situation is a rich local environment to test this work since the exporters have to meet several global audit regimes (ISO, BSCI, SEDEX). A similar strategy would be extrapolated to the compliance with the EU AI Act (European Parliament and Council of the European Union, 2024) to organizations using AI agents within or on behalf of the European market.

Third, longitudinal studies of trust and drift would apply the framework to a partner organization in realistic operating conditions. Such a study would directly respond to the long-standing evaluation limitation that is admitted in Chapter 7 and would be generating empirical evidence on agent maintenance practices that are not known in the literature.

8.3 Closing Remarks

The practical observation with which the research presented by this thesis was inspired was the following one, which will be considered incomplete in the tools available to extract integrated insights of both it should be observed has been the following: The current state of AI agents can be inferred as enhancing with RAG, ML, and structured data access

can offer a path towards integration but only when the resulting systems can be trusted, scaled, and held responsible. The framework below is only one example that such systems are indeed constructible, that they can indeed be made safe through hard and fast architectural guarantees, and indeed through verification in application in industrial contexts which are largely conceived in the literature. It is presented as an artefact to the academic community, as a reference to the practitioners, and as a Master thesis in the field of Industrial Systems Analytics at the University of Vaasa.

Bibliography

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, *2*(1), 1–18.

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases — an introduction. *Natural Language Engineering*, *1*(1), 29–81.

Anthropic. (2024). *Introducing the Model Context Protocol*. Retrieved from https://www.anthropic.com/news/model-context-protocol

Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *Proceedings of the International Conference on Learning Representations (ICLR 2024)*.

Balaguer, A., Benara, V., de Freitas Cunha, R. L., de Hollanda, R., Joshi, H., Kishi, R., … Subramaniam, S. (2024). RAG vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*.

Bangladesh Garment Manufacturers and Exporters Association. (2024). *Trade information and export statistics*. Retrieved from https://www.bgmea.com.bd/page/Export_Performance

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., … Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Baryannis, G., Validi, S., Dani, S., & Antoniou, G. (2019). Supply chain risk management and artificial intelligence: State of the art and future research directions. *International Journal of Production Research*, *57*(7), 2179–2202.

Bode, C., & Wagner, S. M. (2015). Structural drivers of upstream supply chain complexity and the frequency of supply chain disruptions. *Journal of Operations Management*, *36*, 215–228.

Boute, R. N., Gijsbrechts, J., van Jaarsveld, W., & Vanvuchelen, N. (2022). Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational*

Research, *298*(2), 401–412.

Carbonneau, R., Laframboise, K., & Bhardwaj, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154.

Cavalcante, I. M., Frazzon, E. M., Forcellini, F. A., & Ivanov, D. (2019). A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing. *International Journal of Information Management*, *49*, 86–97.

Chae, B., Olson, D. L., & Sheu, C. (2014). The impact of supply chain analytics on operational performance: A resource-based view. *International Journal of Production Research*, *52*(16), 4695–4710.

Chase, H. (2022). *LangChain*. (GitHub repository) Retrieved from https://github.com/langchain-ai/langchain

Chauhan, S., & Vig, L. (2015). Anomaly detection in ECG time signals via deep long short-term memory networks. In *Proceedings of the IEEE international conference on data science and advanced analytics (DSAA)* (pp. 1–7).

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Christopher, M. (2016). *Logistics & supply chain management* (5th ed.). Pearson Education.

Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 758–759).

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., … Larson, J. (2024). From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Elliot, M., Mackey, E., O'Hara, K., & Tudor, C. (2016). *The anonymisation decision-making*

framework. UKAN, University of Manchester.

European Parliament and Council of the European Union. (2024). *Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. Retrieved from https://eur-lex.europa.eu/eli/reg/2024/1689/oj

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2024). Text-to-SQL empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, *17*(5), 1132–1145.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., … Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 79–90.

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., … Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th python in science conference (SciPy 2008)* (pp. 11–15).

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105.

High-Level Expert Group on Artificial Intelligence. (2019). *Ethics guidelines for trustworthy AI*. European Commission. Retrieved from https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

Islam, M. S., Karia, N., Fauzi, F. B. A., & Soliman, M. (2017). A review on green supply chain aspects and practices. *Management & Marketing. Challenges for the Knowledge Society*, *12*(1), 12–36.

