Azeem Raza

DESIGN AND IMPLEMENTATION OF AN AI AGENT CUSTOMER SUPPORT IN FINNISH SME'S

School of Technology
2025

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Azeem Raza
Title	Design and Implementation of an AI Agent Customer Support in Finnish SME's
Year	2025
Language	English
Pages	38
Name of Supervisor	Harri Lehtinen

In today's fast-moving business world, customers expect fast, personalized, and multilingual support as a normal part of good service. Big companies often have the tools and budget to use advanced AI to meet these needs. But many small and medium-sized businesses (SMEs) in Finland struggle to find options that are both affordable and flexible especially for handling more than one language. This thesis looks into how to design and build an AI customer support agent made for mobile retail businesses, focusing on the needs of Finnish SMEs.

The system uses the advanced LLaMA3-70B language model to understand and respond to questions in both Finnish and English. It can figure out the purpose of a message whether it is about a product or an order and give useful replies in real time. Tools like LangChain and LangGraph help manage the conversation flow, and the Groq API keeps responses quick. A simple, user-friendly interface was created with Streamlit to make it easy to use. Tests showed that the system helps businesses reply faster and keeps customers happier. While this project focuses on Finnish SMEs, the system can be adapted for other industries and languages, showing how AI can really improve customer service.

Keywords Artificial Intelligence (AI), LLaMA3, LangGraph, Groq API, Streamlit-UI

CONTENTS

ABSTRACT

1 INTRODUCTION 7
1.1 Background and Motivation 8
1.2 Problem Statement 8
1.3 Objectives of the Thesis 9
2 LITERATURE REVIEW 10
2.1 Understanding Artificial Intelligence and Neural Networks 10
2.2 Rise of Large Language Models (LLMs) 11
2.3 Evolution of AI in Customer Support 12
2.4 Research Gap and Thesis Contribution 12
2.5 Comparison with Traditional AI Tools 13
3 DESIGN AND IMPLEMENTATION 14
3.1 Tools and Frameworks 14
3.1.1 LangChain 14
3.1.2 Workflow Orchestration 14
3.1.3 Groq API for Advanced Language Models 15
3.1.4 User Interface Development 15
3.1.5 Python Programming Language 15
3.2 System Architecture Overview 16
3.2.1 Product Functions Workflow 17
3.2.2 Order Functions Workflow 18
3.3 Project Structure 19
3.3.1 Core Technologies 20
3.4 Implementation Details 20
3.4.1 State-Based Workflow for Query Handling 21
3.4.2 Query Routing Using LLM Reasoning 22
3.4.3 Workflow Orchestration with LangGraph 22

3.4.4 Data Retrieving logic 24
3.4.5 Response Generation 24
3.5 Data Schema and Inventory Management 25
3.6 Result 27
4 TESTING AND ANALYSIS 29
4.1 Testing Approach 29
4.2 Analysis 31
4.2.1 Strengths 31
4.2.2 Limitations 31
5 DISCUSSION AND CONCLUSION 33
5.1 Future Work 34
5.2 Acknowledgement of Language Assistance 34
REFERENCES 35

LIST OF FIGURES AND TABLES

Figure 1. Hierarchical Relationship Between AI, ML, DL, and NLP.....11
Figure 2. AI Agent System Architecture.....16
Figure 3. Product Inquiry Workflow Diagram – From Input to Response.....17
Figure 4. Order Inquiry Workflow Diagram – Status Retrieval Logic 18
Figure 5. Project Directory Structure.....19
Figure 6. Initialization of Groq + LLaMA3 and Shared Agent State (AgentState) .22
Figure 7. Router Node Processing Flow – Intent Recognition 22
Figure 8. Conditional Edge for Product Inquiry Routing – LangGraph Logic.....23
Figure 9. Conditional Edge for Order Inquiry Routing – LangGraph Logic 23
Figure 10. Prompt Engineering and Product Info Retrieval Logic.....24
Figure 11. Response Generation from Retrieved Data.....25
Figure 12. Products Inventory Schema Example – JSON Representation.....26
Figure 13. Order Schema Example – Order and Product Linkage.....27
Figure 14. AI Response to English Query: "What is the price of iPhone 15?".....28
Figure 15. AI Response to Finnish Query: "Mikä on iPhone 15:n hinta?".....28
Figure 16. Example of a query test 29
Figure 17. AI Response to Product with Zero Stock.....30
Figure 18. Database Record of Out-of-Stock Product.....30

Table 1. Comparative Analysis of AI Systems: Rasa, Zendesk, and Proposed Solution.....13
Table 2. Core Technologies Used in the AI Agent System 20

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
NLP	Natural Language Processing
SMEs	Small and Medium Enterprises
API	Application Programming Interface
UI	User Interface
LPU	Language Processing Unit
LLM	Large Language Model
LLaMA	Large Language Model Meta AI
ML	Machine Learning
RNN	Recurrent neural network
GPT	Generative Pre-trained Transformer
NLG	Natural Language Generation
FAQ	Frequently Asked Questions
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation (data format used in APIs)
HTTP	Hypertext Transfer Protocol (used for web communication)
REST	Representational State Transfer (API architecture)
GUI	Graphical User Interface

1 INTRODUCTION

In today's highly competitive business environment, customer expectations are higher than ever. Recent studies show that people expect faster, more personalized service than before the pandemic (Hyken, 2024). Good customer service has become a main cause of loyalty. 93% of customers are likely to come back to businesses that give good support, Almost half have changed brands because of bad service (Fontanella, 2024). To meet these growing demands, many companies are now using AI tools to deal with staff and service problems (Brynjolfsson, Li, & Raymond, 2024). However, AI customer support systems can cost between €5,000 and €100,000 a year, making them too expensive for many small and medium-sized enterprises (SMEs) (Rowe, 2025).

This thesis focuses on how AI agents can be used to offer affordable and multilingual customer support for small and medium-sized businesses (SMEs) in Finland. The system was built to help mobile retail customers ask questions about products and orders in both Finnish and English. It uses Natural Language Processing (NLP) based on transformer models first introduced by Vaswani et al. (2017). The language model used is LLaMA3-70B, developed by Meta in 2023, which is accessed through the Groq API to ensure quick responses. To manage the conversation steps, the system uses LangChain and LangGraph, and the user interface was created with Streamlit. This makes it easier for SMEs to use the system without the need of advanced technical knowledge.

The results demonstrate real-time performance with response times under one second, accurate query classification, and smooth multilingual interaction. By combining open-source components in a modular, scalable architecture, this project delivers a practical, affordable AI solution for Finnish SMEs. While the focus is on mobile retail, the system is flexible enough to be extended to other industries and languages. Ultimately, this project supports the broader adoption of AI in

customer service and shows that even smaller companies can benefit from advanced AI technologies (An, 2025; Huseyn, 2025).

1.1 Background and Motivation

Customer support is important for a business to do well, but Small and Medium Enterprises (SMEs) in Finland often struggle due to limited resources and need of communication with customer (Hakola, 2025). She further mentioned that it is getting very hard for the companies to find educated and skilled customer support workers. Therefore, it is important to develop such applications that can help by giving fast and helpful customer support. AI-powered agents offer a solution by automating responses and improving efficiency (Hayes & Downie, 2025). However, many SMEs lack access to affordable, tailored solutions, especially for non-English businesses (Anand, 2024). This thesis aims to close that gap by developing an AI customer support agent designed specifically for SMEs in Finland combining multilingual support and dynamic response generation.

1.2 Problem Statement

Even though customer support is a key part of business success, many SMEs in Finland face real challenges in delivering consistent, high-quality service. Limited staff and resources often make it difficult to keep up with customer needs (Yrittäjät, 2023). One of the biggest struggles is handling customer questions in different languages while still providing fast and accurate responses (Porubská, 2025).

