Republic of Tunisia

*****

Ministry of Higher Education and Scientific Research

*****

University of Monastir

*****

Higher Institute of Computer Science and Mathematics of Monastir

Department of Computer Science

Order N°: L15

End of Studies Project Report

Presented in order to obtain the

National Bachelor's Degree in Computer Science

Specialization:

Software Engineering and Information Systems

By:

Ebdelli Khalyl

Agent-Based Solution for Non-Conformity Detection

Presented on 23/06/2025 in front of the jury composed of:

Ms. Sameh Hbaieb	President
Ms. Achich Nassira	Member
Ms. Imen Ben Ahmed	Academic Supervisor
Mr. Oussema Chargui	Professional Supervisor

Academic Year: 2024 / 2025

Abstract

Standard Operating Procedure (SOP) compliance audits in retail present significant research challenges due to their labor-intensive nature, susceptibility to error, and lack of scalability. This research addresses these challenges by proposing and evaluating an automated compliance assessment framework offered as Multi-Agent System (MAS) Architecture. Our core contribution lies in the design, orchestration and integration of domain-specialized intelligent agents, with each being paired with a fine-tuned AI model (CamemBERT, Gemma3-4B, Gemma3-1B) capable of processing complex multimodal survey data (structured, semi-structured, and unstructured). This agent-based approach moves beyond simple automation, the agents autonomously detect non-conformities and generate actionable recommendations, replacing subjective manual reviews with objective, data-driven evaluations. Empirical results from preliminary experiments validate the System, demonstrating improvements in non-conformity Mean-Time-To-Resolve (MTTR) and overall processing efficiency.

Keywords: Multi-Agent System, SOP compliance, automation, AI Agents, Non-Conformity Detection, Fine-Tuning.

Dedication

This project is dedicated to:

My father **Wejden** and my mother **Hedia**, thank you for always believing in me, even when I doubted myself, thank you for all the sacrifices you are still making till this day for me and my siblings till this day, thank you for teaching me the values of honesty, respect, and the true meaning of family.

To my brother and mentor, **Moemen**, thank you for being the guiding light in my life with your wisdom and patience. Your presence has been a source of reassurance throughout my life and I am truly grateful and proud to have you as my brother.

To my sister **Zayneb**, Thank you for your love and support in this journey. You've supported and uplifted everyone with a maturity beyond your years, and for that, I'm incredibly proud of you. I hope to always be someone you can look up to.

To my uncle **Ali**, throughout my life you've been more than just an uncle, a role model and a mentor. From day one, you took me under your wing, shared with me life lessons that no classroom could teach, and passed on values and skills that helped shape who I am today. For that, I'm forever grateful, and I'll always carry the impact you've had on me with pride and respect.

To the rest of my family, thank you for your constant love, encouragement, and support throughout this journey. Thank you for keeping me in your prayers, I'm truly grateful to have you in my life.

To my friends and special ones, thank you for being there for me through it all, through the laughs, late night talks, reality checks, and constant encouragement. Your presence made this journey lighter, better, and unforgettable. Your friendship means the world to me, now and always.

اللهم إن نعمك كثيرة علينا لا نحصيها ولا نحصي ثناء عليك ولا نقدر وأنت سبحانك كما أثنيت
على نفسك وأنت سبحانك غني عن العالمين.

Acknowledgment

The successful completion of this project would not have been possible without the support and dedication of several individuals, to whom I extend my sincere gratitude.

I would like to express my heartfelt thanks to my academic supervisor, **Ms. Imen Ben Ahmed**, for her insightful feedback, continuous guidance, and genuine commitment throughout this journey. Her expertise and encouragement have been instrumental to the progress and success of this work.

I would also like to express my deep appreciation to my professional supervisor, **Mr. Chargui Oussema**, for his unwavering support, care, and the time he generously dedicated to guiding me. His willingness to invest in my growth, both professionally and personally, made a significant impact throughout this journey.

A big thanks goes to the whole **Winshot** Team for their responsiveness, quick thinking and collaboration. Their openness to new ideas greatly contributed to the smooth progress of this project.

I convey my heartfelt gratitude to the members of the jury **Ms. Sameh Hbaieb Turki** and **Ms. Nassira Achich** who agreed to examine and evaluate my work.

Finally, I would like to thank the few individual teachers and staff at ISIMM for their dedication and help throughout these three past years.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
MAS	Multi-Agent System
MCP	Model Context Protocol
NLP	Natural Language Processing
LLM	Large Language Model
SOP	Standard Operating Procedure
MVP	Minimum Viable Product
CRISP-DM	Cross Industry Standard Process for Data Mining
IDE	Integrated Development Environment
VLM	Vision-Language Model
API	Application Programming Interface
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
EDA	Exploratory Data Analysis
ML	Machine Learning
SaaS	Software as a Service
QLORA	Quantized Low-Rank Adaptation
VM	Virtual Machine
AWS	Amazon Web Services
GCP	Google Cloud Platform

General Introduction

Establishment

Retail chains are made up of multiple branches that must adhere to internal policies and Standard Operating Procedures (SOPs) to ensure consistency, quality, and compliance across all locations. As the number of branches grows, maintaining this consistency becomes increasingly difficult, especially when relying on manual processes.

Traditionally, store compliance has been monitored through daily surveys manually reviewed by specialists. However, this approach is time-consuming, error-prone, and lacks scalability. These challenges are amplified as businesses expand and the volume of data increases.

In this context, our project, developed within **WinShot**, aims to design and implement an AI-powered solution for automating non-conformity detection based on a Multi-Agent System (MAS). The objective is to provide a smart and scalable way to analyze survey data in real time and identify deviations from SOPs.

Our system combines agent-based design with advanced AI models to deliver an efficient, modular, and intelligent platform. It enhances decision-making, ensures faster response to non-conformities, and improves overall compliance monitoring across retail networks.

Work Organization

This report presents a comprehensive overview of the research, design, and implementation of the proposed Multi-Agent System. The structure of the report is as follows.

- **Chapter 1: Scope of the Project**
  Introduces the project scope and the institutional context. Describes the motivation, objectives, and methodology adopted throughout the project.

- **Chapter 2: Project Background**
  Provides a review and background on existing systems. It discusses the state of the art in

retail business, AI, and traditional non-conformity detection systems.

- **Chapter 3: Technical Study**
  Describes the technical setup used in the project and details the development environment, libraries, tools, and technologies employed.

- **Chapter 4: Data Analysis** Explores the data used in the system, including collection, curation, and preprocessing. This chapter also presents any structured knowledge representations built from the data.

- **Chapter 5: Modeling and Evaluation** Details the models and algorithms developed for MAS, including their evaluation metrics and performance benchmarks.

- **Chapter 5: Deployment and Integration** Covers the deployment environment for the Multi-Agent System, as well as its integration within Winshot's platform.

Chapter 1

Project Scope

1.1 Introduction

The purpose of this chapter is to place the project in its general context, namely, the problem to be addressed and the proposed solution in detail.

In the first section, a brief presentation of the host organization and its respective areas of expertise is given, followed by an introduction to the general context and the methodology employed for this project.

1.2 Host Organization Presentation

Gofield

Gofield is a startup founded in September 2019. It specializes in designing, developing, and marketing innovative and high-value-added niche solutions in the retail industry.

From the very beginning, Gofield was selected by renowned startup accelerators such as Orange Fab, the corporate accelerator of Orange Tunisia, and Flat6Labs, a leading accelerator in the startup ecosystem.

This allowed **GoField** to develop **Winshot**, a product quickly adopted by many Tunisian brands and distributors, such as Orange Tunisia, Total Tunisia, and Cosmitto. Gofield has also been certified under the StartupAct law since October 2019.

(a) Gofield **(b) Winshot**

Figure 1.1: Winshot and Gofield Logos

Winshot

Winshot is a comprehensive SaaS platform developed by GoField,as shown in Figure 1.2 it is designed to streamline and optimize retail operations. It consolidates store visits, operational tasks, and brand standards into a single, user-friendly interface, empowering frontline teams to perform with autonomy and precision across every location.

The platform offers tools to automate workflows, provide real-time updates, and facilitate seamless communication between retail stores and headquarters. This ensures that daily tasks are simplified and operational efficiency is boosted.

Winshot is trusted by various retail businesses in Tunisia, including fashion like *Exist*, telecommunications like *Orange*, and food sectors.

Figure 1.2: Winshot Activities

1.3 Project Presentation

1.3.1 Project Context

This project was carried out as part of a four-month end-of-studies internship at GoField, in fulfillment of the requirements for a Bachelor's degree in Computer Science. It focuses on the integration of a **Multi-Agent System** into **WinShot's** platform, with the goal of automating non-conformity detection from survey reports.

1.3.2 Problem Statement

Retail chains often rely on daily surveys based on Standard Operating Procedures (SOPs) to monitor compliance across their branches. These surveys are typically reviewed manually by store operations specialists, making the process time-consuming, repetitive, and prone to human error.

This process is prone to the following issues:

- **Human error:** Manual review of survey results can lead to oversight, inconsistencies, and subjective judgment.
- **Inefficiency:** The process is time-consuming and requires significant human effort, especially as the number of branches grows.
- **Delayed response:** Identifying and addressing non-conformities takes longer, potentially impacting store performance and compliance.

To address these issues, this project aims to answer the following questions:

Identified Issue	**Guiding Question**
Human error during manual review	How can we reduce reliance on subjective and error-prone human evaluation?
Inefficient and time-consuming process	What solutions can automate the review process to improve efficiency and scalability?
Delayed detection of non-conformities	How can we ensure timely identification and resolution of non-compliance to maintain operational standards?

Table 1.1: Project-related issues and corresponding guiding questions

1.3.3 Project Purpose

The primary objective of this project is to automate the detection of non-conformity from survey results through advanced AI capabilities.

By following an agentic approach, we aim to achieve the following goals as an answer to the previous posed questions:

Guiding Question	Project Objective
How can we reduce reliance on subjective and error-prone human evaluation?	Delegate evaluation to specialized agents for more consistent and objective decision-making.
What solutions can automate the review process to improve efficiency and scalability?	Design an agentic system that automates survey analysis and scales with operational demands.
How can we ensure timely identification and resolution of non-compliance to maintain operational standards?	Enable real-time detection and flagging of non-conformities for faster resolution.

Table 1.2: Guiding questions and corresponding project objectives

The 5 W's

Aspect	Description
What?	• The project involves the development and integration of a MAS into WinShot's platform.
Why?	• Reduce human error, increase efficiency, and provide real-time insights • Automate non-conformity detection form survey results.
Who?	• The MAS is intended for retail stores.
Where?	• The MAS will be integrated into Winshot platform.
When?	• The MAS is being developed as part of the current project and is planned to be ready by June.

Table 1.3: The Project's 5 W's

1.3.4 Project Steps

This project will follow a phased approach to develop the AI-powered non-conformity detection system based on a Multi-Agent architecture.

Each phase focuses on specific tasks and deliverables to ensure successful project completion. Figure 1.3 shows each project step duration.

- **Data Acquisition:** Collect historical survey data from the host organization and assess its quality and relevance.
- **Data Preparation:** Clean, format, annotate (for supervised learning), and split the data into training, validation, and testing sets.
- **Architecture Development:** Design the Multi-Agent System (MAS), define agent roles (e.g., SupervisorAgent, TextAgent), and choose appropriate models and frameworks (e.g., Transformers, Hugging Face, LangGraph).

- **Testing and Evaluation:** Evaluate model performance using relevant metrics, conduct agent integration testing, and verify system robustness.
- **Deployment:** Integrate the finalized MAS into WinShot's infrastructure for automated, real-time processing of survey reports.

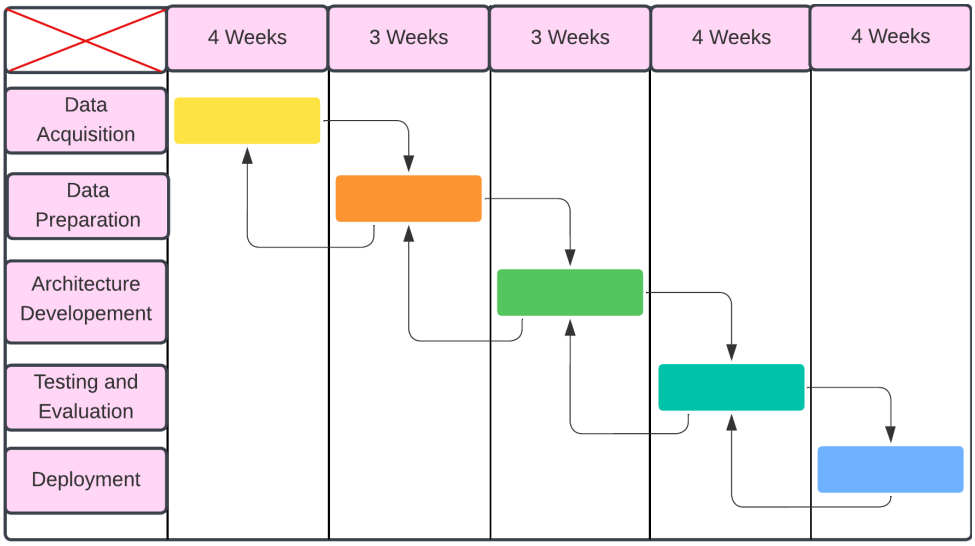


Figure 1.3: Project Steps Gantt Chart

1.4 Adopted Project Management Methodology

There are various project management methodologies like Scrum, Kanban, and Waterfall. Figure 1.4 presents statistics on the most widely used methodologies in AI projects.