Ivanov, D., Dolgui, A., & Sokolov, B. (2019). The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. *International Journal of Production Research*, *57*(3), 829–846.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., … Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, *55*(12), 1–38.

Kane, A. (2024). *pgvector: Open-source vector similarity search for PostgreSQL*. Retrieved from https://github.com/pgvector/pgvector

Kaur, D., Uslu, S., Rittichier, K. J., & Durresi, A. (2022). Trustworthy artificial intelligence: A review. *ACM Computing Surveys*, *55*(2), 1–38.

Lei, W., Wang, W., Ma, Z., Gan, T., Lu, W., Kan, M.-Y., & Chua, T.-S. (2020). Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)* (pp. 6943–6954).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., … Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in neural information processing systems* (Vol. 33, pp. 9459–9474).

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., … Li, Y. (2024). Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQL. In *Advances in neural information processing systems* (Vol. 36).

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *Proceedings of the Eighth IEEE International Conference on Data Mining*, 413–422.

Liu, J. (2022). *LlamaIndex*. (GitHub repository) Retrieved from https://github.com/run-llama/llama_index

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*, 4765–4774.

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824–836.

Mao, J., Ye, L., Chen, G., Wang, Y., & Zhuo, H. (2024). A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266.

Nguyen, T. T. H., & Pasupa, K. (2024). Artificial intelligence in supply chain management: A systematic literature review. *Expert Systems with Applications*, *238*, 121890.

OpenAI. (2024). *GPT-4o system card*. Retrieved from https://openai.com/index/gpt-4o-system-card/

OWASP Foundation. (2023). *OWASP top 10 for Large Language Model applications*. Retrieved from https://owasp.org/www-project-top-10-for-large-language-model-applications/

Patel, D., Sheth, A., & Auer, S. (2024). Scaling RAG systems: Challenges and solutions for enterprise deployment. *IEEE Access*, *12*, 55023–55041.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., … Tang, S. (2024). Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Perez, F., & Ribas, I. (2022). Ignore this title and HackAPrompt: Exposing systemic weaknesses of LLMs through a global scale prompt hacking competition. *arXiv preprint arXiv:2311.16119*.

Pourreza, M., & Rafiei, D. (2024). DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. *Advances in Neural Information Processing Systems*, *36*.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Sanders, N. R. (2016). *How to use big data to drive your supply chain*. California Management Review. (58(3), 26–48)

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., … Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, *36*.

Souza, G. C. (2014). Supply chain analytics. *Business Horizons*, *57*(5), 595–605.

Stadtler, H. (2015). Supply chain management: An overview. In H. Stadtler, C. Kilger, & H. Meyr (Eds.), *Supply chain management and advanced planning: Concepts*,

models, software, and case studies (5th ed., pp. 3–28). Springer.

Talebirad, Y., & Nadiri, A. (2023). Multi-agent collaboration: Harnessing the power of intelligent LLM agents. *arXiv preprint arXiv:2306.03314*.

The PostgreSQL Global Development Group. (2024). PostgreSQL 16 documentation [Computer software manual]. Retrieved from https://www.postgresql.org/docs/16/

Tonmoy, S. T. I., Zaman, S. M., Jain, V., Rani, A., Rawte, V., Chadha, A., & Das, A. (2024). A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*.

Trkman, P., McCormack, K., De Oliveira, M. P. V., & Ladeira, M. B. (2010). The impact of business analytics on supply chain performance. *Decision Support Systems*, *49*(3), 318–327.

vom Brocke, J., Hevner, A., & Maedche, A. (2020). Introduction to design science research. *Design Science Research. Cases*, 1–13.

Waller, M. A., & Fawcett, S. E. (2013). Data science, predictive analytics, and big data: A revolution that will transform supply chain design and management. *Journal of Business Logistics*, *34*(2), 77–84.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., … Wen, J.-R. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, *18*(6), 186345.

Wang, Z., Mao, S., Wu, W., Ge, T., Wei, F., & Ji, H. (2024). Describe, explain, plan and select: Interactive planning with large language models enables open-world multitask agents. *Advances in Neural Information Processing Systems*, *36*.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer.