Moreover, many SMEs find it hard to accept new technologies, especially when these tools do not easily fit into their existing ways of working (Gagyi, 2023). This can slow down the things, can cause confusion and leave both the business and its customers feeling frustrated. The need of affordable, flexible, and easy-to-use automation tools that can help SMEs offer better service without needing large teams or big investments (Sellberry, 2024).

This thesis deals with these issues by designing AI-powered mobile retail customer support agent built specifically for Finnish SMEs. The goal is to create a system that can handle multilingual inquiries, understand customer intent and generate helpful responses in real time. The goal is to make customer service better respond faster, save money, and keep customers happier.

1.3 Objectives of the Thesis

The main goal of this thesis is to design and implement an AI-powered customer support agent tailored to the needs of Finnish SMEs.

The system is intended to utilize natural language processing (NLP) techniques, including transformer-based models, to understand and respond to customer inquiries in both Finnish and English. To ensure smooth and scalable operations, the system integrates tools such as LangChain and LangGraph, which help automate and organize customer support workflows. In addition, an easy-to-use interface is developed using Streamlit, allowing small businesses to interact with the AI without requiring technical skills. The project also aims to create a flexible and adaptable framework that can be used across various industries and regions, supporting the broader adoption of AI-powered customer service solutions.

2 LITERATURE REVIEW

This section presents key concepts, previous studies, and theoretical foundations related to AI-powered customer support, focusing on large language models (LLMs), neural networks, and comparison with earlier tools like Rasa and Zendesk. The aim is to show how this thesis builds upon and contributes to existing research.

2.1 Understanding Artificial Intelligence and Neural Networks

Artificial Intelligence (AI) is the ability of machines to do tasks that normally need human intelligence, like understanding language, finding patterns, or making decisions (Stryker & Kavlakoglu, 2024). A key idea in AI is the use of neural networks, which are systems modeled after the human brain. These networks are made up of layers of simple units (called nodes) that pass information to each other. Each layer helps the system learn more about the input data. When there are many layers, this is called deep learning (DL), which helps machines do more complex things like understanding speech or language (Qamar & Zardari, 2023). This is where natural language processing (NLP) comes in. NLP is the field of AI that allows machines to read, understand, and respond to human language in useful ways (Eppright, 2021).

Figure 1 shows the hierarchical relationship between AI, ML, DL, and NLP. This diagram demonstrates how Natural Language Processing (NLP) is a subset of Deep Learning (DL), which is itself part of Machine Learning (ML), all under the broader domain of Artificial Intelligence (AI).

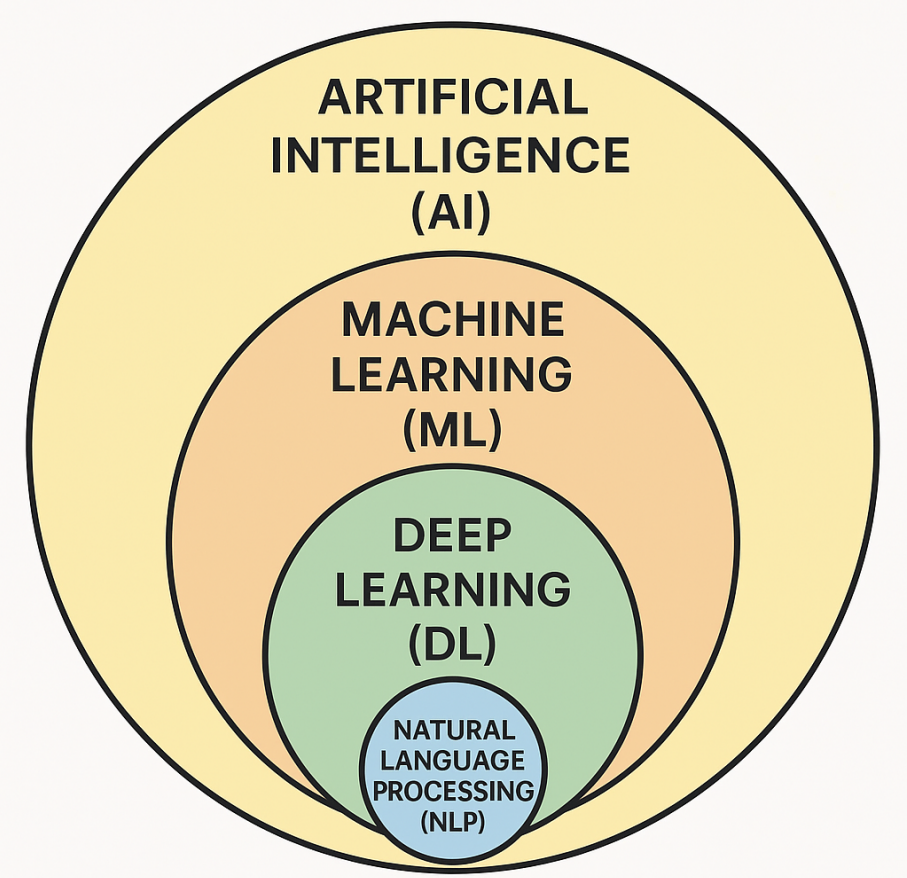

Figure 1. Hierarchical Relationship Between AI, ML, DL, and NLP

2.2 Rise of Large Language Models (LLMs)

Large Language Model (LLM) like LLaMA3 is a significant leap forward in natural language processing. This model is trained on massive text datasets and can understand, translate, summarize, and generate human-like responses. They are built using the transformer architecture introduced by (Vaswani et al., 2017), which allows for handling long-range dependencies in text more efficiently than previous models like Recurrent neural networks (RNNs). LLaMA3, developed by Meta, is among the latest LLMs optimized for performance and cost-efficiency (Grattafiori et al., 2024).

2.3 Evolution of AI in Customer Support

AI agent technologies have improved a lot over the past few decades because of big advances in artificial intelligence (AI), natural language processing (NLP), and machine learning (ML) (Liz, 2025). In the beginning, early AI systems were quite basic. They could automate simple tasks and respond to simple questions. Even though their capabilities were limited these early systems built the base for the more advanced AI agents we see today (Julia, 2024).

As the technology improved, AI agents became much smarter. They started to understand natural language better by making more informed decisions and interact with users in a more context-aware and personalized way. This allowed AI agents to do more than just answer questions. They began handling more complex tasks, offering tailored support, and actively assisting customers in real time (Sridhar, 2024).

2.4 Research Gap and Thesis Contribution

While LLMs are becoming more popular, few studies focus on applying them to Finnish SMEs with limited budgets (Mäntylä, 2024). This thesis fills that gap by combining LLaMA3 with Groq's fast, low-cost API and modular tools like LangChain and LangGraph to build an AI support agent that is fast, with response times under one second, which is important for keeping customers satisfied. It is also affordable because it uses free and open-source tools, making it suitable for small businesses. The system supports both Finnish and English languages, which is especially useful in Finland. Many systems are built in English (Yle, 2020). Lastly, it is flexible and scalable, so it can grow with the needs of the business or be adjusted for different industries. This work contributes to the field by showing how cutting-edge AI tools can be applied in smaller, local business environments and adapted for broader industry needs.

2.5 Comparison with Traditional AI Tools

Earlier AI systems for customer support often relied on rule-based logic bots, such as those built with Rasa. These systems required manual rule writing and language specific training (Muller, 2025). Zendesk AI, a commercial tool, offers good integration and analytics but comes at a high cost (€5,000–€30,000 annually) and lacks flexibility (Pangan, 2025).