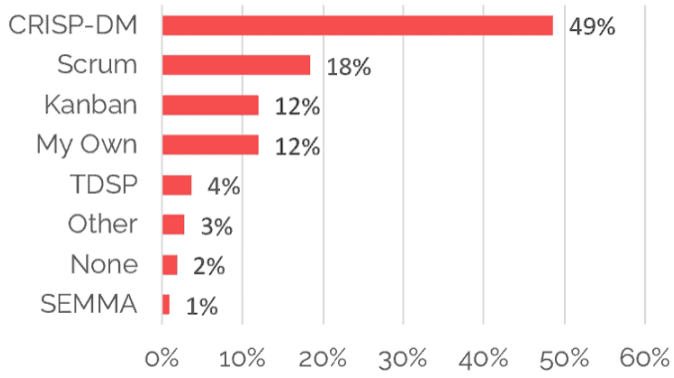


Figure 1.4: Most used Project Management Frameworks in AI

For this project, we adopted the **CRISP-DM** (Cross Industry Standard Process for Data Mining) methodology. CRISP-DM is a well-established framework in data mining and AI that ensures structured, iterative progress.

CRISP-DM Phases

1. **Business Understanding:** Define project objectives and understand the business problem.
2. **Data Understanding:** Explore data sources, assess quality, and identify initial insights.
3. **Data Preparation:** Clean, transform, and select data to prepare it for modeling.
4. **Modeling:** Apply machine learning and AI techniques to build predictive or descriptive models.
5. **Evaluation:** Assess model performance regarding business goals and success criteria.
6. **Deployment:** Integrate the final model into a production environment for real-time use.

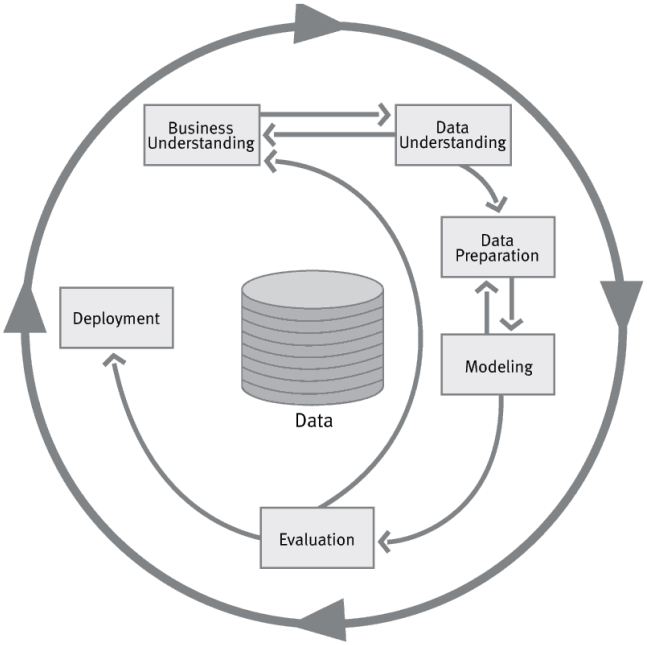


Figure 1.5: CRISP-DM Phases

The CRISP-DM methodology is iterative by design. Phases can be revisited as new insights are discovered, ensuring flexibility and robustness throughout the project life-cycle. Figure 1.5 shows CRISP-DM iterative phases and Table 1.4 shows our adaptation and implementation of these phases into our project.

CRISP-DM Phase	Project Adaptation
Business Understanding	Define the goal of automating non-conformity detection using intelligent agents in WinShot's survey platform.
Data Understanding	Collect and explore historical survey reports to understand structure, answer types, and common compliance issues.
Data Preparation	Clean, annotate, and split the dataset (survey entries) for model training and evaluation, especially for text-based questions.
Modeling	Fine-tune transformer models (e.g., CamemBERT, Gemma3-4B) for non-conformity detection, action recommendation and justification generation; integrate within agent roles.
Evaluation	Evaluate agent performance using accuracy, F1-score, and qualitative analysis of justifications to ensure system effectiveness.
Deployment	Deploy the Multi-Agent System within WinShot's infrastructure for real-time analysis.

Table 1.4: CRISP-DM Methodology and its Adaptation in the Project

1.5 Conclusion

This chapter provided an overview of the project's purpose, its context within WinShot, and the methodology used to guide its development. It defined the problem, presented the proposed agent-based solution, and detailed the project steps.

Chapter 2

Project Background

2.1 Introduction

This chapter lays the ground foundations for the project by highlighting the current method of non-conformity detection, survey validation, and how our proposed **MAS** can address its challenges. The goal is to dive into the deficiencies of traditional audit validation methods, time-consuming protocols, and discuss how advancements in Deep Learning and Computer Vision offer irresistible solutions.

2.2 Project Context

2.2.1 Contextual Background

Current non-conformity detection workflows remain almost entirely **manual**. Operations specialists or store managers must review each question-answer pair individually.

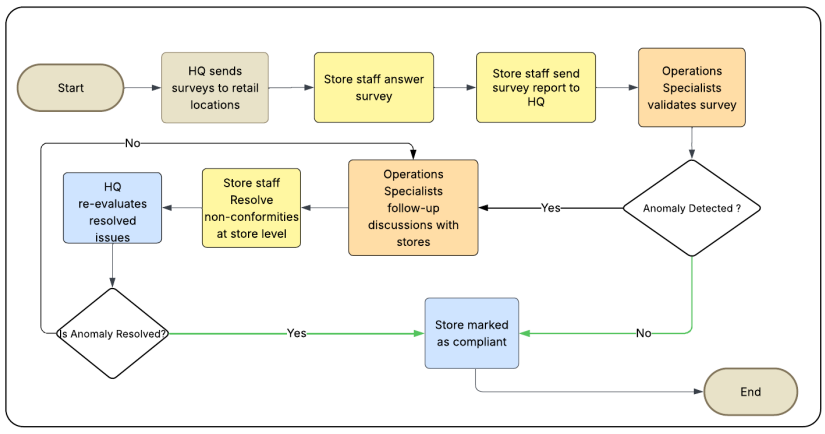


Figure 2.1: Current Non-Conformity Detection Workflow

The Flowchart presented in Figure 2.1, explains the traditional approach for non-conformity detection provided through **Winshot** platform.

2.2.2 Limitations

While this hands-on approach can catch overt violations of *SOPs*, it doesn't scale, especially as surveys grow in length and complexity.

The process consumes significant staff time, drives up operational costs, and introduces variability: different reviewers may interpret the same survey reports in various ways, and fatigue only heightens the risk that subtle deviations will slip through. The consequences of this manual, subjective process go beyond inefficiency.

First, delayed detection of non-conformity can lead to prolonged SOP violations, degrading customer experience, increasing compliance risk, and exposing the brand to reputational damage. Second, the lack of standardization means stores receive uneven feedback, some may be corrected immediately, while others continue operating out of spec due to review backlog or oversight.

Lastly, even with the best efforts, the process remains inherently slow, labor-intensive, and inconsistent. These structural inefficiencies and long command chain make it difficult to maintain high operational standards across a growing network of stores.

2.3 Technical Background

Traditional AI approaches, such as rule-based systems or classical machine learning models, often fall short in tasks involving unstructured or semi-structured data. These models typically rely on predefined features and shallow pattern recognition, which limits their ability to capture context, handle linguistic variability, or adapt to subtle nuances in human language. In the context of survey analysis for compliance, where answers can vary widely in form and intent, such limitations become especially problematic.

Given the complexity of interpreting free-text responses and the need for models that can generalize beyond surface-level cues, more advanced techniques are required. This has led to the adoption of modern architectures capable of deeper semantic understanding, particularly, transformer-based models.

2.3.1 Transformers Architecture

In recent years, transformer architectures have fundamentally transformed the field of artificial intelligence, particularly in natural language processing (NLP). Since their introduction by Vaswani et al. in the landmark paper "Attention is All You Need" (2017)[15], transformers have

rapidly replaced traditional recurrent and convolutional models across a wide range of NLP tasks.

Central to the transformer's success is the self-attention mechanism, which enables the model to evaluate the relevance of each word (or token) in a sentence relative to all others, regardless of their position. This approach allows the model to understand context more effectively than earlier architectures, which struggled with long-range dependencies.

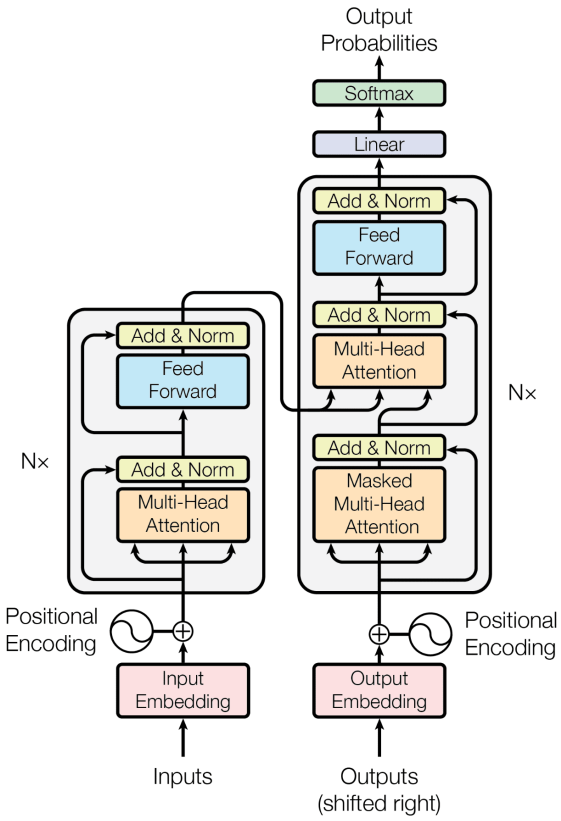


Figure 2.2: Transformers Architecture

Building on this mechanism, as shown in Figure 2.2 transformers are composed of stacked layers that include both self-attention and feedforward components. Unlike sequential models, transformers process input data in parallel, making them highly efficient for handling large datasets. As a result, they have become the foundation of many state-of-the-art systems in tasks such as machine translation, text summarization, and question answering, pushing the boundaries of what's possible in NLP.

In our case, transformer-based models enable deep contextual understanding of survey responses, making them ideal for detecting subtle textual non-conformities. Thanks to its self-attention mechanism, the model is able to assess the relevance of each word in context, improving accuracy over traditional methods.

Language Models for Textual Non-Conformity Detection

In survey analysis, Transformer-based language are especially useful for detecting anomalies expressed in free-text form, where rule-based systems often fail to grasp implicit signals of non-compliance.

Fine-tuning such models, such as **CamemBERT**[13], on domain-specific data allows it to align closely with the structure and semantics of compliance checklists, leading to an improvement in the detection accuracy.