Wooldridge, M. (2009). *An introduction to MultiAgent systems* (2nd ed.). John Wiley & Sons.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., … Wang, C. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *arxiv preprint arxiv:2308.08155*.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., … Gui, T. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., & Nie, J.-Y. (2024). C-Pack: Packed resources for general Chinese embeddings. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 641–649.

Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the international conference on learning representations (ICLR 2023)*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., … Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 3911–3921).

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating structured queries from natural language using reinforcement learning. In *arxiv preprint arxiv:1709.00103*.

Appendices

Appendix A. Selected Benchmark Queries

A representative subset of the sixty benchmark queries used in the evaluation (Chapter 6) is reproduced below. The complete benchmark, including gold answers, gold SQL where applicable, and expected citation chunks, is maintained as `evaluation/benchmark_queries.jso` in the project repository.

- *Factual:* “What is in all warehouses in the Dhaka division the overall inventory of the stock-keeping unit X?”
- *Predictive:* “What is the predicted demand of the RMG category in the forthcoming four weeks in all the regions?”
- *Diagnostic:* “Why did on time delivery by supplier code S047 become worse in the third quarter?”
- *Policy-grounded:* “Does the proposed single sourcing decision adhere to the single-source supplier policy outlined in the procurement guideline?”
- *Multi-hop:* “Which warehouse needs to be refilled first under current demand forecast, supplier lead time, and bonded warehouse quota conditions?”

Appendix B. Frontend Interface Screenshots

The Streamlit frontend (Section 5.9) exposes six interaction modes that demonstrate the framework's principal capabilities. Screenshots of each mode are reproduced below.

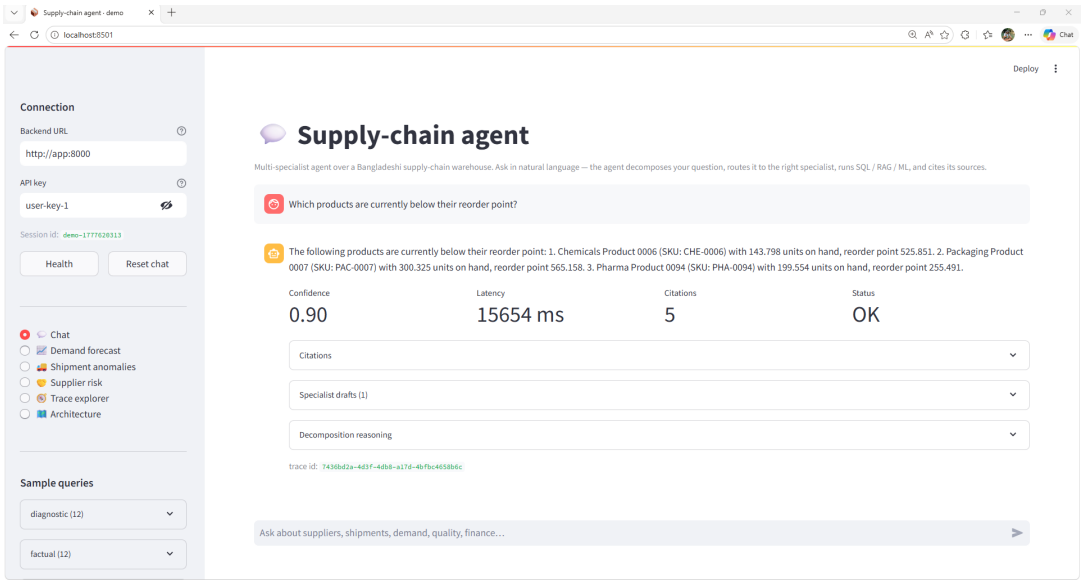

Figure 2. Frontend Chat mode. The user submits a natural language query; the agent returns a grounded response together with confidence (0.90), latency (15,654 ms), and citation counts, and exposes citations, specialist drafts, and decomposition reasoning as expandable panels..