Table 1. Comparative Analysis of AI Systems: Rasa, Zendesk, and Proposed Solution

Feature	*Rasa*	*Zendesk AI*	*This Project*
Multilingual Support	Partial (manual)	Available (paid)	Finnish + English built-in
Open Source	Yes	No	Yes
Low Latency	Moderate	Depends	<1s (via Groq API)
Custom Workflow Logic	Manual setup	Limited	LangGraph workflow engine
Easy to Use Interface	CLI-focused	GUI	Streamlit web UI
Real-Time Capable	With effort	Yes	Demo only (static data)
Designed for Finnish SMEs	General use	No general use	Tailored to SMEs
Cost	Free	€5k–€100k/year	Free (open-source stack)

3 DESIGN AND IMPLEMENTATION

The AI customer support agent for Finnish SMEs uses a strong technology stack made for scalability, efficiency, and support for multiple languages. The Natural Language Processing (NLP) features are powered by LangChain and the Groq API, allowing it to sort queries and create dynamic responses. Workflow is managed by LangGraph, which directs queries based on their category. The user interface is built with Streamlit, providing an easy way to interact with the system. Python is the main programming language.

3.1 Tools and Frameworks

3.1.1 LangChain

Compared to alternatives like Rasa or Zendesk, LangChain is a tool that makes it easier to add language models to apps. It helps developers create a series of steps like understanding a question, finding information, and giving an answer (AWS, n.d.). LangGraph adds to this by using a workflow system that follows a set path depending on the kind of question. Together these tools help make the AI system easier to manage, better organized, and able to grow. They also allow new features to be added without changing much of the overall system (LangChain, 2024).

3.1.2 Workflow Orchestration

The system’s workflow is managed using LangGraph, a framework designed for stateful workflow orchestration. LangGraph enables conditional routing based on query category and ensuring each query is managed smoothly. According to LangGraph (2024), "LangGraph provides a modular and scalable approach to building workflows, making it easy to integrate multiple components and adapt to changing requirements". This flexibility helps handle a variety of customer questions and makes it easier to grow the system as needed.

3.1.3 Groq API for Advanced Language Models

The Groq API gives access to powerful language models like LLaMA 3 and GPT-4, helping the system create accurate and context-aware responses. These advanced models can understand difficult questions and write human like answers, which makes them perfect for customer support. One main reason for choosing the Groq API instead of running large language models locally was better performance and efficiency. Running these models locally needs a lot of computing power, which can be expensive and hard to handle, especially for small businesses. Another reason for choosing Groq is that it give response under one second (Bheda, 2025). With Groq, we can use powerful models whenever we need, making sure responses are fast without needing costly hardware (Groq, n.d.).

3.1.4 User Interface Development

The user interface (UI) for the customer support agent is built using Streamlit, a Python-based framework for creating interactive web applications (Miloiu, 2025). Streamlit's simplicity and easy integration make it an ideal choice for SMEs. Alternatives like Flask or React would require more developer effort and are less SME-friendly. According to the Streamlit (2023), "Streamlit allows developers to quickly build and deploy interactive applications, making it perfect for prototyping and production use". This project's UI offers an easy-to-use platform for SMEs to ask questions and see results, improving the overall user experience.

3.1.5 Python Programming Language

The whole system is built using the Python programming language. Python was chosen because it's flexible, easy to read, and has many useful AI libraries. It's simple style and flexibility make it great for building apps quickly and working well with other tools (Teradata, n.d.). According to Python Software Foundation, "Python is an interpreted, object-oriented, high-level programming language with

dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and binding, make it highly attractive for rapid application development".

3.2 System Architecture Overview

The AI-powered mobile retail customer support agent follows a modular and state-driven architecture enabling LangGraph and LLaMA3-70B. The system processes user queries, classifies their intent, retrieves relevant data (products or orders), and generates context-aware responses.

The architecture consists of four main components. The User Input Handler accepts queries submitted through the Streamlit interface. The Intent Classifier determines whether the query relates to products, orders, or general support. The Data Retrieval Module fetches relevant product or order details from structured data sources. Finally, the Response Generator uses the LLaMA3-70B model to generate responses that resemble natural human conversation.

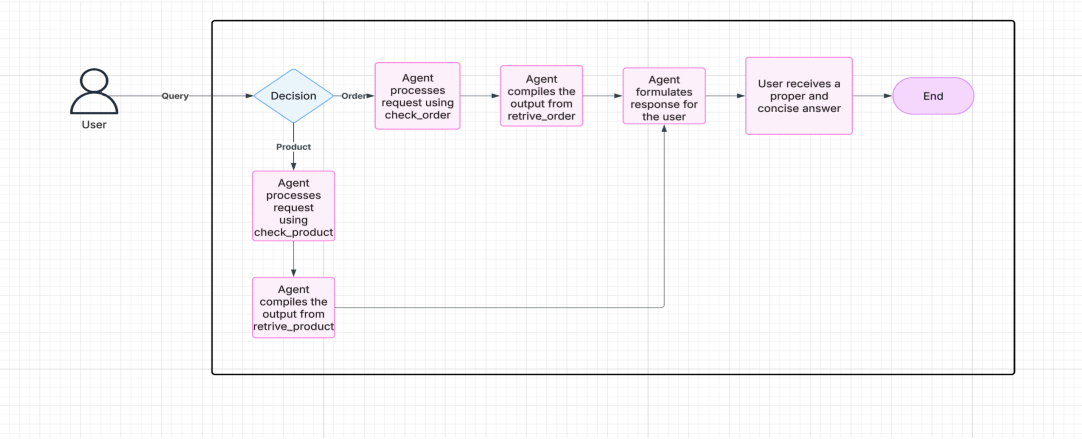

Figure 2. AI Agent System Architecture

3.2.1 Product Functions Workflow

The system first checks if the user's query relates to a product like availability, specifications. The decision logic is shown in below Figure 3:

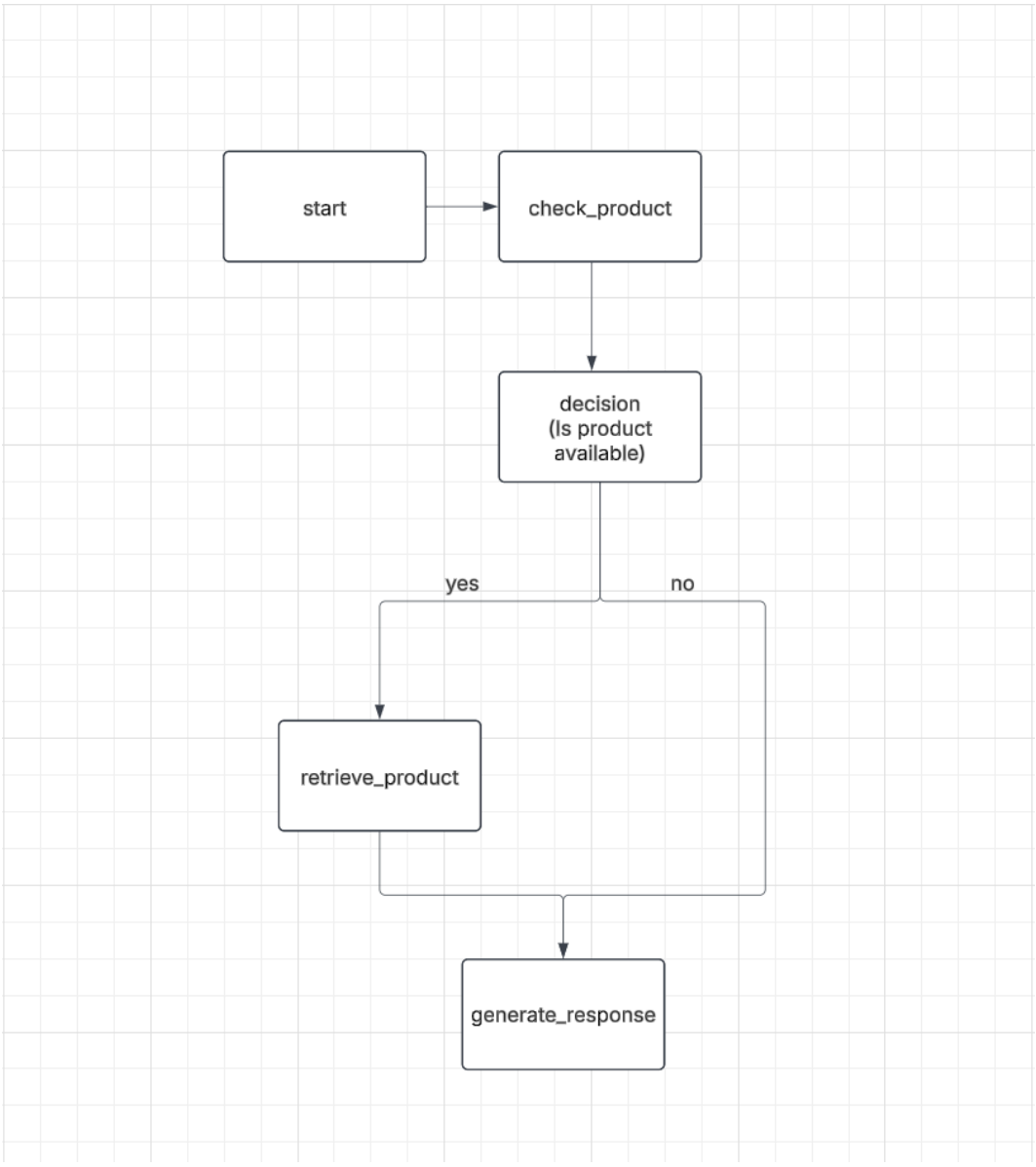

Figure 3. Product Inquiry Workflow Diagram – From Input to Response

The Key Functions are as follows:

should_retrieve_product_info(): Uses LLaMA3 to classify if the query is product-related.

retrieve_product_info(): Extracts the product name like "iPhone 15 Pro" and searches the data.

generate_response(): Produces a structured reply like stock status, pricing.

3.2.2 Order Functions Workflow

If the query is not related to product, the system checks for order-related intent like status, tracking:

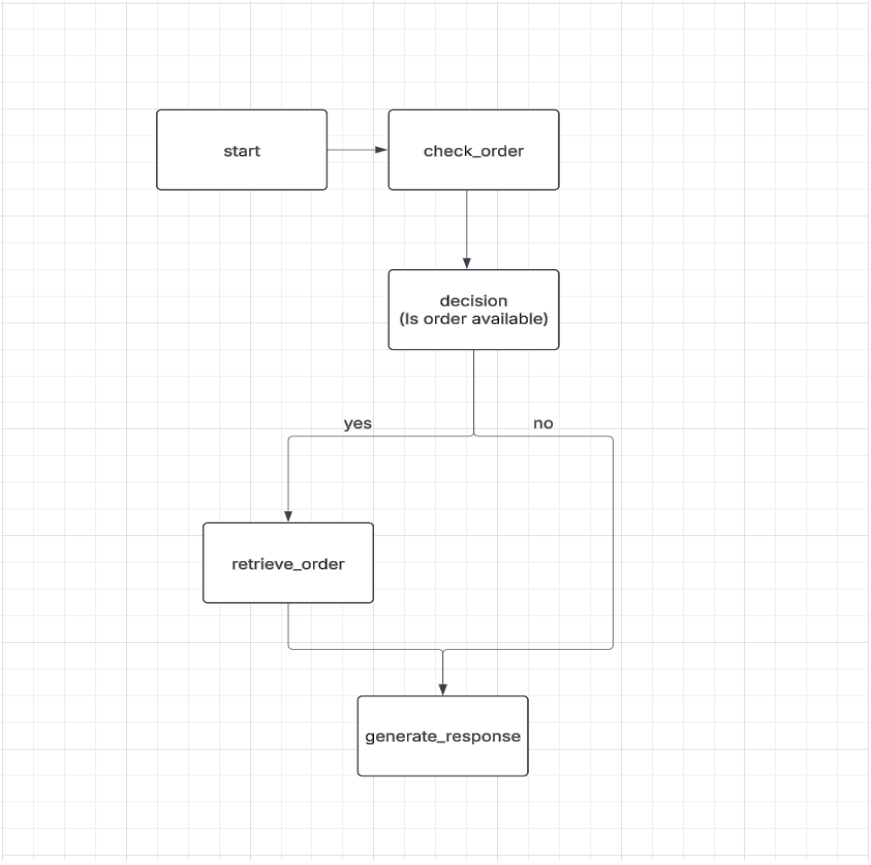

Figure 4. Order Inquiry Workflow Diagram – Status Retrieval Logic

The Key Functions are as follows:

should_retrieve_order_info(): Determines if the query references an order ID.

retrieve_order_info(): Fetches order details like shipping status, product linked to order.

generate_response(): Formats order data into a user-friendly response.

3.3 Project Structure

The project was developed using Visual Studio Code and organized into a Python structure. Figure 5 is an overview of the main files in the project's folder:

```
thesis-project/
├── README.md                # Project overview and setup instructions
├── agent.py                 # Core logic for AI query handling and LangGraph workflows
├── app.py                   # Streamlit frontend interface
├── data.py                  # Product and order data used by the agent
├── requirements.txt         # Python package dependencies
├── __pycache__/             # Compiled Python bytecode (auto-generated)
└── venv/                    # Virtual environment for dependency isolation
```

Figure 5. Project Directory Structure

Each component of the system plays a specific role. The agent.py file contains the core logic responsible for processing user messages, routing queries, managing the conversation state, and generating responses using the language model. The app.py file defines the user interface, which is built using Streamlit, allowing users to interact directly with the AI system. The data.py file includes hardcoded product and order data used for testing and demonstration purposes. The requirements.txt file lists all the necessary Python packages—such as Streamlit, LangChain, and LangGraph—ensuring that the project can be easily installed and reproduced. Lastly, the venv/ directory contains the isolated Python environment that helps manage dependencies without affecting the global system setup.

3.3.1 Core Technologies

Table 2. Core Technologies Used in the AI Agent System

Components	**Technologies**	**Role**
Language Model	LLaMA3-70B (via Groq)	Multilingual query understanding and response generation.
Workflow Engine	LangGraph	Manages stateful workflows and conditional routing based on query intent (e.g., product or order inquiries).
Orchestration	LangChain	Integrates modular components like NLP, data retrieval into pipelines.
UI	Streamlit	Web interface for customer interactions.
Backend	Python	Handles business logic, database operations (product/order data), and API integrations.

3.4 Implementation Details

The system integrates several tools chosen for their specific functions. Python was selected for its extensive AI ecosystem and easy in integration (Procter, 2025). LLaMA3-70B was used via Groq for its strong multilingual capabilities and fast inference (<1s) (Groq, 2024). LangChain allows modular integration of AI components, while LangGraph structures the conversation workflow as a state machine

(Kevinnjagi, 2025). Streamlit provides a lightweight, web-accessible frontend (Streamlit, n.d.).

3.4.1 State-Based Workflow for Query Handling

To handle different types of customer questions, like product info or order updates, this project uses a state-based workflow. This setup lets the system make decisions one step at a time based on what's happening in the conversation. It also keeps things organized and easy to follow, even when the logic becomes more complex.

A shared state (AgentState) is at the base of this workflow, which holds important information as the query moves through different steps. Each part of the process has a specific job, such as figuring out if the query is about a product or finding order details. A state graph connects all these steps and decides what should happen next based on the current state.