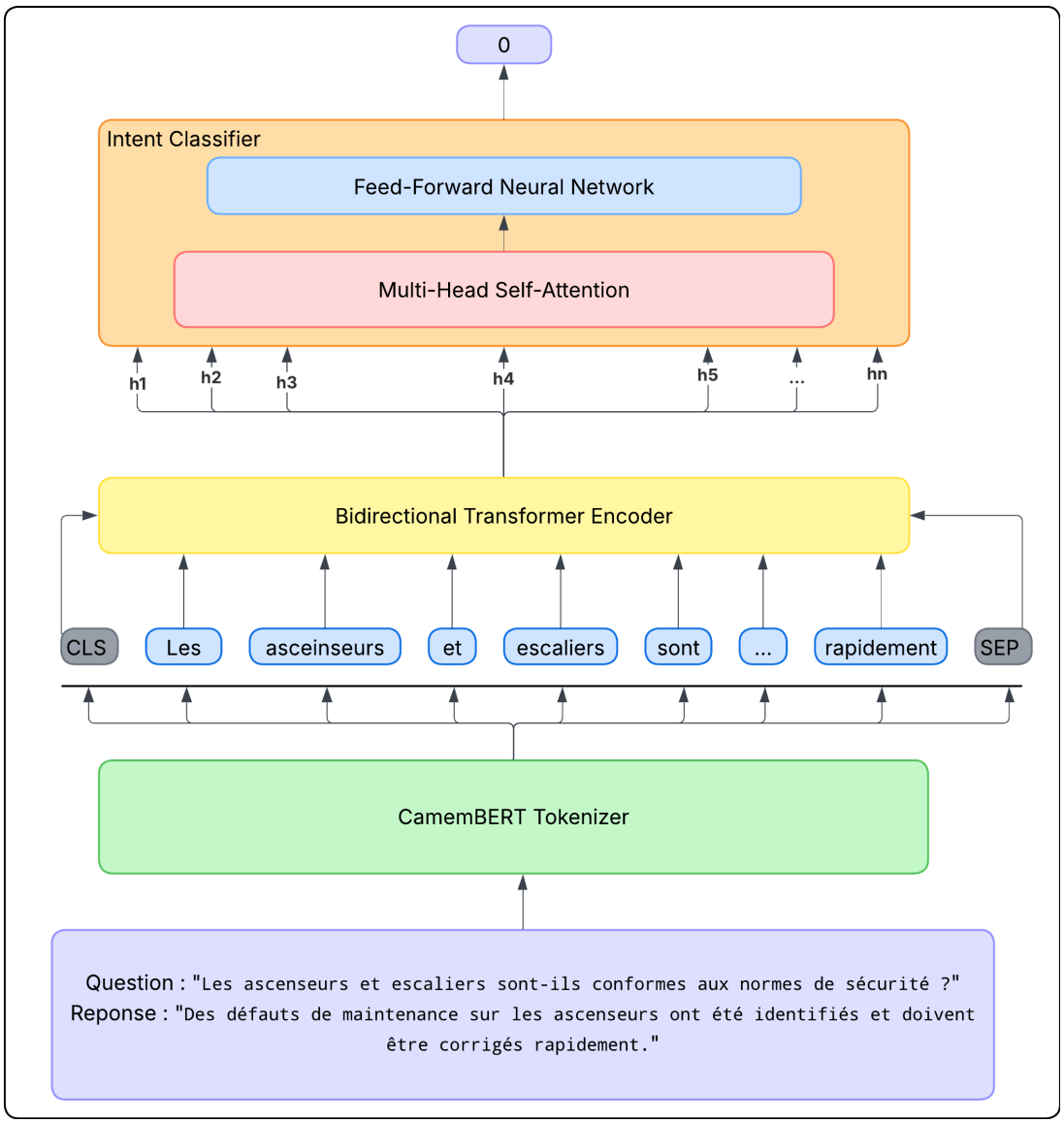


Figure 2.3: CamemBERT Architecture

The figure 2.3 shows the proposed architecture for **CamemBERT** after Fine-Tuning.

2.3.2 MultiModal AI

A multimodal [4] model is a ML model that is capable of processing information from different modalities, including images, videos, and text. Figure 2.4 shows how the difference between Unimodal vs Multimodal Ai models, their input and output types.

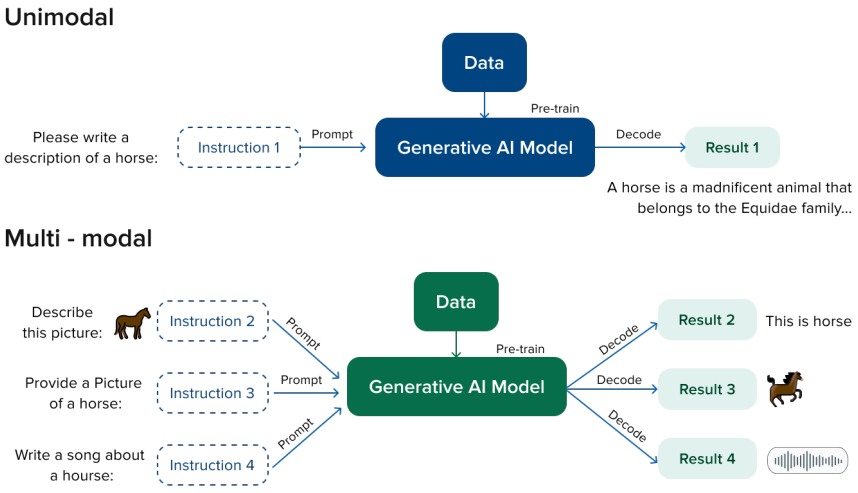


Figure 2.4: Uni Modal Vs. Multi Modal

MultiModals for Visual Non-Conformity Detection

Recent advances in multimodal models such as **Gemma3**[6], **Llava**[12], and similar architectures have demonstrated strong performance in tasks requiring simultaneous interpretation of images and text.

By attending to both visual cues (e.g., product placement, cleanliness) and linguistic context (e.g., descriptive task, justifications), such models are able to identify non-conformities with high precision. This capability makes them highly relevant for real-world applications in compliance audits and retail execution monitoring.

Figure 2.5: Visual Non Conformity Detection

The figure 2.5 shows how a **MultiModal Ai**, a VLM[8] in this case, can be utilized in visual non-conformity detection.

2.4 Agents and Multi-Agent Systems

A Multi Agent System,**(MAS)**[9], consists of multiple agents, each could be accompanied by an AI model, working collectively to perform tasks on behalf of a user or another system.

Each **agent** within a MAS has individual properties and a set of **Tools** which can be interpreted as functions or tasks that it can call or execute, but all agents behave collaboratively to lead to desired global properties.

The figure 2.6 shows three of the most common architectures for Multi Agent Systems.

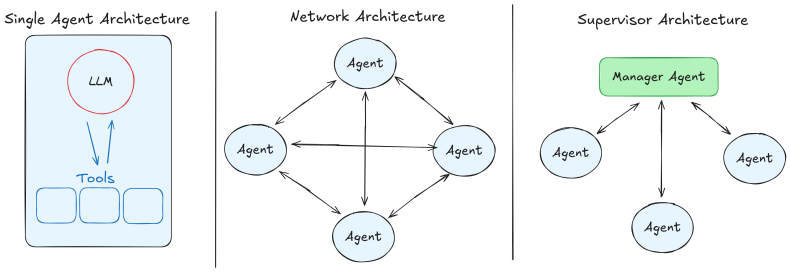


Figure 2.6: Multi Agent Systems Architecture

The table 2.1 provides an overview of three Multi Agent System architectures under consideration for this project.

Architecture	**Description**
Single Agent Architecture	A single agent powered by a large language model (LLM) interacts with a set of tools to perform all tasks. It is simple to implement but can be limited in scalability and task specialization.
Network Architecture	A decentralized system where multiple agents communicate directly with one another. This allows for collaboration and redundancy but can lead to complexity in coordination and potential message conflicts.
Supervisor Architecture	A hierarchical model where a central Supervisor Agent coordinates the tasks among specialized agents. It offers better control, modularity, and task delegation, making it well-suited for scalable, structured workflows.

Table 2.1: Comparison of MAS Architectures

In the context of this project, survey data spans multiple formats: text, images, and structured responses, each requiring specialized processing. No single model can effectively handle all these types at once without compromising accuracy or maintainability. This complexity calls for

a modular approach, where each task is delegated to a dedicated agent. Following a **Supervisor Architecture** enables this decomposition, allowing specialized agents to process specific data types while a supervisor agent ensures data preprocessing, coordination and integration across the system.

Model Context Protocol

The Model Context Protocol (MCP) is an open standard that enables AI Agents to access external tools, data sources, and services in a standardized, modular way.

In our system, MCP acts as the communication backbone: The Supervisor Agent can use it to interact with external agents and tools that are foreign from its ecosystem.

2.5 Conclusion

This chapter introduces a modular, agent-based AI system for automating non-conformity detection in structured survey reports. By combining transformer and multimodal models within a Multi Agent Architecture, the system enables specialized agents to handle diverse data types, while a supervisor agent coordinates their interaction to ensure accurate and interpretable results.

Chapter 3

Proposed Solution

3.1 Introduction

In the following sections, we outline the proposed MAS Architecture, the development stack that enabled us to orchestrate the system, including the programming language, libraries, frameworks, runtime infrastructure, and version control tools.

3.2 Proposed Solution Overview

The proposed system adopts a `Supervisor Architecture` designed to automate non-conformity detection in survey data.

The MAS consists of specialized agents, each responsible for processing different types of survey entries:

- **Supervisor Agent:** Coordinates the workflow by managing survey traversal and dispatching each entry to the appropriate agent based on its type.
- **Textual Agent:** Handles textual survey entries, performing compliance detection.
- **Visual Agent:** Processes photo-based entries to detect anomalies via multimodal analysis.
- **Generic Agent:** Manages structured survey data (e.g., multiple choice, ratings) by applying rule-based compliance logic.

This modular design allows scalable and maintainable processing of diverse survey data while enabling future extension to additional modalities and functionalities.

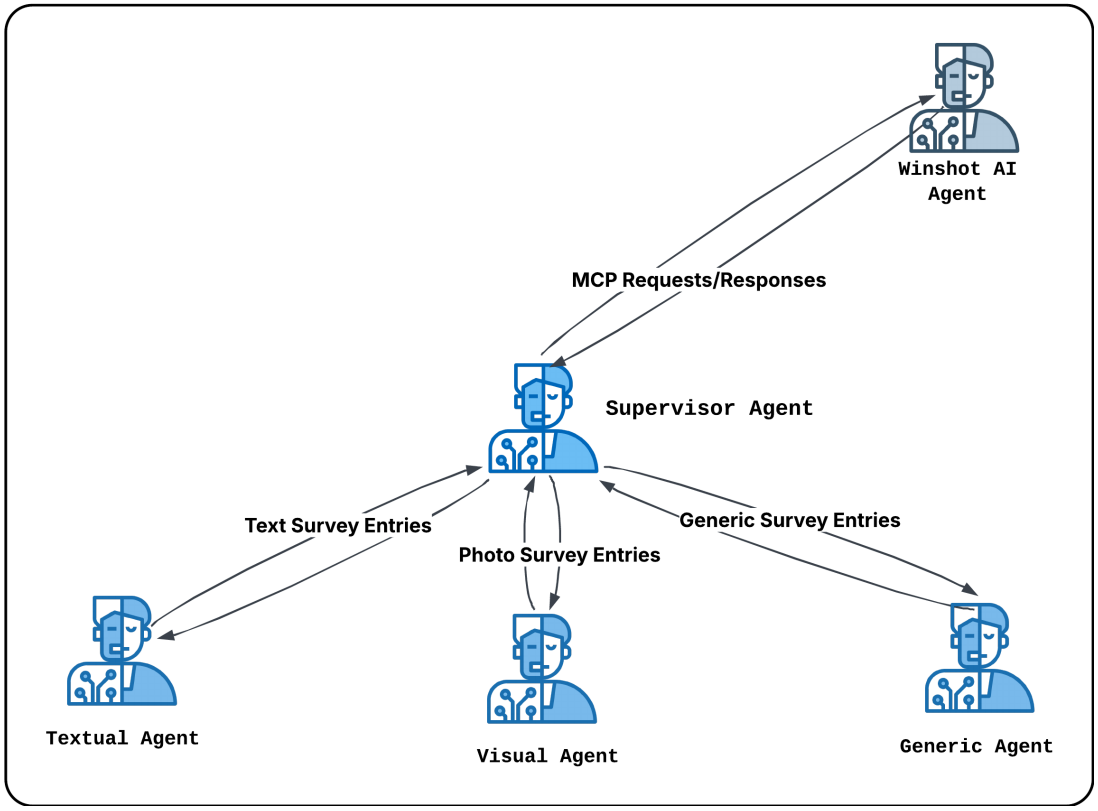


Figure 3.1: Proposed Multi-Agent System Architecture

The workflow in Figure 3.1 begins with the **Supervisor Agent** receiving an MCP request from Winshot AI Agent. This request contains raw survey data that serves as the primary input for the system.

Upon receiving the request, the Supervisor Agent performs several preprocessing steps, including: Data validation, cleaning, and formatting.

After preprocessing, the Supervisor Agent inspects the `questionType` of each survey entry and dispatches it to the appropriate specialized agent. Each agent then processes its assigned entries by evaluating compliance, either through a rule-based approach or using specialized model inference.

Upon completion, the agent assigns a compliance attribute to the survey entry and returns the validated entry back to the Supervisor.

After collecting all the processed entries, the Supervisor Agent:

- Reconstructs the original survey structure.
- Embeds the validated and annotated results.
- Sends the validated dataset back to Winshot AI Agent via an MCP response.