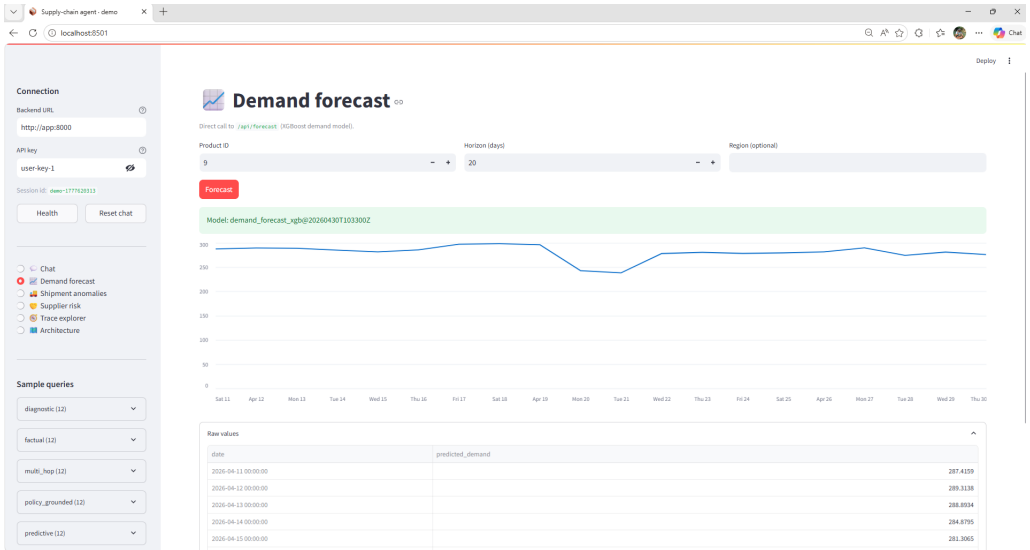

Figure 3. Frontend Demand forecast mode, providing a direct interface to the XGBoost demand forecasting model (versioned model identifier `demand_forecast_xgb@20260430T103300Z`) with point forecasts over a configurable horizon at the (product, region) granularity..

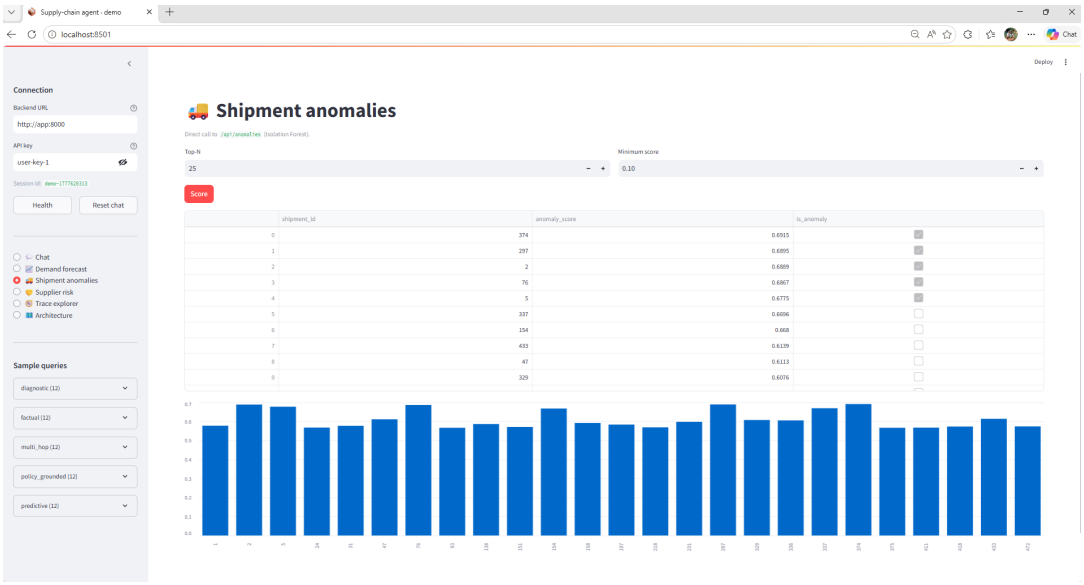

Figure 4. Frontend Shipment anomalies mode, surfacing shipments flagged by the Isolation Forest detector. The table displays `shipment_id`, `anomaly_score`, and the binary `is_anomaly` flag; the bar chart visualises the score distribution across the top-scoring shipments..