The AgentState, defined as a TypedDict, is used to track the state of the conversation. It includes several key fields.

The user_input stores the raw query entered by the user, which can be in either Finnish or English.

The product_info field holds any matched product information retrieved from the database.

The order_info contains relevant order details based on the user's query.

Finally, the response field stores the AI-generated reply that is presented to the user.

Figure 6 illustrates how the AgentState is initialized in the system.

```python
# Initialize the Groq LLM
llm = ChatGroq(
    api_key=os.getenv("GROQ_API_KEY"),
    model="llama3-70b-8192"
)

# Define the state for our graph
class AgentState(TypedDict):
    """Represents the state of our agent throughout the interaction."""
    user_input: str
    product_info: Optional[List[Dict]]
    order_info: Optional[Dict]
    response: Optional[str]
```

Review next file

Figure 6. Initialization of Groq + LLaMA3 and Shared Agent State (AgentState)

3.4.2 Query Routing Using LLM Reasoning

When the user submits a query for example, “Where is my order ORD-1002?”, the system sends this input to the LLaMA3 model through the Groq API. Using a structured prompt, the model decides what action to take:

Check a product’s availability, Retrieve order status, Suggest a product, Or give a direct reply.

```python
def build_graph():
    # Define state graph
    workflow = StateGraph(AgentState)

    # Add nodes
    workflow.add_node("check_product", should_retrieve_product_info)
    workflow.add_node("retrieve_product", retrieve_product_info)
    workflow.add_node("check_order", should_retrieve_order_info)
    workflow.add_node("retrieve_order", retrieve_order_info)
    workflow.add_node("generate_response", generate_response)
    workflow.add_node("next_step", lambda x: x)  # Pass-through node

    # Set the entry point
    workflow.set_entry_point("check_product")
```

Figure 7. Router Node Processing Flow – Intent Recognition

3.4.3 Workflow Orchestration with LangGraph

Based on the decision from the router node, LangGraph moves the conversation to the appropriate function node such as **check_product_availability()** or

check_order_status(). This is handled using conditional edges defined in the state graph.

```
# Set the entry point
workflow.set_entry_point("check_product")

# Add edges using conditional_edges for router nodes
workflow.add_conditional_edges(
    "check_product",
    lambda x: x.get("next", "next_step"),
    {
        "retrieve_product": "retrieve_product",
        "next_step": "next_step"
    }
)

workflow.add_edge("retrieve_product", "next_step")
workflow.add_edge("next_step", "check_order")
```

Figure 8. Conditional Edge for Product Inquiry Routing – LangGraph Logic

```
workflow.add_conditional_edges(
    "check_order",
    lambda x: x.get("next", "generate_response"),
    {
        "retrieve_order": "retrieve_order",
        "generate_response": "generate_response"
    }
)

workflow.add_edge("retrieve_order", "generate_response")
workflow.add_edge("generate_response", END)

return workflow.compile()
```

Figure 9. Conditional Edge for Order Inquiry Routing – LangGraph Logic

3.4.4 Data Retrieving logic

The Order and Product information retrieval process from data.py happens when the query is identified. If the query is related to the product, the product information will be retrieved by LLM. Also on the other side, if the query is Order related, the order retrieval function will be triggered. Figure 10 is the example that showed prompt engineering and how the data related to product rederived:

```python
def should_retrieve_product_info(state: AgentState) -> Dict[str, str]:
    """Determine if we need to retrieve product information based on user input."""
    # Using the LLM to decide if this is a product-related query
    prompt = ChatPromptTemplate.from_template(
        """Determine if the following customer query is asking about a specific mobile phone product.

        Customer query: {query}

        If the customer is asking about a specific phone, its availability, price, features, etc.,
        respond with "RETRIEVE_PRODUCT".

        If the customer is asking about an order status or anything related to an order,
        respond with "NO".

        If the query is not about a specific product or is a general greeting or question,
        respond with "NO".

        Respond with just "RETRIEVE_PRODUCT" or "NO"."""
    )

    chain = prompt | llm
    result = chain.invoke({"query": state["user_input"]}).content.strip()

    if "RETRIEVE_PRODUCT" in result:
        return {"next": "retrieve_product"}
    else:
        return {"next": "next_step"}
```

Figure 10. Prompt Engineering and Product Info Retrieval Logic

3.4.5 Response Generation

Once the relevant data is fetched from the database from data.py, the LLM generates a natural, human-like response using another structured prompt. This ensures that the reply includes exact details like product names, prices, stock levels, or order dates but never makes up information not found in the database.

```python
def generate_response(state: AgentState) -> AgentState:
    """Generate a response based on the current state."""
    new_state = state.copy()

    if state.get("product_info") is not None:
        product_info = state["product_info"]

        if not product_info:  # No products found
            response_prompt = ChatPromptTemplate.from_template(
                """You are a helpful customer service agent for a mobile phone retailer.
                The customer asked: "{query}"

                We don't have any products matching their description in our inventory.
                Our current inventory includes:
                - iPhone 15 Pro (Apple)
                - Samsung Galaxy S24 Ultra (Samsung)
                - Google Pixel 8 Pro (Google)
                - Xiaomi 14 Ultra (Xiaomi)
                - OnePlus 12 (OnePlus)

                Provide a helpful response informing them that we don't have the product they're looking for.
                Suggest they check out our available phones instead.
                Be polite and professional."""
            )

            chain = response_prompt | llm
            response = chain.invoke({
                "query": state["user_input"]
            }).content
```

Figure 11. Response Generation from Retrieved Data

3.5 Data Schema and Inventory Management

The system relies on two structured datasets: product inventory and order records, stored as Python dictionaries for simplicity in prototyping. This approach was chosen for its simplicity, readability, and ease of use during early development and testing. Python dictionaries allow fast lookup of values by key and support nested structures, which is useful for representing product specifications and detailed order information (Patnaik, 2023).

Using dictionaries made it possible to prototype the application without setting up a full database or external storage system. This helped reduce technical complexity and allowed the focus to remain on AI agent logic and natural language interactions. For small-scale applications like this, especially in early phases or for demo purposes, dictionary-based data management offers a flexible and lightweight alternative to more complex database systems.

Figure 12 and figure 13 are the schema designs and its integration with the AI agent. The Product Inventory Schema defines the structure of each product entry. Each entry includes an id, which serves as a unique product identifier such as "P001." The fields name, brand, price, and stock represent the core product metadata. The description field contains marketing text used for generating natural language responses. Additionally, the specs field is a nested dictionary that holds technical details such as screen size, storage capacity, and available colors.

Example Entry:

```
products = [
    {
        "id": "P001",
        "name": "iPhone 15 Pro",
        "brand": "Apple",
        "price": 999.99,
        "stock": 15,
        "description": "Latest iPhone with A17 Pro chip, 48MP camera, and titanium design",
        "specs": {
            "screen_size": "6.1 inches",
            "storage": "256GB",
            "colors": ["Black", "White", "Blue", "Natural"]
        }
    },
```

Figure 12. Products Inventory Schema Example – JSON Representation

Key functions within the system rely on product data to generate accurate responses. The retrieve_product_info() function is responsible for matching user queries such as “iPhone 15 Pro” against the name or brand fields in the product dataset to find relevant results.

The Order Management Schema defines the structure of each order entry. Each entry contains an order_id, which is a unique identifier such as "ORD10001." The product_id field links the order to a corresponding product in the inventory. Additional fields such as status, tracking_number, and order_date provide essential shipping metadata that the system uses to inform users about their order status.