3.3 Development Environment

To build this workflow, we must intelligently choose the right development environment, as it is considered the manufacturing factory for this system. In which the models are carefully chosen, trained, linked to their accompanying agents, and data are properly cleaned, processed, and fed to the models.

By carefully selecting the right development environment, we can create a productive and efficient workspace that facilitates the creation of the MAS.

3.3.1 Programming Language: Python

Python serves as the foundation for this project, leveraging its well-established position in the world of data science and machine learning. Python's readability and extensive libraries make it an ideal choice for rapid development and clear code structure. Figure 3.2 shows Python Logo.

Figure 3.2: Python Logo

3.3.2 Agent Framework

Tool	Purpose	Logo
LangGraph	A graph-based agent orchestration framework used to model and manage multi-agent workflows efficiently.	LangGraph

Table 3.1: Agent Framework

3.3.3 Deep Learning Framework

Tool	Purpose	Logo
PyTorch	Used for building and training deep learning models, handling tensors, and optimizing GPU-accelerated workloads.	PyTorch

Table 3.2: Deep Learning Framework

3.3.4 Python Libraries

Tool	Purpose	Logo
Pandas	Used for data exploration, cleaning, transformation, and understanding dataset structure for better modeling decisions.	
NumPy	Enabled efficient numerical computations and vectorized operations on large datasets, crucial for deep learning tasks.	
Matplotlib	Used to create visual representations of data such as plots, charts, and graphs to support data analysis.	
Transformers	Provided pretrained models for fine-tuning and inference, enabling agents to perform NLP and vision tasks.	

Table 3.3: Python Libraries Used

3.3.5 Version Control

Tool	Purpose	Logo
Git	Enabled tracking changes, collaboration, and experimentation through branching and merging during development.	
CodeCommit	Cloud-based Git repository hosting with CI/CD integration and secure team collaboration capabilities.	

Table 3.4: Version Control Tools

3.3.6 Integrated Development Environment

Tool	Purpose	Logo
Visual Studio Code	Lightweight development environment used with Jupyter, Git extensions, and for efficient local code development.	

Table 3.5: Integrated Development Environment

3.3.7 Remote Runtime

Tool	Purpose	Logo
Kaggle	Provided cloud-based compute with free GPUs/TPUs for lightweight training and evaluation.	kaggle

Table 3.6: Remote Runtime Environment

3.3.8 Training Environment

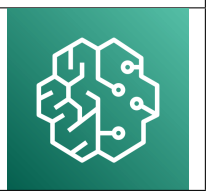
Tool	Purpose	Logo
Amazon SageMaker	Used to train computationally expensive models efficiently using scalable, managed infrastructure.	

Table 3.7: Training Environment

3.4 Conclusion

The success of the Multi-Agent System was driven by careful selection of tools across the tech stack. Key technologies like Python, transformers, Kaggle, And Git enabled efficient development and agentic workflows.

Chapter 4

Data Exploration

4.1 Introduction

This chapter details the **Data Understanding** and **Data Preparation** phases of the CRISP-DM methodology. It covers the dataset's `Structure, Sources, Distribution,` and `Quality`, while addressing key challenges like data scarcity, missing labels, and inconsistencies. Solutions to these issues are outlined, establishing a solid foundation for reliable modeling and agent-specific workflows.

4.2 Exploratory Data Analysis

The Data used in this project was compiled through a combination of real-world and simulated data sources. Prior to the project initialization, data was continuously gathered over a period of 11 months to ensure coverage of various scenarios and workflows. The collection process involved aggregating structured Survey Entries, sorted by type, from existing systems as well as generating synthetic data to complement and enhance the dataset's representativeness.

4.2.1 Data Sources

The dataset used in this project originates from two main sources:

- **Data Provided by the Hosting Company** This includes operational and survey data collected from companies' audit systems. These records represent real surveys and reports, offering insight into actual usage patterns, performance metrics, and feedback.
- **Synthetic Data** To mitigate the gaps in the available data, synthetic records were generated to replicate the structure and properties of production data. This was particularly useful in scenarios where data was insufficient, unlabeled, or unavailable. Synthetic data was

carefully designed to maintain realism and consistency with the actual data while enhancing coverage across different Survey Entry Types.

4.2.2 Data Structure

Our data's structure is similar to a JSON object or a Python dictionary, holding crucial information such as Store Name, Compliance rate, and most importantly, the report, which is a **list** of JSON objects, referred to as **Survey Entry**.


```
[
  {
    "store": "123 : Magasin du Futur",
    "store type": "boutique",
    "Compliance rate": 72.5,
    "report": [
      {
        "question": "Propreté de l'entrée du magasin",
        "questionType": "STAR_RATING",
        "answer": 3,
        "comment": "Quelques traces de pas visibles",
        "photos": [
          "image1.png",
          "image2.jpg"
        ],
        "compliance": 60
      }
    ]
  }
]
```


Figure 4.1: Data Structure as JSON

Figure 4.1 shows an example of the data structure as a JSON object.

Figure 4.2: Data Structure as Graph

Figure 4.2 shows an example of the data structure as a Graph.

Our main focus in the data here are the **Survey Entries**, which contains useful entities such as question, question type, answer, etc.. Based on these entities, the workflow will be designed and orchestrated.

4.2.3 Volume

The Bar Chart in figure 4.3 visualizes the overall data distribution by question type.

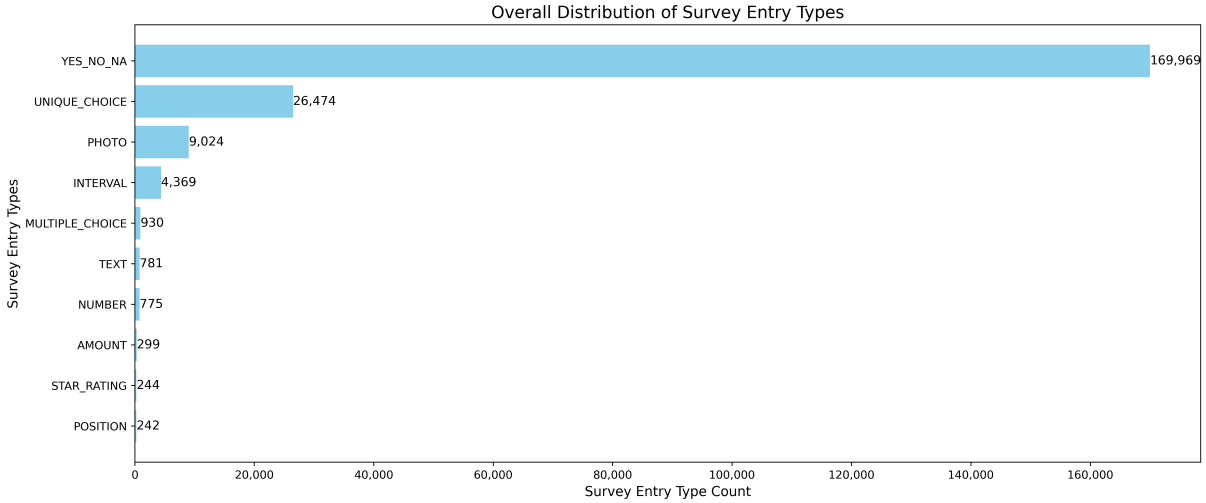


Figure 4.3: Overall Distribution of Survey Entry Types

Survey Entry Type	Number of Instances
YES_NO_NA	170,000
UNIQUE_CHOICE	26,500
PHOTOS	9,000
INTERVAL	4,500
MULTIPLE_CHOICE	930
TEXT	780
NUMBER	775
AMOUNT	300
STAR_RATING	250
POSITION	250

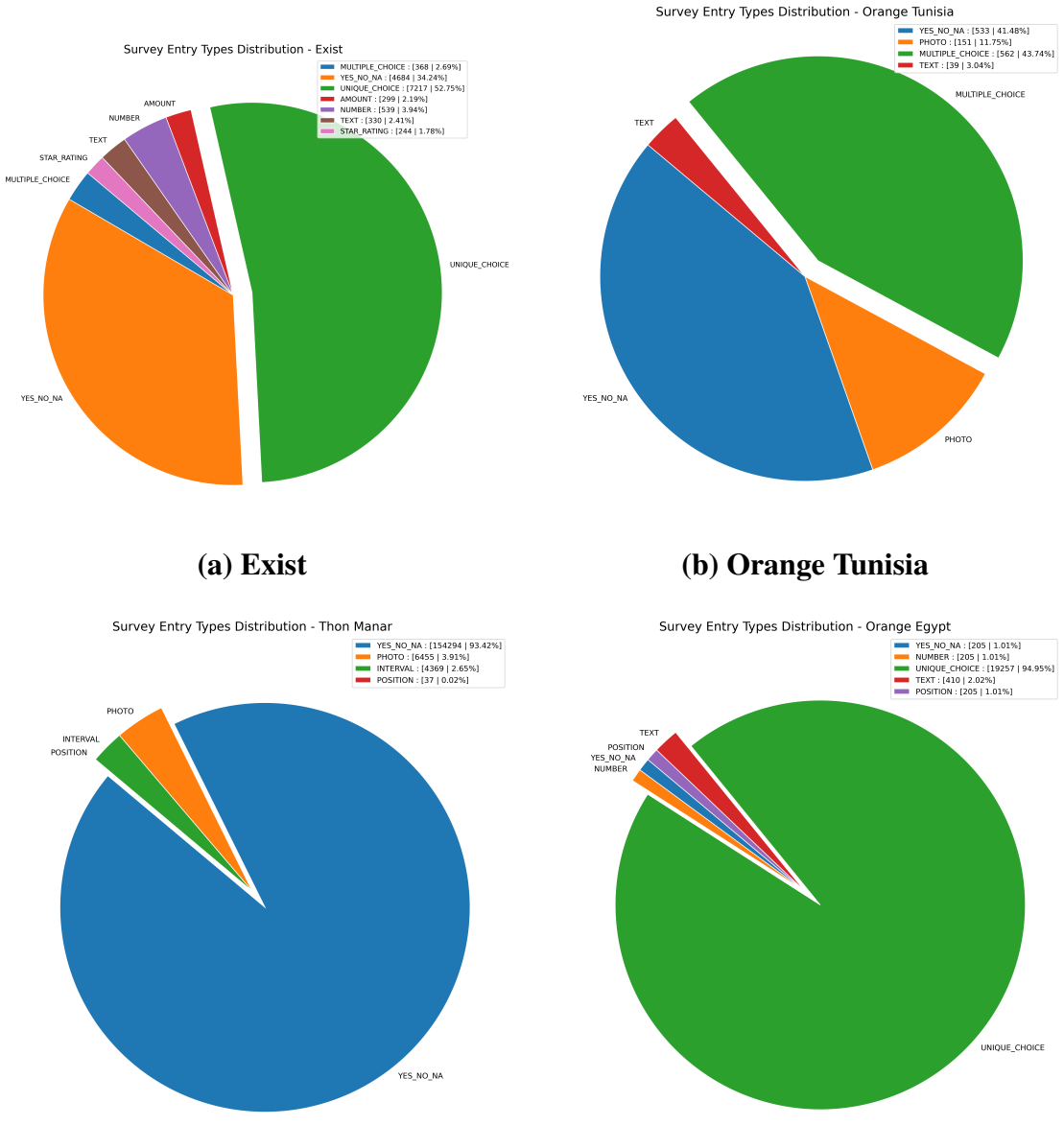
Table 4.1: Distribution of Survey Entry Types

As shown in Table 4.1, the dataset presents a markedly uneven distribution across question types. A small number of formats dominate the dataset, while others appear only sparsely. This **long-tail** distribution introduces a class imbalance that may affect model performance, particularly for underrepresented types.

4.2.4 Data Distribution

This section presents the distribution of survey entries across Winshot's client base. To effectively illustrate this, a pie chart has been utilized, where each slice represents the proportion of different survey entry types within the overall dataset for each client.

This visual breakdown provides a clear view of how survey data is spread across various entry types, highlighting key strengths and weaknesses or imbalances in data contribution among clients.



(a) Exist **(b) Orange Tunisia**

(c) Thon Manar **(d) Orange Egypt**

Figure 4.4: Survey Entry Type Distribution across Clients

The distribution of survey entry types across Winshot's client base, as illustrated in Figure 4.4, reveals a notable imbalance in data contributions among different clients. Each client represents a distinct industry sector such as Orange Tunisia and Orange Egypt in the telecommunication

and technology sector, and Exist in the fashion and clothing industry. This sector-based variation in data input brings both opportunities and risks.

A closer examination of the data at the individual client level reveals that each client has a distinct distribution, often dominated by a particular type of survey entry. While this proficiency provides valuable, in-depth data for specific entry types, it also highlights a key limitation: no single client contributes a well-balanced dataset encompassing a diverse range of survey entry types. This situation presents a double-edged sword offering depth within isolated categories but lacking the breadth needed for holistic system training.