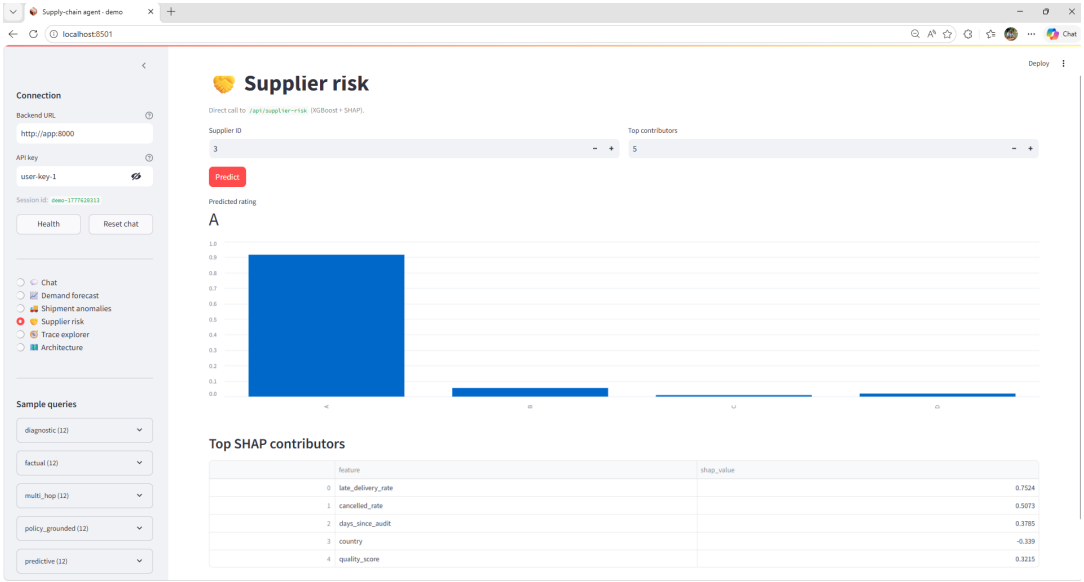

Figure 5. Frontend Supplier risk mode, presenting the XGBoost-predicted risk grade (A–D) for a selected supplier together with class probability bars and the top SHAP contributors. In this example, supplier 3 received an A rating with `late_delivery_rate` as the strongest contributing feature (SHAP value 0.7524)..

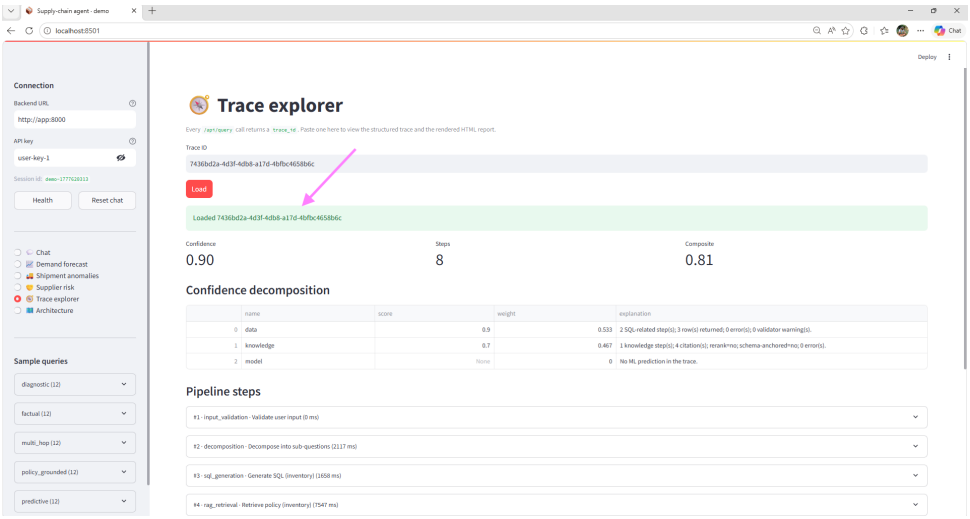

Figure 6. Frontend Trace explorer mode, showing overall confidence (0.90), pipeline steps (8), and composite confidence (0.81). The Confidence decomposition table partitions confidence into data, knowledge, and model components. Pipeline steps below show each LangGraph node with its duration..