Example Entry:

```
# Order data
orders = [
    {
        "order_id": "ORD10001",
        "customer_name": "John Smith",
        "product_id": "P001",
        "quantity": 1,
        "status": "Shipped",
        "shipping_address": "123 Main St, New York, NY",
        "tracking_number": "TRK78945612",
        "order_date": "2023-04-15"
    },
    {
        "order_id": "ORD10002",
        "customer_name": "Emma Johnson",
        "product_id": "P002",
        "quantity": 1,
        "status": "Processing",
        "shipping_address": "456 Oak Ave, Los Angeles, CA",
        "tracking_number": None,
        "order_date": "2023-04-18"
    },
```

Figure 13. Order Schema Example – Order and Product Linkage

Key functions also operate on order data to provide detailed and helpful responses. The retrieve_order_info() function extracts order IDs from user queries, such as “Status of ORD10001?”, and retrieves the corresponding order details. It then joins this information with the products table to enrich the response with relevant product data. The generate_response() function uses this combined information to produce natural, informative replies for example: “Order ORD10001 (iPhone 15 Pro) shipped on 2023-04-15. Tracking: TRK78945612.”

3.6 Result

Figure 14 and figure 15 shows the results that AI agent’s frontend is built with Streamlit, providing an interactive web interface for customer queries.

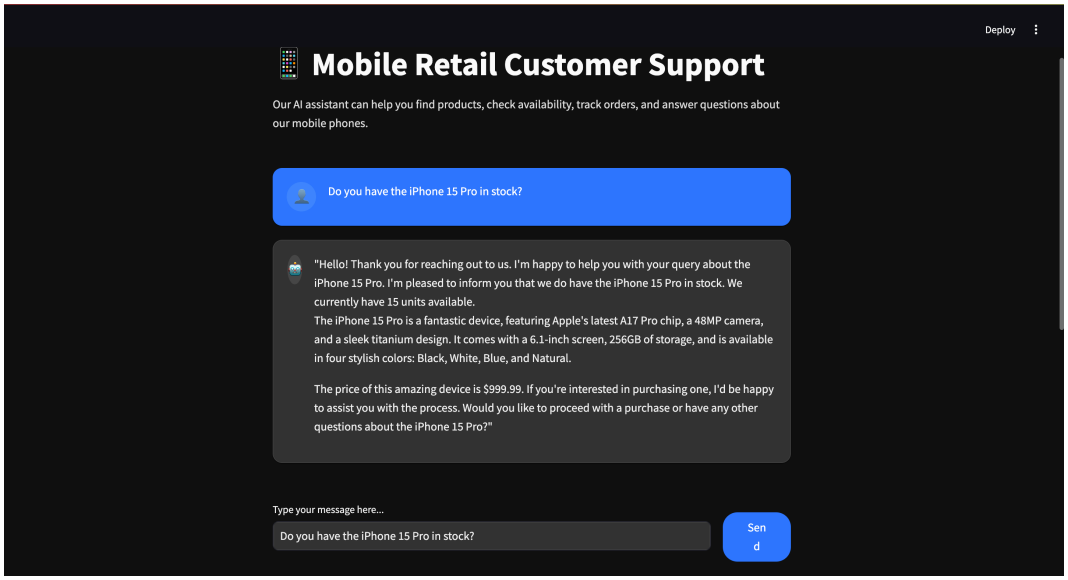

Figure 14. AI Response to English Query: "What is the price of iPhone 15?"

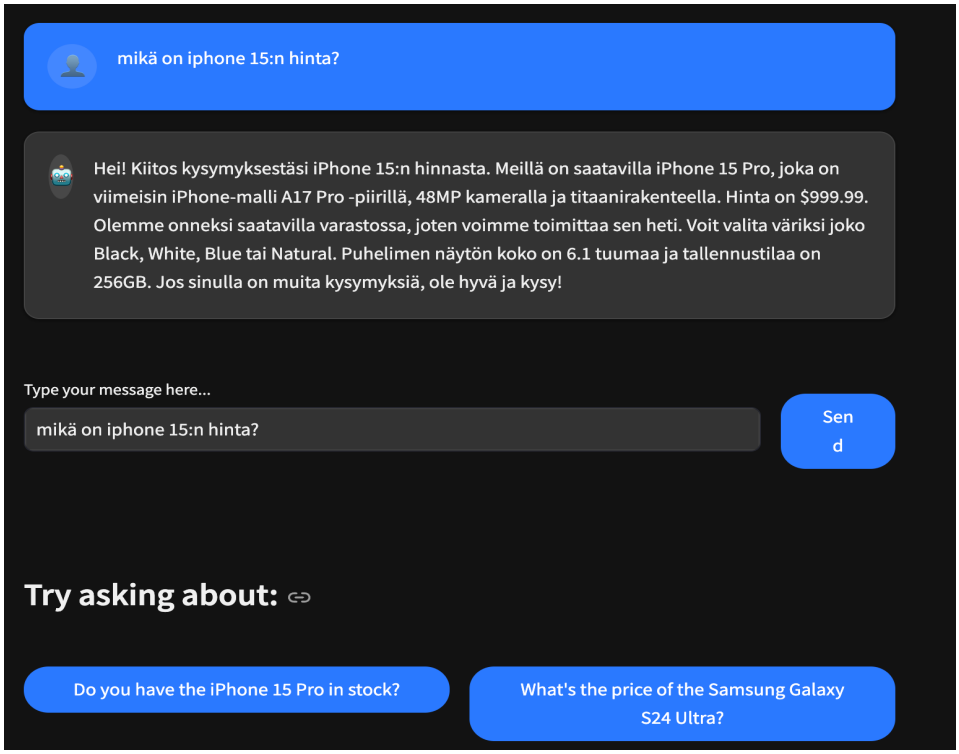

Figure 15. AI Response to Finnish Query: "Mikä on iPhone 15:n hinta?"

The user asks, "what is the price of Iphone 15?", and the AI responds with availability, price, and specifications.

4 TESTING AND ANALYSIS

4.1 Testing Approach

To check the performance of AI agent, 50 common support questions in both English and Finnish using the Streamlit interface. These questions included things like checking product availability, order status, getting product recommendations, and general chat. The goal was to see how fast and accurate the AI performed how well it could handle ongoing conversations, and it stuck to the available data without making stuff up. Most of the questions were answered correctly, but there were a few issues. Sometimes the AI did not quite understand vague questions like “Can you help me?” Also, since it is not connected to live data, it cannot give real-time updates. And it does not recognize emotions, which can make the responses feel a bit robotic in some situations.

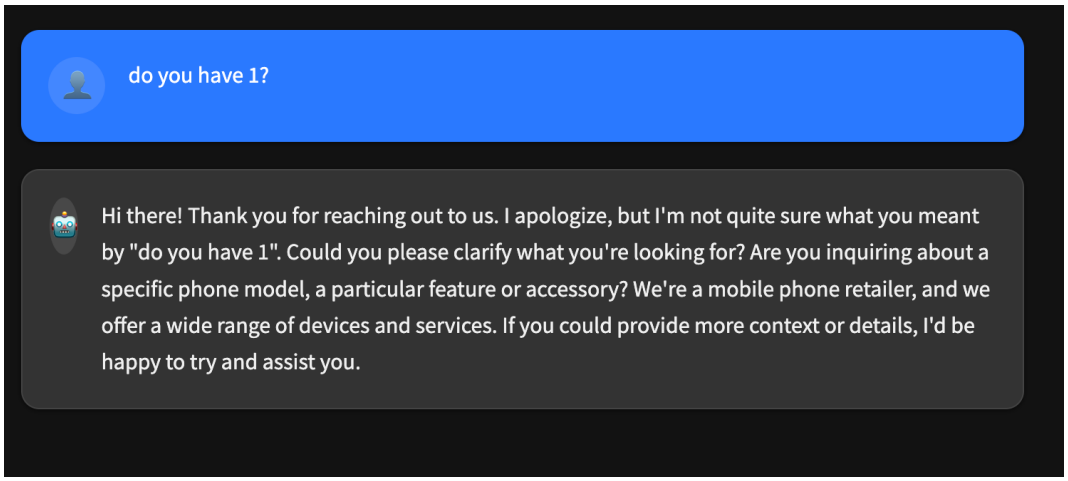

Figure 16. Example of a query test

Figure 17 is an example test that demonstrates how the agent responds when a product is out of stock.