To address these challenges, the upcoming sections and chapter will detail the methods and techniques we employed to ensure that the system generalizes effectively across diverse modalities and question formats, maintaining both robustness and reliability regardless of the client or domain.

4.3 Data Quality Assessment

This section of the Data Analysis Chapter provides an assessment of the data quality, the issues we faced, and the techniques we used to help mitigate the issues.

4.3.1 Issues

As discussed in subsection 4.2.3 and subsection 4.2.4, the dataset exhibits a significant imbalance in the overall distribution of survey entries.

In addition to this, the `compliance` attribute used as an indicator of whether a survey entry meets the expected standard also shows a marked skew.

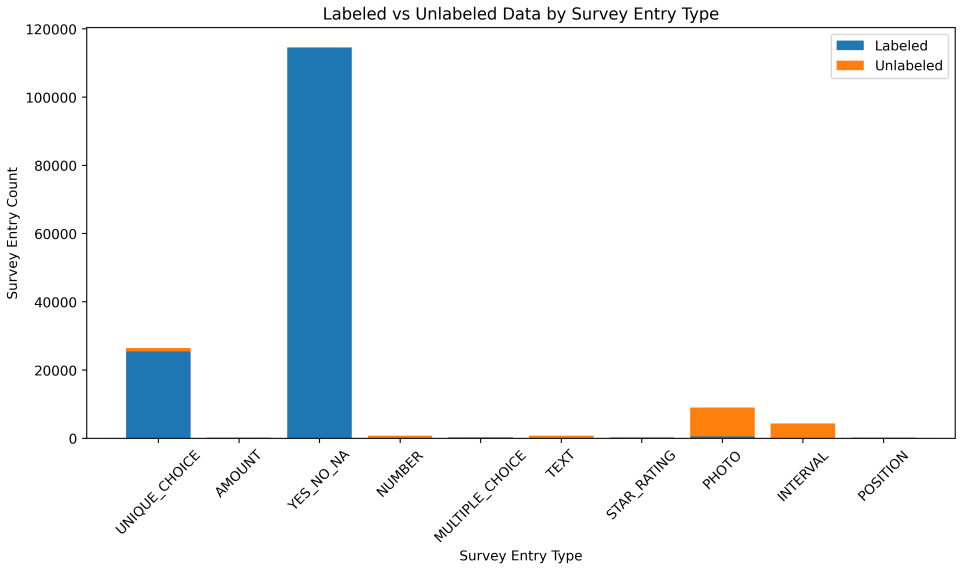


Figure 4.5: Labeled VS. Unlabeled Distribution

This imbalance in validation outcomes can impact the learning process, especially in supervised models that rely on balanced class representation. The stacked bar chart in Figure 4.5illustrates the distribution of compliance levels across different survey entry types, further emphasizing the disparity in class frequencies.

4.3.2 Challenges and Mitigations

Several challenges were encountered during the data analysis phase.

1. The dataset exhibited significant class imbalance, both in terms of Survey Entry type distribution and the `compliance` attribute which can bias model training toward dominant classes.

2. The heterogeneity of Survey Entry formats (e.g., binary choices, free text, images, numerical scales) required modality-specific preprocessing pipelines and increased system complexity.

3. Inconsistencies and noise in the data, such as ambiguous answers, null comments, or low-quality image entries, necessitated extensive data cleaning and validation procedures.

Additionally, interpreting compliance labels without explicit ground truth posed a challenge for supervised learning, as many entries relied on subjective human validation. These issues collectively underscore the need for careful data preparation and robust evaluation strategies in the downstream workflow.

To address the challenges outlined above, several strategies were implemented across different stages of the data preparation and modeling pipeline:

1. Handling Class Imbalance in Survey Entry Types

The pronounced imbalance in survey entry types, particularly the under-representation of `TEXT` entries, was treated as a data augmentation challenge. Multiple approaches were explored to generate synthetic `TEXT`-type survey entries:

- **Image Captioning-Based Augmentation:** One approach involved extracting `PHOTO`-type entries, captioning the associated images using vision-language models (VLMs), and repurposing those captions as textual answers to the original questions.

  This would effectively transform a visual entry into a `TEXT` entry. However, the generated captions were often inconsistent or not directly relevant to the original question, which made the resulting entries unsuitable. Despite its limited success, this experiment provided valuable insights into the use of VLMs such as Florence2, BLIP2, and the use of Open Ai's CLIP model to compute Similarities between images and texts.

Figure 4.6 describes the captioning pipeline used to create TEXT Survey Entries.

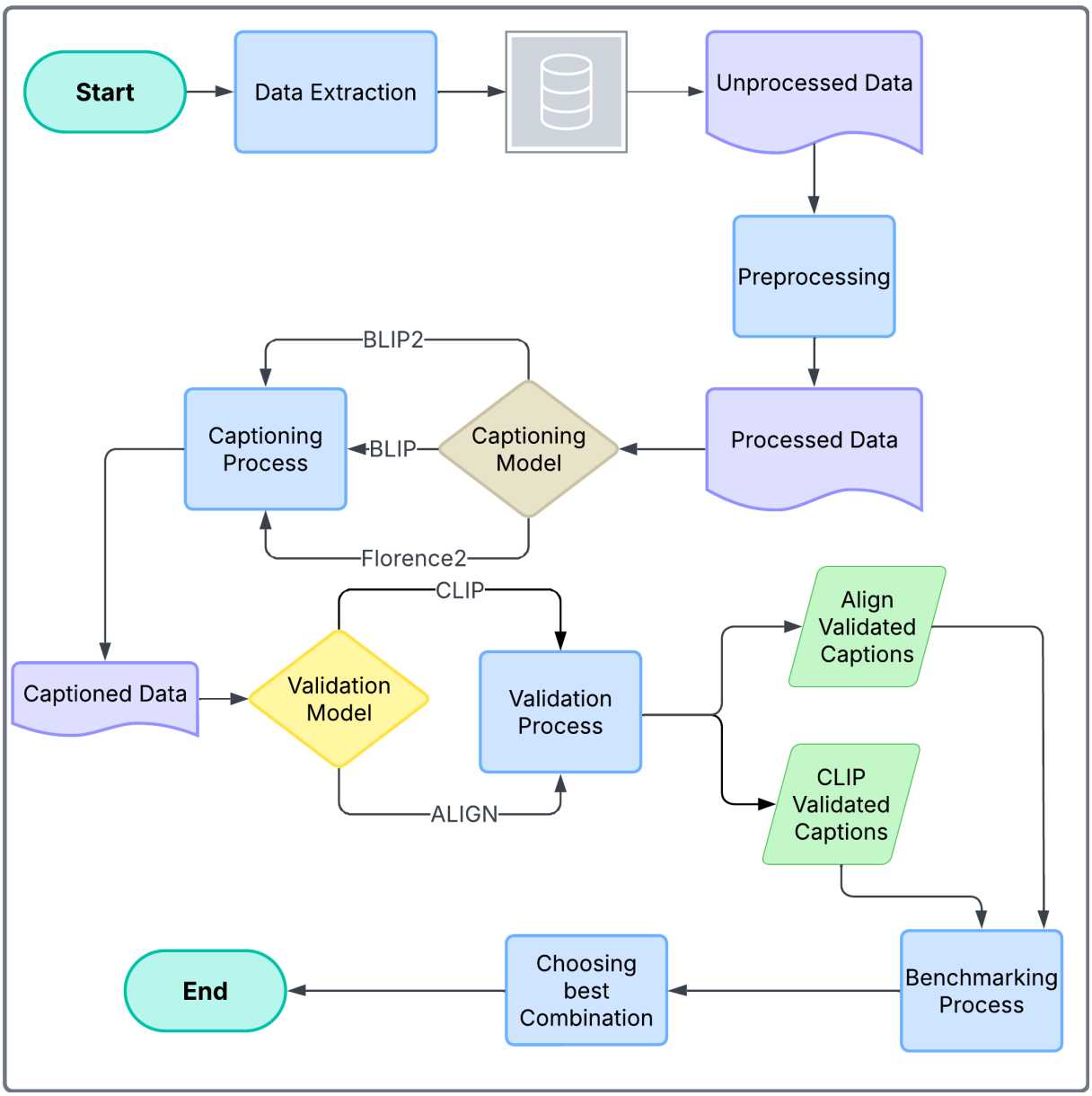


Figure 4.6: Captioning Pipeline

- **LLM-Based Rephrasing:** A more successful strategy consisted of using Large Language Models to rephrase existing answers from structured formats, such as `YES_NO_NA`, `UNIQUE_CHOICE`, and `MULTIPLE_CHOICE` into open-ended responses. The original question was retained, and the LLM-generated answer was used to construct a new `TEXT`-type entry. This method not only produced syntactically rich examples but also preserved semantic integrity.

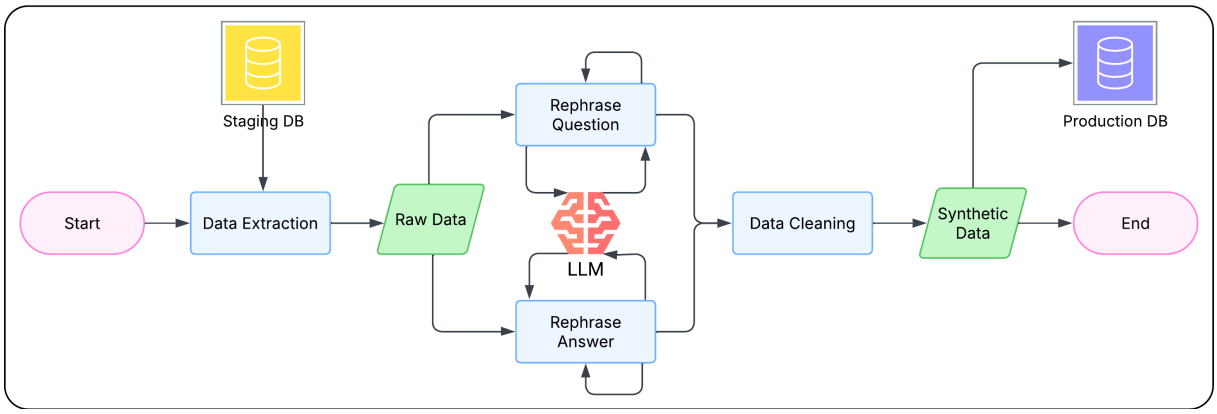


Figure 4.7: Rephrasing Workflow

Figure 4.7 presents the rephrasing workflow used to create synthetic `TEXT` and `PHOTO` Survey Entries.

- **Synthetic Text Generation:** Building on the success of LLM prompting, tools like ChatGPT and Gemini were employed to generate entirely synthetic survey entries of type `TEXT`. Through careful prompt engineering, approximately 200 high-quality entries were created, enabling scalable data augmentation and improved model generalization on free-text inputs.

2. Managing Heterogeneity of Survey Entry Types

To accommodate the wide variety of survey entry formats, a modular architecture was proposed. Each entry type will be processed by a specialized or generic agent based on its structure:

- `PHOTO` and `TEXT` entries are handled by dedicated agents equipped with vision and language understanding capabilities, respectively.
- Structured types such as `YES/NO/NA`, `MULTIPLE CHOICE`, and `AMOUNT` are delegated to a generic agent. These entries often include predefined answer-weight mappings that facilitate automatic compliance assessment without the need for deep semantic analysis.

This separation of concerns provides a clearer overview of the MAS structure, ensuring that each agent is optimized for the specific modality and data structure it handles, enhancing both interpretability and performance.

3. Data Quality and Preprocessing

Manual labeling and systematic preprocessing techniques were applied to mitigate noise and inconsistencies across the dataset. Key steps included:

- **Missing Value Handling:** Incomplete entries were either imputed or excluded based on their relevance, value, and impact on training.
- **Standardization of Format:** Text fields were cleaned, normalized, and structured consistently to facilitate tokenization and downstream analysis.
- **Duplicate Detection and Merging:** Identical or near-duplicate entries often resulting from human error or repeated submissions were identified and consolidated.
- **Questions Rephrasing:** Ambiguous questions and/or answers were rephrased either manually or with the aid of LLMs to have clear and meaningful training data.

These mitigation strategies formed the foundation for a robust, scalable, and multimodal non-conformity detection system. Together, they enabled more balanced model training and laid the groundwork for agentic workflows.

4.4 Non-Conformity Analysis

Within the context of this project, non-conformities or anomalies are simply any form of non-compliance, unexpected responses, "wrong" responses in some cases, or a miscarried task, etc.

4.4.1 Contextual Indicators and Observations

Survey authors, upon designing the forms to launch, will place "weights" to certain questions and the expected answers, and a "compliance" metric based on their specifications and preferences.