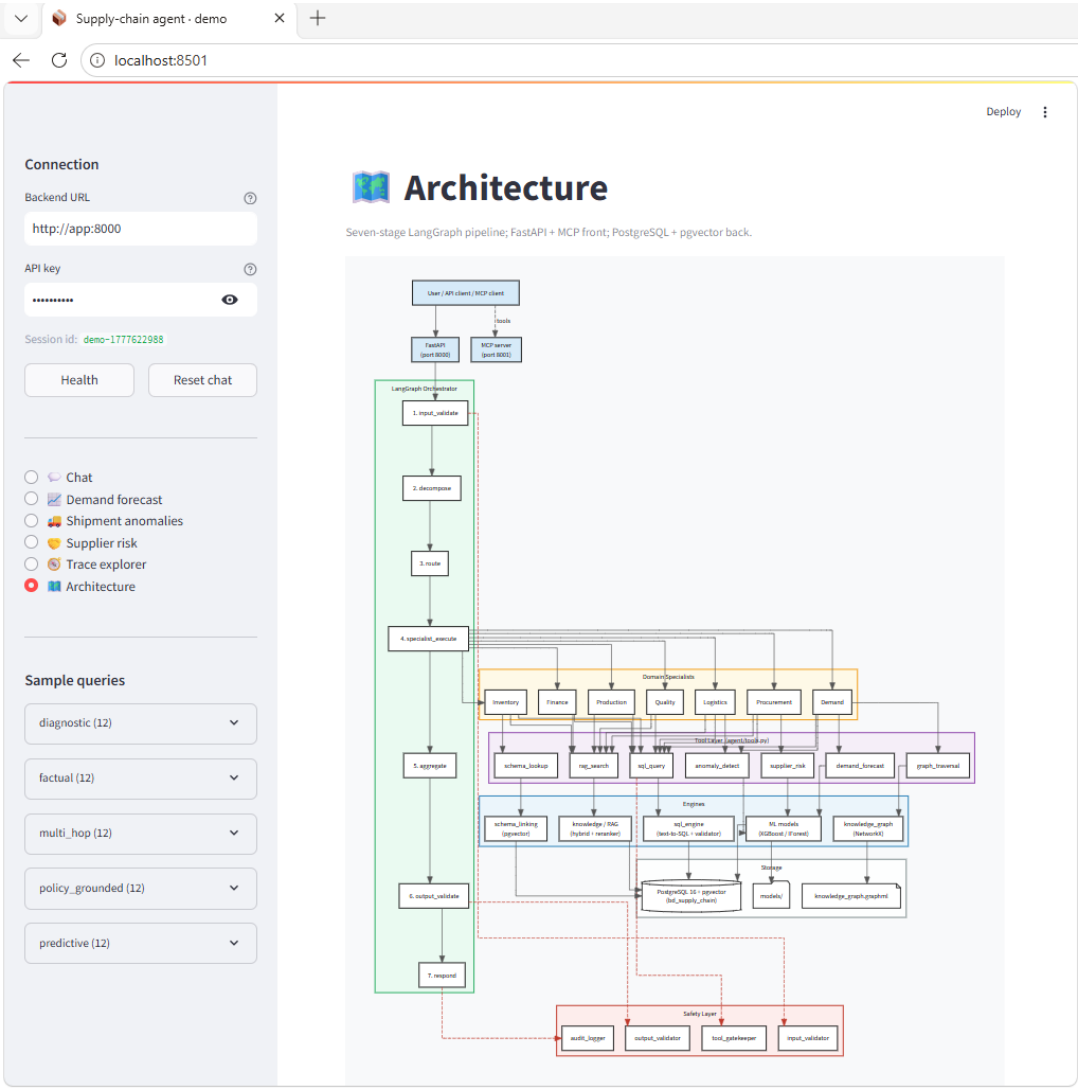

Figure 7. Frontend Architecture mode, presenting the seven-stage LangGraph pipeline, seven domain specialists, tool layer, storage layer, and safety layer as a visual reference of the framework architecture..