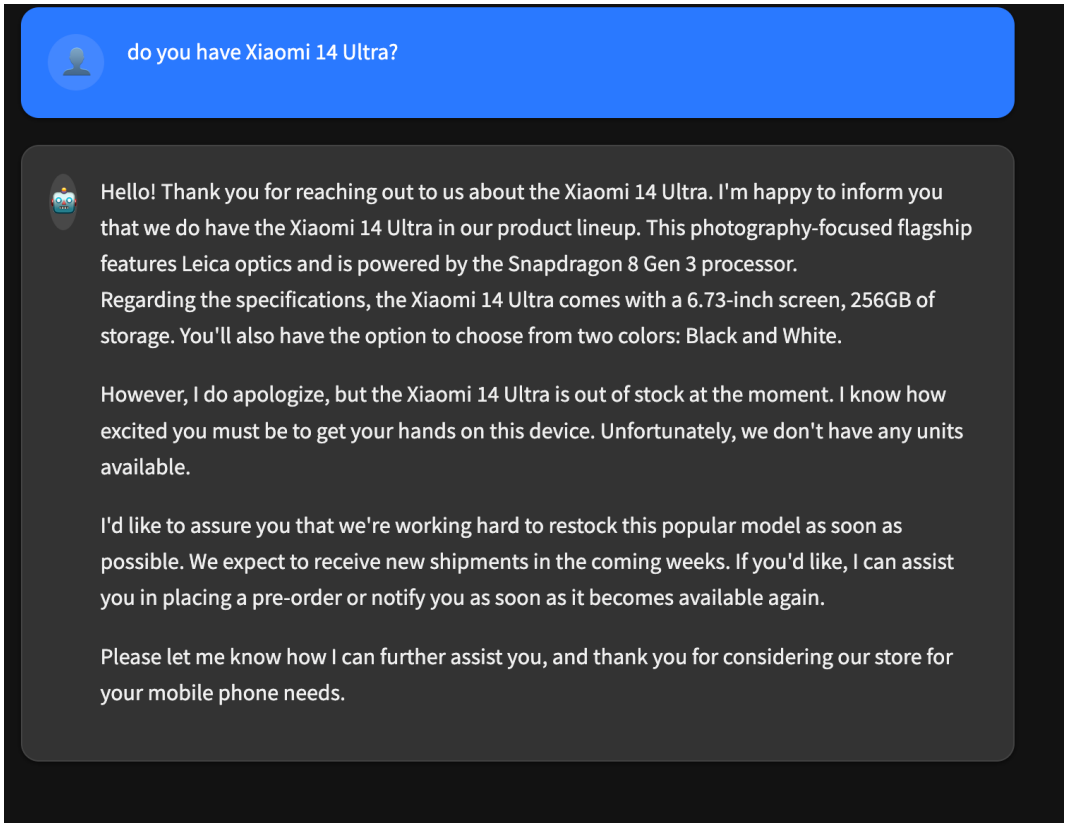

Figure 17. AI Response to Product with Zero Stock


```
{
    "id": "P004",
    "name": "Xiaomi 14 Ultra",
    "brand": "Xiaomi",
    "price": 1099.99,
    "stock": 0,
    "description": "Photography-focused flagship with Leica optics and Snapdragon 8 Gen 3",
    "specs": {
        "screen_size": "6.73 inches",
        "storage": "256GB",
        "colors": ["Black", "White"]
    }
},
```


Figure 18. Database Record of Out-of-Stock Product

4.2 Analysis

4.2.1 Strengths

The implemented AI agent demonstrated several practical strengths during testing. One of the most significant benefits observed was its high-speed performance, achieved through the integration of the Groq API. The system was able to return responses in under one second, which is essential for maintaining user engagement and satisfaction in customer service scenarios. This speed advantage enables businesses to handle queries more efficiently, especially during peak hours.

Another big advantage of the system is that it can understand and respond in both Finnish and English without needing separate setups. This is very useful for Finnish SME's, where customers often speak different languages. It also means businesses don't need to buy extra systems or translation tools, which saves time and money.

The modular structure of the system, made possible by LangChain and LangGraph, allows for scalability and maintainability. Developers can easily add or modify workflow components without disrupting the overall architecture. This makes the system suitable not only for retail businesses but also adaptable to various sectors that require custom workflows and intent handling.

4.2.2 Limitations

Despite its advantages, the system has a few limitations. The current prototype relies on static, hardcoded product and order data. This restricts the system's real-world application unless it is connected to a dynamic database.

Even though the system works well in many ways, it still has some weaknesses. Right now, it uses fixed product and order data that doesn't change. This makes it less useful in real business settings unless it is connected to a live database. Also, it does not remember users or keep track of conversations, so it cannot give more personal replies. The system also cannot tell if a user is angry or upset, so some

answers may feel cold or not human enough. Finally, while the language model works well for clear questions, it sometimes gets confused by unclear or vague ones. This shows that better handling of tricky questions is still needed.

5 DISCUSSION AND CONCLUSION

This thesis presented the development and implementation of an AI agent customer support system tailored to the needs of Finnish SMEs. The solution leverages state-of-the-art technologies, including the LLaMA3-70B model accessed via the Groq API, along with LangChain and LangGraph for managing modular and stateful workflows. Streamlit was used to create a user-friendly web interface, making the system accessible to non-technical users. The system successfully handled both English and Finnish queries, accurately classified user intent, and provided real-time responses with sub-second latency. These results demonstrate the feasibility of deploying low-cost, high-performance AI support systems within small business environments.

The project offers a useful and flexible AI solution for small and medium-sized businesses, which often lack the financial and technical resources to adopt advanced technology. By relying on open-source tools and cloud-based services, the system reduces costs and increases accessibility. Its modular and adaptable design makes it suitable for other industries beyond retail, such as healthcare and hospitality. This shows that powerful AI tools can also benefit smaller companies, not just large enterprises.

In terms of broader application, the system could be extended to various fields. In healthcare, it could assist with appointment booking or basic symptom triage. In the finance sector, it could help customers navigate services or accounts. In hospitality, it could support tasks such as bookings or responding to guest inquiries. Because the system supports multiple languages and offers fast response times with intelligent task routing, it is well-suited for any industry that requires real-time, multilingual customer interaction.

5.1 Future Work

There are a few ways this system could be improved in the future. First, adding a real-time database which would let it show up-to-date product and order information. Second, if the system could remember users and past chats, it could give more personal answers. Third, teaching the system to understand emotions would help it give more caring and human like replies. Last, adding checks to make sure the system stays fair and does not treat anyone differently would help keep it ethical as it grows.

5.2 Acknowledgement of Language Assistance

Some minor assistance was used from an AI tool (ChatGPT) to improve the language clarity and grammar of English sentences during the writing of this thesis. The core ideas, structure, content, and analysis are fully original and based on my own work.