These combinations of weights and compliance attributes help us detect the anomalies from the survey reports.

Survey entries can fall into any of the following categories :

The Bar Chart if Figure 4.8 shows the distribution of compliance rates across all Winshot's clients.

Analysis of the compliance rate distribution across clients reveals notable variability in performance. As illustrated in Figure 4.8, **Lnko** exhibits the highest average compliance rate at **96.3%**, followed by **Thon el Manar** with **91.34%**, and **Orange Tunisia** with a comparatively lower rate of **60.3%**.

Survey Entry Class	Example	Anomaly Indicator
Task Execution	"You are tasked with <TASK>. Make sure this task is completed successfully. Report this with an image showcasing a successful or unsuccessful execution."	Poorly executed task or missing/irrelevant image (for `PHOTO` question types).
Compliance Check	"Is the store layout following the brand guidelines? (YES/NO)."	Negative or unexpected response (e.g., `NO`) where `YES` is weighted as fully compliant (100) and `NO` as non-compliant (0).
Inventory and Stock Verification	"Are all required products [<PRODUCT_1>, <PRODUCT_2>...] available on shelves? If NO, specify missing items."	Indication of missing products or restocking issues in `YES_NO_NA` or free-text entries.

Table 4.2: Survey Entry Classes and Corresponding Anomaly Indicators

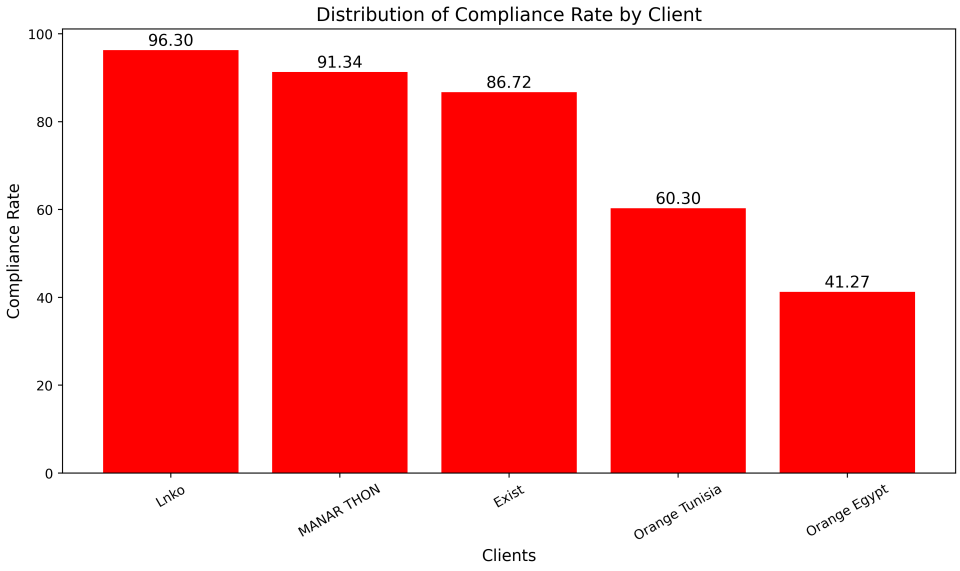


Figure 4.8: Compliance Rate Distribution

4.5 Conclusion

In this chapter, we laid the groundwork for the multi-agent system by tackling data understanding and preparation. We highlighted challenges like class imbalance, diverse entry types, data annotation, and addressed them through synthetic data generation, LLM-based augmentation, and tailored workflows.

Chapter 5

Modeling and Evaluation

5.1 Introduction

This chapter examines the proposed MAS architecture, its development process, and subsequent evaluation. We'll delve into the chosen Models, their fine-tuning process, and their evaluation.

5.2 Architecture

This section provides a detailed overview of the architecture of the Multi-Agent System, designed to automate survey compliance analysis. The architecture follows a modular agent-based approach to ensure scalability, flexibility, and task specialization.

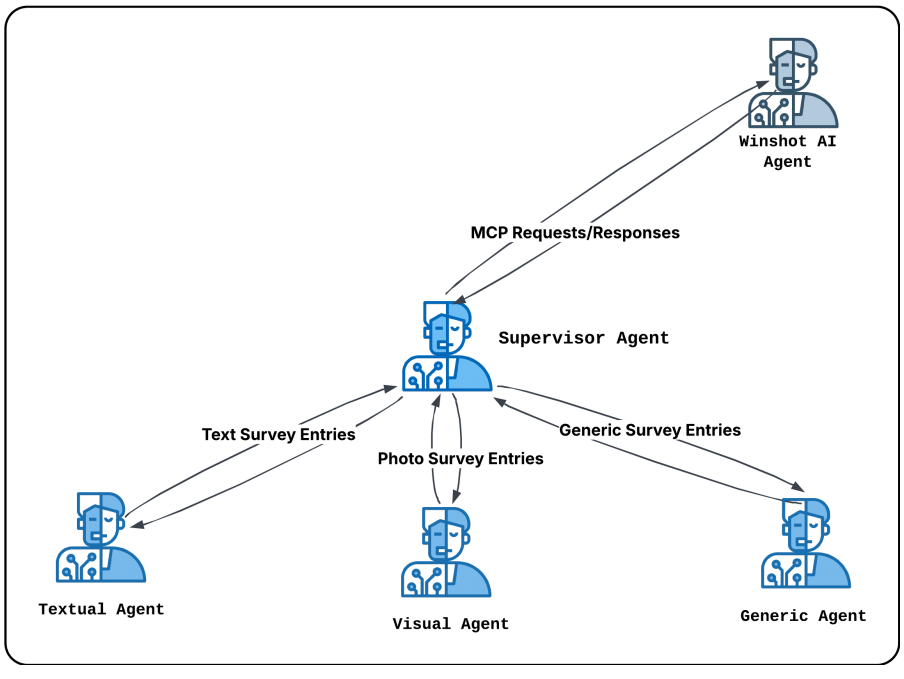


Figure 5.1: Proposed Multi Agent System Architecture

5.2.1 Multi-Agent Design

The system adopts a Multi-Agent System architecture to handle non-conformity detection across diverse survey entry types efficiently.

Each agent specializes in a specific modality: text, image, or generic input. At the core lies the Supervisor Agent, which handles multiple responsibilities such as pre-processing Survey Entries and distributing each for the appropriate agent.

This design promotes modularity, simplifies maintenance, and allows for seamless integration of additional agent types in the future.

5.2.2 Supervisor Agent

The Supervisor Agent serves as the coordinator for the multi-agent system. Its primary role is to receive MCP (Model Context Protocol) requests from **Winshot AI Agent**, parse the contained data, and orchestrate the dispatch of each individual survey entry to the appropriate specialized agent.

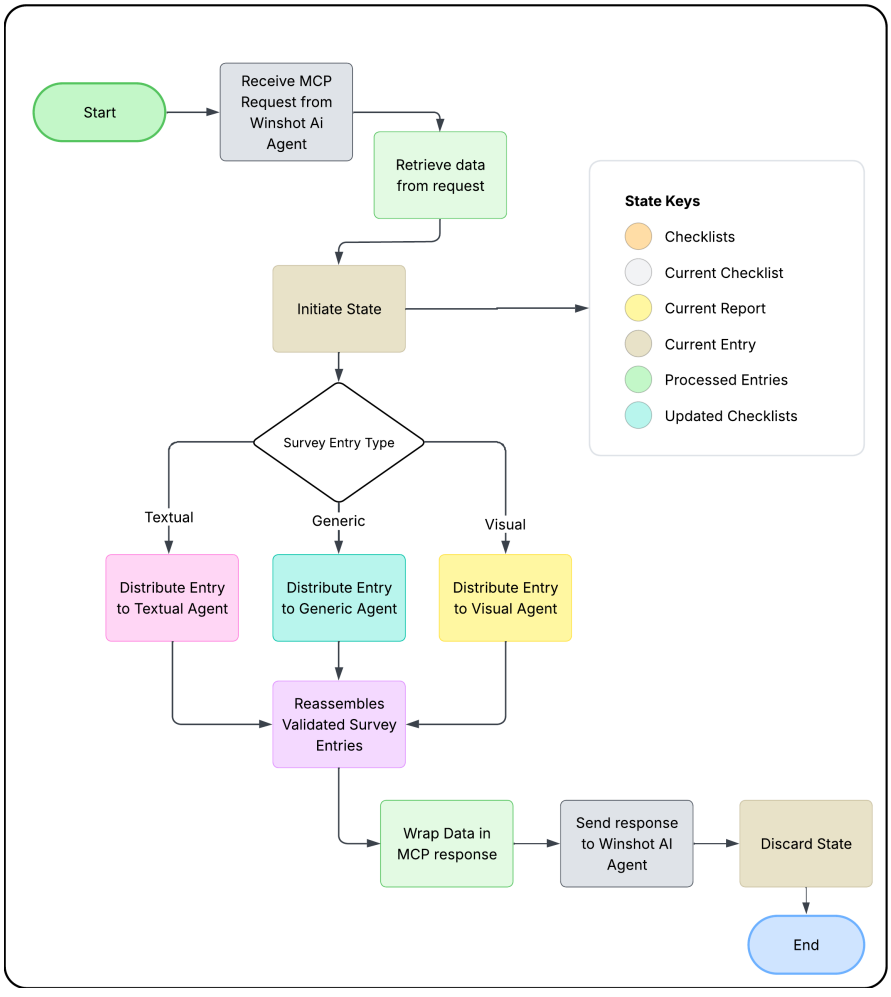


Figure 5.2: Supervisor Agent Workflow

At each step, the Supervisor Agent:

- Parses and retrieve data from the received request and tracks the current survey and survey entry via the `Graph State`.
- Dispatches entries based on their `questionType` using a dynamic routing function.
- Reassembles processed entries back into their original survey structure.
- Send validated data within MCP response to Winshot AI Agent.

As shown in Figure 5.2, To track progress, the Supervisor Agent relies on a mutable state structure, which acts as a dynamic checklist throughout the workflow and includes fields such as `current_survey`, `current_entry`, and `processed_entries`.

These fields help the agent determine which entries have been processed, which remain, and when to transition between different workflow states (e.g., initiate, distribute, discard).

5.2.3 Textual Agent

The Textual Agent is responsible for handling `TEXT` and `YES/NO/NA` entries. Its pipeline includes:

- Rephrasing and/or Elaboration of questions and answers using prompt-based tool calling, done through the **Gemma3-1B** Model.
- Compliance prediction using a fine-tuned transformer model, CamemBERT.
- Generation of human-readable justifications and actionable recommendations in cases of non-compliance.

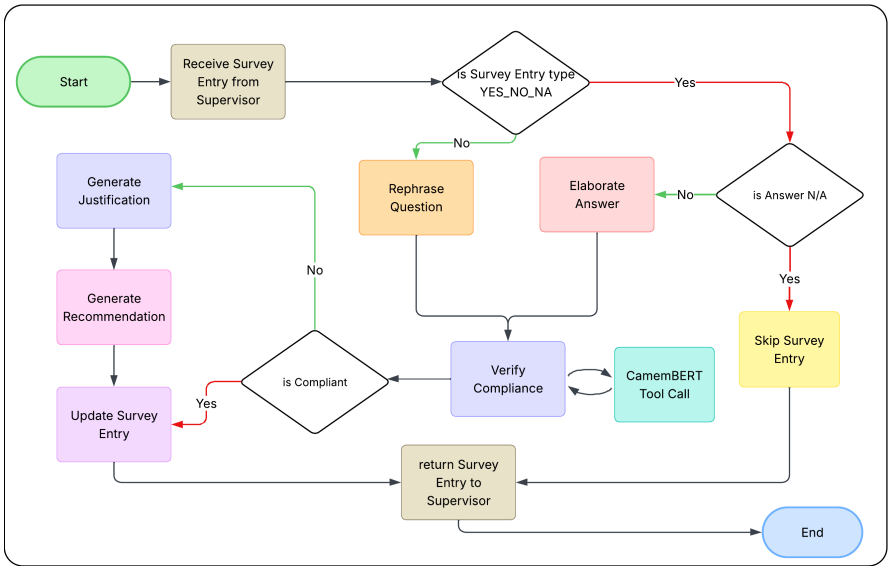


Figure 5.3: Textual Agent Workflow

The Flowchart in Figure 5.3 highlights the Textual Agent workflow.
This agent combines semantic understanding and generative tasks to enrich textual survey entries with interpretability and decision support outputs. All predictions are logged in the shared state as structured fields: `detectedCompliance`, `justification`, and `recommendation`.

5.2.4 Visual Agent

The Visual Agent is designed to process entries of type `PHOTO`. Although not fully implemented at this stage, it is intended to:

- Ingest image-based evidence from task execution or store monitoring.
- Perform visual compliance analysis using the accompanied multimodal modal Gemma3-4B.
- Provide visual justifications or and actionable recommendations for anomalous content.

This agent will extend the MAS toward multimodal processing, ensuring that photo-based survey tasks are evaluated with the same semantic depth as textual ones.

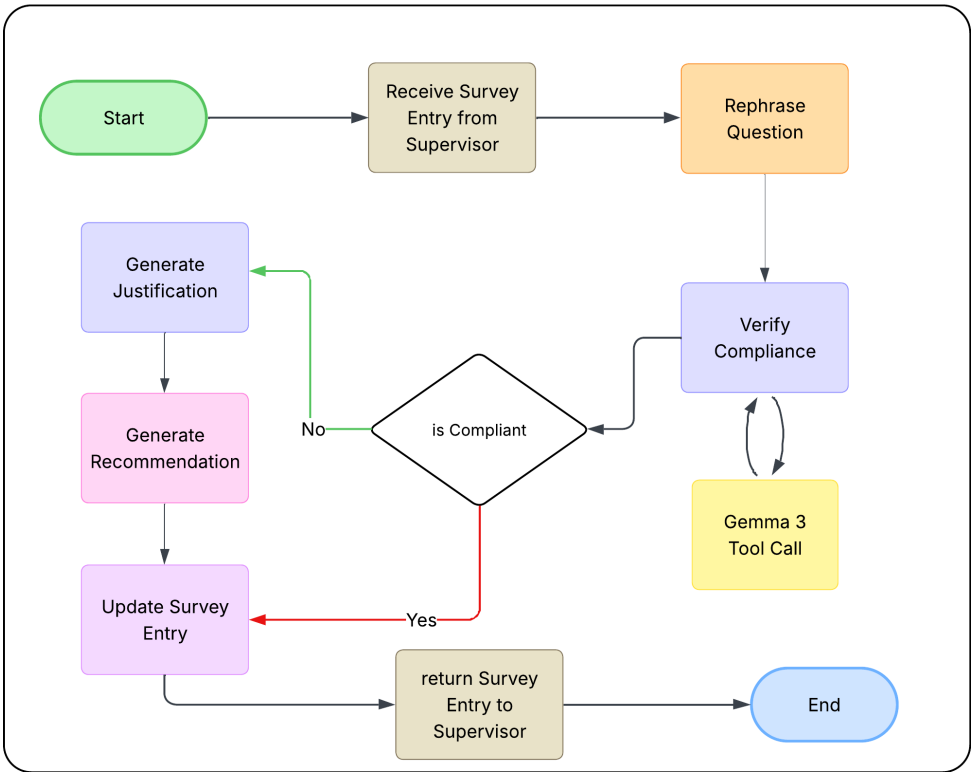


Figure 5.4: Visual Agent Workflow

The Flowchart in Figure 5.3 highlights the Visual Agent workflow.

5.2.5 Generic Agent

The Generic Agent is tasked with handling all remaining structured Survey Entry types such as:

- `MULTIPLE CHOICE`, `UNIQUE CHOICE`
- `STAR RATING`, `POSITION`, `INTERVAL`, `NUMBER`, `AMOUNT`

For these, it applies logic-based compliance inference, such as:

- Weighted answer mapping for choice-based types.
- Distance thresholds for geolocation (`POSITION`).
- Rating-to-score mapping for star evaluations.

In cases of non-compliance, it generates contextual justifications and recommendations using prompt-driven reasoning, ensuring equal evaluation for all Survey Entry types.

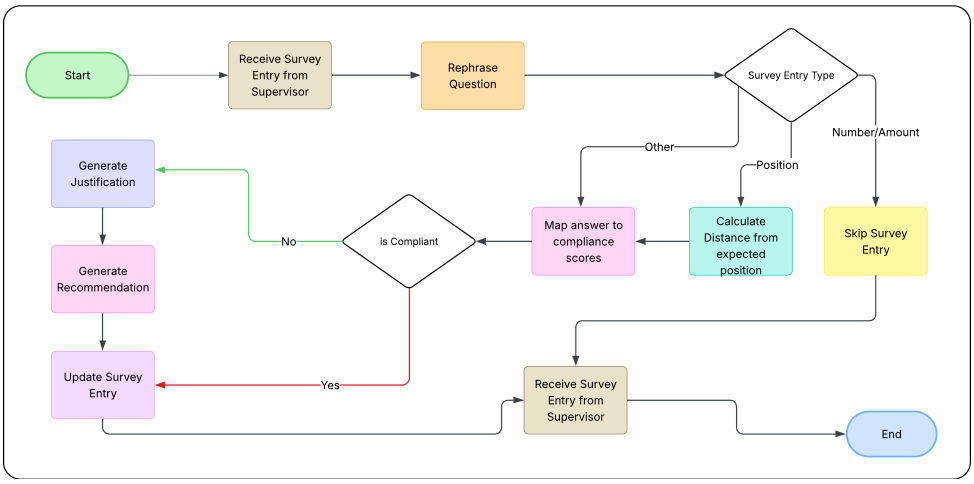


Figure 5.5: Generic Agent Workflow

The Flowchart in Figure 5.3 highlights the Generic Agent workflow.

5.3 Models Fine-Tuning

Fine-Tuning Definition

According to IBM, Fine-Tuning[10] in machine learning is the process of adapting a pre-trained model for specific tasks or use cases. It is a subset of transfer learning, leveraging the knowledge an existing model has already acquired to learn new tasks more efficiently.

The goal of fine-tuning is to reduce the computational resources and labeled data required to tailor large models to specific applications and business needs. By refining a pre-trained model

with domain-specific data, we can customize sophisticated AI models without the need to train them from scratch.

Fine-Tuning Techniques

Type	Description	Use Case
QLoRA	A memory-efficient fine-tuning method that uses Quantization[3] and low-rank adapters[11] to adapt large models. It freezes most of the model and inserts small, trainable LoRA adapters, using 4-bit quantization to save memory.	Fine-tuning Gemma3-4B.
Feature Extraction	Use a pre-trained model as a fixed feature extractor, training only the output layer for the specific task, while the rest of the layers are frozen.	Fine-Tune CamemBERT.

Table 5.1: Fine-Tuning Techniques

Table 5.1 describes the Fine-Tuning techniques implemented in this project.

5.3.1 CamemBERT for Textual Non-Conformity Detection

To detect non-conformities in textual survey responses, we fine-tuned CamemBERT. The model is trained to classify whether a Survey Entry is compliant or non-compliant based on the associated question and answer.

Training and Evaluation Environment

We opted for **Kaggle's Notebook** for Training and Evaluation, as one Single Notebook offers:

- 12 Hours access to 2x Nvidia'a T4 GPU, equivalent to 32GB of VGPU with parallel processing.
- 30GB of CPU RAM.
- 20GB of Output Storage Size.
- 57GB of Input Storage Size.

This environment is well-suited for our use-case, enabling both data preparation and model evaluation to be performed in one place. Moreover, the CamemBERT model is relatively lightweight, requiring no more than **8GB of VRAM**, making it highly compatible with the resources available on Kaggle.

Dataset Overview

The dataset consists of annotated survey entries with the following structure:

Attribute	Description
Columns	question, answer, label
Label	Binary value where `1` indicates compliance, and `0` indicates non-compliance
Initial Size	600 entries
After Augmentation	1021 entries
Training Split	80% training, 20% evaluation

Table 5.2: Summary of the dataset used in the study

Training Arguments

Argument	Value	Explanation
`num_train_epochs`	5	Number of complete passes over the training set.
`per_device_train_batch_size`	16	Number of examples per batch during training.
`per_device_eval_batch_size`	16	Number of examples per batch during evaluation.
`evaluation_strategy`	epoch	Evaluation is triggered after each training epoch.
`metric_for_best_model`	Recall	Model selection criteria.

Table 5.3: CamemBERT Fine-Tuning Hyperparameters

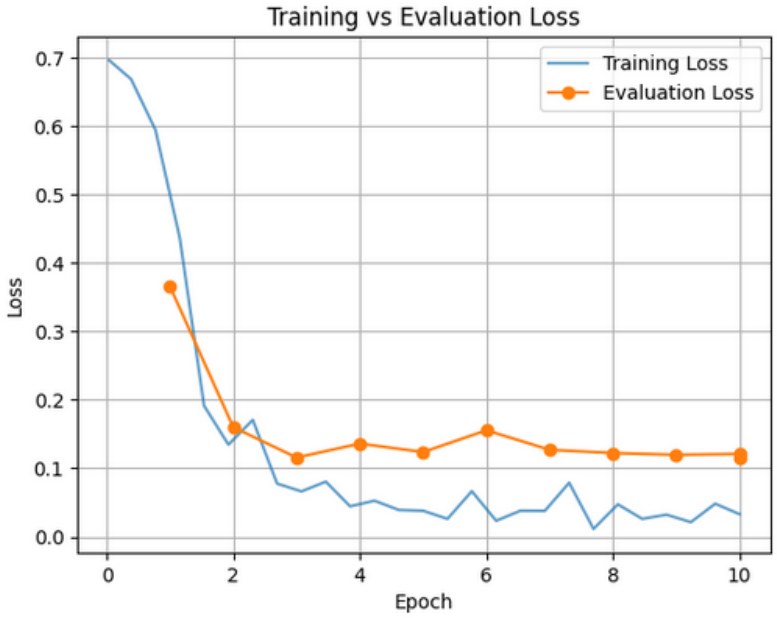


Figure 5.6: Training vs. Evaluation Loss

Figure 5.6 shows the Training vs. Evaluation Loss across 10 epochs. The evaluation loss stabilizes after the 7th epoch and becomes nearly constant from epoch 8 onward, suggesting that the model has reached a generalization plateau without overfitting.

Evaluation Metrics

The following metrics are computed during evaluation:

- **Accuracy:** The ratio of correctly predicted entries over the total number of entries.
- **Recall:** The ability to detect all true non-compliant entries.
- **Precision:** The ratio of correctly identified non-compliant entries.
- **F1-Score:** The harmonic mean of precision and recall; balances false positives and false negatives.

Training Results

All training and evaluation results are covered in Section 5.5.

Model Architecture

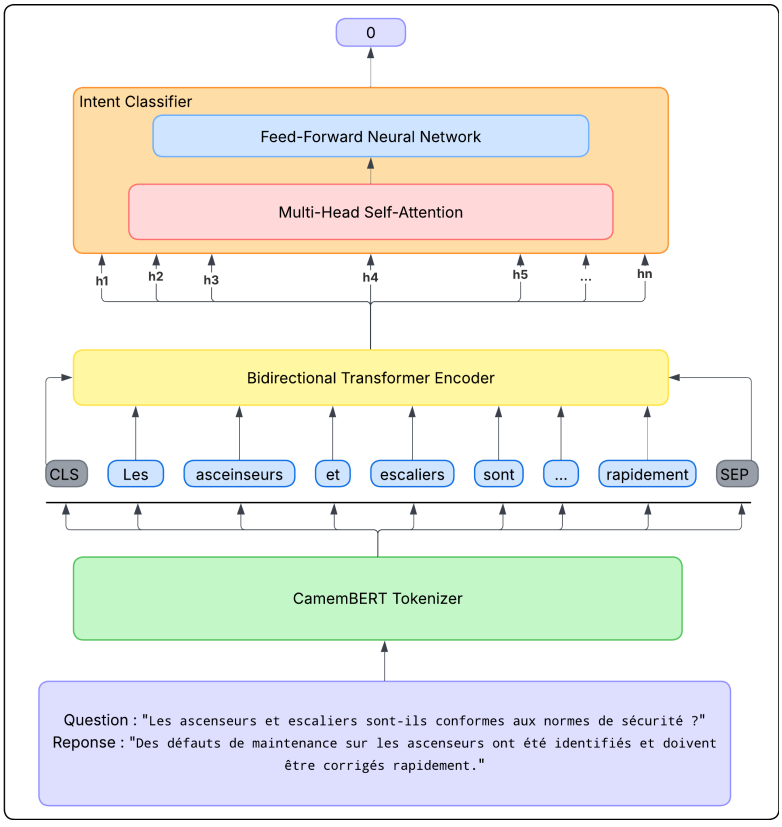


Figure 5.7: CamemBERT Architecture After Fine Tuning

Figure 5.7 shows CamemBERT Architecture after Fine Tuning.

5.3.2 VLM-Model Selection

For **PHOTO** Survey Entry type, we opted for using a MultiModal, as this emerges from the need of processing both `Textual` and `Visual` cues simultaneously.

Choosing the right model from the diverse selection that the Transformers Library offer is a crucial task, each model has an intended use-case, some of which are presented in Table 5.4.