REFERENCES

An, F. (2025). *Key AI in customer service statistics for 2025 and beyond*. Sobot. Retrieved 2025-03-15, from https://www.sobot.io/article/key-ai-in-customer-service-2024/

Anand, M. (2024). *AI is breaking down language barriers in customer service*. Reverie Inc. Retrieved 2025-03-19, from https://reverieinc.com/blog/ai-is-breaking-down-language-barriers-in-customer-service/

AWS. (n.d.). *What is LangChain?* Amazon Web Services. Retrieved 2025-03-22, from https://aws.amazon.com/what-is/langchain/

Bheda, B. (2025). *Groq API: The fastest LLM API for real-time AI applications*. DhiWise. Retrieved 2025-03-20, from https://www.dhiwise.com/post/groq-api-fastest-llm

Brynjolfsson, E., Li, D., & Raymond, L. (2024). *Generative AI at work* (Version 2) [Working paper]. arXiv. Retrieved 2025-03-22, from https://arxiv.org/abs/2304.11771

Eppright, C. (2021). *What is natural language processing (NLP)*. Oracle. Retrieved 2025-03-19, from https://www.oracle.com/artificial-intelligence/what-is-natural-language-processing

Fontanella, C. (2024). *33 customer service tips*. HubSpot. Retrieved 2025-03-13, from https://blog.hubspot.com/service/customer-service-tips

Gagyi, S. (2023). *7 IT challenges faced by SMEs in 2023*. LabyrinthIT. Retrieved 2025-03-19, from https://www.labyrinthit.com/7-it-challenges-faced-by-smes-in-2023/

Grattafiori, A., et al. (2024). *The Llama 3 herd of models* (Version 3) [Preprint]. arXiv. Retrieved 2025-03-25, from https://doi.org/10.48550/arXiv.2407.21783

Groq. (2024). *Groq supercharges fast AI inference for Meta LLaMA 3.1*. Retrieved 2025-03-23, from https://groq.com/news_press/groq-supercharges-fast-ai-inference-for-meta-llama-3-1/

Groq. (n.d.). *LLaMA-3.3-70B-versatile model documentation*. Retrieved 2025-03-23, from https://console.groq.com/docs/model/llama-3.3-70b-versatile

Hakola, D. (2025). *Finnish SMEs need more courage, support and networks – Growth opportunities discussed at the BOOST networking workshop in*

January. University of Jyväskylä. Retrieved 2025-04-03, from https://www.jyu.fi/en/news/finnish-smes-need-more-courage-support-and-networks-growth-opportunities-discussed-at-the-boost

Hayes, M., & Downie, A. (2025). *Better customer service with AI agents*. IBM. Retrieved 2025-03-19, from https://www.ibm.com/think/topics/ai-agents-in-customer-service

Huseyn. (2025). *Multilingual customer support explanation + how it can help your business*. Trengo. Retrieved 2025-03-15, from https://trengo.com/blog/multilingual-customer-support-explanation

Hyken, S. (2024). *Bad customer service could cost more than $3.7 trillion*. Forbes. Retrieved 2025-03-20, from https://www.forbes.com/sites/shephyken/2024/03/17/bad-customer-service-could-cost-more-than-37-trillion/

Julia, G. P. (2024). *The evolution of AI agents: From simple tasks to advanced problem-solving*. BritannicaWeb. Retrieved 2025-04-09, from https://britannicaweb.com/the-evolution-of-ai-agents-from-simple-tasks-to-advanced-problem-solving

Kevinnjagi. (2025). *LangGraph: Building stateful AI agents*. Medium. Retrieved 2025-04-03, from https://medium.com/@kevinnjagi83/langgraph-building-stateful-multi-ai-agents-b8427238da91

LangChain. (2023). *LangChain documentation*. Retrieved 2025-03-29, from https://www.langchain.com/docs

LangChain. (2024). *LangGraph: Multi-agent workflows*. Retrieved 2025-03-29, from https://blog.langchain.dev/langgraph-multi-agent-workflows/

Liz, G. (2025). *The evolution of AI agents: From simple programs to agentic AI*. WWT. Retrieved 2025-04-09, from https://www.wwt.com/blog/the-evolution-of-ai-agents-from-simple-programs-to-agentic-ai

Mäntylä, A. (2024). *Artificial intelligence in Finnish SME marketing: Adoption, potential, and ethical challenges* [bachelor's thesis, Jamk University of Applied Sciences]. Theseus. Retrieved 2025-04-19, from https://www.theseus.fi/bitstream/handle/10024/876704/Mantyla_Artturi.pdf

Miloiu, C. (2025). *Genezio has Streamlit support: Simplified AI prototyping and interactive dashboards*. Genezio. Retrieved 2025-03-19, from https://genezio.com/deployment-platform/blog/streamlit/

Muller, M. (2025). *Rules*. Rasa. Retrieved 2025-04-15, from https://legacy-docs-oss.rasa.com/docs/rasa/rules

Pangan, K. (2025). *A complete guide to Zendesk AI agents: Setup, costs, and best practices*. Eesel. Retrieved 2025-04-15, from https://www.eesel.ai/blog/a-complete-guide-to-zendesk-ai-agents-setup-costs-and-best-practices

Patnaik, A. (2023). *Python dictionary: A powerful tool for data engineers*. DZone. Retrieved 2025-04-10, from https://dzone.com/articles/python-dictionary-a-powerful-tool-for-data-enginee

Porubská, S. (2025, March 21). *How to facilitate multilingual customer support*. CloudTalk. Retrieved 2025-03-29, from https://www.cloudtalk.io/blog/multilingual-customer-support

Procter, A. (2025, February 28). *Why Python is the first choice for AI innovation*. Okoone. Retrieved 2025-03-29, from https://www.okoone.com/spark/technology-innovation/why-python-is-the-first-choice-for-ai-innovation/

Python Software Foundation. (n.d.). *What is Python?* Retrieved 2025-03-29, from https://www.python.org/doc/essays/blurb/

Qamar, R., & Zardari, B. A. (2023). *Artificial neural networks: An overview*. *Mesopotamian Journal of Computer Science, 2023*, 130–139. https://doi.org/10.58496/MJCSC/2023/015

Rowe, A. (2025). *Businesses spend €50k–€100k on AI*. Tech.co. Retrieved 2025-03-27, from https://tech.co/news/businesses-spend-50k-100k-ai

Sellberry. (2024). *Affordable AI for small businesses: A comprehensive guide*. Retrieved 2025-03-25, from https://sellbery.com/blog/affordable-ai-for-small-businesses-a-comprehensive-guide/

Sridhar, C. S. (2024). *The evolution of customer service from chatbots to conversational AI agents*. Purplescape. Retrieved 2025-04-09, from https://purplescape.com/evolution-of-customer-service-from-chatbots-to-conversational-ai-agents/

Streamlit. (2023). *Streamlit documentation*. Retrieved 2025-03-26, from https://docs.streamlit.io

Streamlit. (n.d.). *Understanding Streamlit's client-server architecture*. Retrieved 2025-03-26, from https://docs.streamlit.io/develop/concepts/architecture/architecture

Stryker, C., & Kavlakoglu, E. (2024, August 9). *What is artificial intelligence (AI)*. IBM. Retrieved 2025-04-27, from https://www.ibm.com/think/topics/artificial-intelligence

Teradata. (n.d.). *What is Python programming language?* Retrieved 2025-03-19, from https://www.teradata.com/insights/data-platform/what-is-python-programming-language

Vaswani, A., et al. (2017). *Attention is all you need*. arXiv. Retrieved 2025-04-07, from https://arxiv.org/abs/1706.03762

Yle. (2020). *Language-based AI leaves Finland behind—Can a new initiative change that?* Retrieved 2025-04-29, from https://yle.fi/a/3-11539612

Yrittäjät. (2023). *Survey: Almost 40% of businesses having difficulty finding employees*. Retrieved 2025-04-13, from https://www.yrittajat.fi/en/news/survey-almost-40-of-businesses-having-difficulty-finding-employees/