Use Case	**Description**	**Models Examples**
Object Detection	Identifying and locating objects in images.	- Ultralytics/yolo - microsoft/table-transformer-detection
Image Classification	Assigning labels to images.	- Facebook/convnext-base-224-22k - Nvidia/mit-b0
Image Captioning	Generating text descriptions of images.	- Nlpconnect/vit-gpt2-image-captioning - Salesforce/blip2-opt-2.7b
Visual Question Answering	Answering questions based on visual input.	- Qwen/Qwen2.5-VL-7B-Instruct - Microsoft/llava-v1.5-7b - Google/gemma-3-4B

Table 5.4: MultiModal Use Cases and Example Models

The nature of the Visual Agent's Tasks, which are validating task execution and compliance, falls under the `Visual Question Answering` Use-case, through which the model must be able to analyze the given task and image simultaneously in order verify whether the task has been well executed or not.

In order to choose the right model for the Visual Agent, we benchmarked multiple models included in which:

- Gemma3-4B
- QWEN2-VL-7B
- Llava 1.5-7B

Benchmark Dataset Overview

The benchmark dataset consists of a subset of real world surveys results with following structure:

- **Columns:** task, image, label.
- **Label:** Binary value where `1` indicates well executed task, and `0` otherwise.
- **Size:** 144 entries.

- **Split:** 100% Inference.

To properly guide the benchmark and set equal foot for the models, we used same the same `generateCompliance` Function, shown in Figure 5.8, across all models, the function takes in a data point sample as **INPUT**, creates a `Prompt` to inference the model to return **ONLY** 1 if the image descries a well executed task, 0 otherwise as **OUTPUT**.

generateCompliance.py

```python
def generateCompliance(sample):
    prompt = f"""
        Task : {sample.get("task")},
        Image : {sample.get("answer")}
        Return  1 if the image shows that task has been well-executed, 0 otherwise.
        Do not provide explanation or reasoning,
        You only return 1 or 0, any other return will be invalid,
        """

    messages = [
        {
            "role": "system",
            "content": [
                {
                    "type": "text",
                    "text": "You are an Audit Validator who validates task execution.",
                }
            ],
        },
        {
            "role": "user",
            "content": [
                {"type": "image", "image": sample["answer"]},
                {"type": "text", "text": prompt},
            ],
        },
    ]
    inputs = processor.apply_chat_template(
        messages,
        add_generation_prompt=True,
        tokenize=True,
        return_dict=True,
        return_tensors="pt",
    ).to(model.device, torch.float16)

    # Generate
    generate_ids = model.generate(**inputs, max_new_tokens=10)
    message = processor.batch_decode(generate_ids, skip_special_tokens=True)
    model_output = message[0].split("ASSISTANT:")[1].strip()
    del inputs
    # Post-process to ensure clean output: "1" or "0"
    if "1" in model_output and "0" not in model_output:
        return 1
    elif "0" in model_output:
        return 0
    else:
        raise ValueError(f"Unexpected model response: {model_output}")
```

Figure 5.8: Generate Compliance Function

Benchmark Results

Once the Benchmarking is done, we can calculate the models' performances and compare them. Table 5.5 shows the Benchmarking results.

Model	Accuracy	Recall	F-1 Score	Precision	Execution Time
Gemma3-4B	0.56	0.62	0.60	0.65	72s
QWEN2-VL-7B	0.54	0.68	0.64	0.54	103s
Llava1.5-7B	0.48	0.62	0.55	0.56	99s

Table 5.5: Benchmarking Results

Based on the metrics presented in Table 5.5, the **Gemma3-4B**[14] model stands out among the evaluated candidates, demonstrating impressive out-of-the-box performance with a **0.75 Recall** Score and an Execution Time of just **72 seconds**.

As an open-source model, **Gemma3-4B** has benefited significantly from community contributions, including the development of 4-bit and 8-bit quantized versions, making it more accessible for training on hardware with limited GPU resources. Furthermore, the model is fully supported by the **Transformers** library, which allows for efficient fine-tuning using QLoRA[5].

Taken together, these advantages make the **Gemma3-4B** model an excellent choice for fine-tuning and integration within the Visual Agent for Visual Compliance Detection tasks.

5.3.3 Gemma3-4B for Visual Non-Conformity Detection

As mentioned previously in subsection 5.3.2, Gemma3-4B stood out as the most performing model, achieving [metricName,metricValue] without Fine-Tuning.

To guarantee a better performance and consistent results, it is essential to adapt this model to our use-case, here comes the need for Fine-Tuning the model on custom data, tailored specifically for retail surveys anomaly detection.

Training Environment

Though it is the smallest of its family, while being a multimodal, **Gemma3-4B** still requires heavy computational power, as many attempts of running the Fine-Tuning script on **Kaggle's Notebooks** only lead to `Out-of-Memory Errors`.

To overcome this obstacle, we decided to use **Amazon SageMaker AI**[2] as the Training Environment for this model for their user-friendly prices. But such prices came at a cost, running the fine-tuning scripts on SageMaker AI was not nearly as seamless as running them on Kaggle.

SageMaker introduced a steeper learning curve than we initially anticipated. The platform follows a production-oriented approach, requiring every training process to be executed as a **Training Job**. Unlike a simple notebook scripts-execution style, this concept has lead us to structure our fine-tuning code into a containerized **Docker image**. This step alone proved to be one of the most time-consuming phases. Dependency versioning was critical and tricky, finding

compatible versions of PyTorch, Transformers and other libraries required repeated trial and error.

Once the Docker image was correctly built, we pushed it to the **Amazon Elastic Container Registry**[1], from where it could be pulled by SageMaker's compute instances. For our training and merging steps, we provisioned a `ml.g6.4xLarge` instance.

Compute	Value
vCPUs	16
Memory (GiB)	64
Memory per vCPU (GiB)	4
Physical Processor	AMD EPYC 7R13 Processor
Clock Speed (GHz)	2.6 GHz
CPU Architecture	x86_64
GPU	1
GPU Architecture	NVIDIA L4
Video Memory (GiB)	24
GPU Compute Capability (?)	8.9
FPGA	0

Networking	Value
Network Performance (Gibps)	up to 25
Enhanced Networking	true
IPv6	true

Figure 5.9: VM Characteristics

Figure 5.9 describes the VM's specifications used during the training.

Ultimately, despite the steep curve and after over 10 attempts, SageMaker's scalability and resource flexibility made it possible to fine-tune Gemma3-4B effectively. The final working pipeline reflected not only a technically sound setup but also the resilience gained through numerous failures.

Dataset Overview

The dataset used for this training job consists of annotated survey entries, aggregated from multiple retail store verticals, with the following structure:

- **Columns:** task, image, label
- **Label:** Binary value where `1` indicates compliance,`0` otherwise.
- **Size:** 2008 Data Point.
- **Training Split:** 100% Training

Evaluation Metrics

Given that the nature of the model's output is binary, meaning a binary classification of task-execution, the evaluation metrics for this model will be the same as those of **CamemBert's**, found in Section 5.3.1

Training Arguments

Argument	Value	Explanation
`num_train_epochs`	2	Number of complete passes over the training set.
`learning_rate`	2e-4	Learning rate used for the optimizer.
`train_batch_size`	1	Number of examples per batch during training.
`gradient_accumulation_step`	4	Steps to accumulate gradients before backpropagation to simulate larger batch sizes.
`gradient_checkpoint`	True	Enables checkpointing to save GPU memory.
`optimizer`	adamw torch fused	Optimizer used for training
`logging_steps`	5	Frequency at which logs are reported.
`metric_for_best_model`	Recall	Model selection criteria.

Table 5.6: Gemma3-4B Fine-Tuning Hyperparameters

Due to Technical and Time limitations, we chose to train the model for only 2 epochs before proceeding with evaluation.

Training Results

All training and evaluation results are covered in Section5.5.

5.4 Justification & Recommendation Generation

To enhance the interpretability of non-conformity detection, **Prompt Engineering** [7] techniques were applied to guide **Gemma3-1B** in generating both justifications for non-compliant entries and actionable recommendations for resolving them.

These prompts encouraged the model to not only identify the issue but also articulate the likely cause of non-compliance (e.g., missing promotional board, incorrect product placement) in natural language.

generateJustification.py

```python
def generateJustification(task: str, execution: Union[str, int, float], agentJob: str):
    if agentJob == "GENERATE_JUSTIFICATION":
        expected_job = "Vous êtes un Audit Validateur qui explique les non-conformités."
    else:
        raise ValueError(
            f"Invalid agentJob Argument, Valid Argument: 'GENERATE_JUSTIFICATION' "
        )

    is_valid_input_type = isinstance(execution, (int, float, str))

    if not is_valid_input_type:
        raise TypeError("Argument `execution` must be a [str,int,float]")

    if is_valid_input_type:
        prompt = (
            f"Question : {task},\n"
            f"Reponse : {execution}\n"
            "Cet pair de Question-Reponse a été classée comme non conforme.\n"
            "Générez une justification d'une seule phrase expliquant pourquoi.\n"
            "Vous ne devez retourner qu'une seule phrase de justification, tout autre
retour sera invalide.\n"
            "Langue : Francais Uniquement."
        )
        user_content = [{"type": "text", "text": prompt}]

    messages = [
        {"role": "system", "content": [{"type": "text", "text": agentJob}]},
        {"role": "user", "content": user_content},
    ]

    inputs = GemmaProcessor.apply_chat_template(
        messages,
        add_generation_prompt=True,
        tokenize=True,
        return_dict=True,
        return_tensors="pt",
    )

    inputs = {
        k: v.to(GemmaModel.device) for k, v in inputs.items()
    }  # send tensors to GPU

    with torch.inference_mode():
        outputs = GemmaModel.generate(**inputs, max_new_tokens=64)

    outputs = GemmaProcessor.batch_decode(outputs)[0]
    result = outputs.split("<start_of_turn>model\n")[-1].strip("\n<end_of_turn>")

    return result
```

Justification Prompting

generateRecommendation.py

```python
def generateRecommendation(
    task: str, execution: Union[str, int, float], agentJob: str, justification: str
):
    if agentJob == "GENERATE_RECOMMENDATION":
        expected_job = "Vous êtes un validateur d'audit qui recommande des correctifs pour
résoudre les non-conformités détectées."
    else:
        raise ValueError(
            f"Invalid agentJob Argument, Valid Argument: 'GENERATE_RECOMMENDATION' "
        )
    is_valid_input_type = isinstance(execution, (int, float, str))
    if not is_valid_input_type:
        raise TypeError("Argument `execution` must be a [str,int,float]")

    if is_valid_input_type:
        prompt = (
            f"Question : {task},\n"
            f"Reponse : {execution}\n"
            f"Justification : {justification}\n"
            "Cet pair de Question-Reponse a été classée comme non conforme. "
            "Tenant compte de la justification comme contexte additionnel, "
            "générez une suggestion d'une seule phrase proposant une action qu'on doit
prendre pour résoudre cette non conformité.\n"
            "Vous ne devez retourner qu'une seule phrase de suggestion, tout autre retour
sera invalide.\n"
            "Langue : Francais Uniquement."
        )
        user_content = [{"type": "text", "text": prompt}]

    messages = [
        {"role": "system", "content": [{"type": "text", "text": agentJob}]},
        {"role": "user", "content": user_content},
    ]

    inputs = GemmaProcessor.apply_chat_template(
        messages,
        add_generation_prompt=True,
        tokenize=True,
        return_dict=True,
        return_tensors="pt",
    )

    inputs = {k: v.to(GemmaModel.device) for k, v in inputs.items()}

    with torch.inference_mode():
        outputs = GemmaModel.generate(**inputs, max_new_tokens=64)

    outputs = GemmaProcessor.batch_decode(outputs)[0]
    result = outputs.split("<start_of_turn>model\n")[-1].strip("\n<end_of_turn>")

    return result
```

Recommendation Prompting

Figure 5.10: Gemma3-1B Prompting

Figure 5.10 shows some Prompt Engineering techniques used in the generation of Justification and Actionable Recommendations in case of Non-Conformity Detection, some of which include:

- **Role Assignment:** Assigning the AI the role of an "Audit Validator" to ensure it adopts a critical and evaluative stance.
- **Instruction Prompting:** Providing explicit instructions to generate a one-sentence justification or recommendation, enforcing output constraints.
- **Contextual Prompting:** Supplying task and response pairs to enable the model to reason about the non-conformity.
- **Language Constraint:** Specifying the required language (e.g., French only) to maintain consistency across outputs.
- **System Message Framing:** Using system-level instructions to define the model's behavior and responsibilities from the outset.

5.5 Evaluation Results

Given the binary classification nature of the agent's task, appropriate evaluation metrics are essential to ensure a robust performance assessment. As mentioned in Subsection 5.3.1, we selected four key metrics: **Accuracy**, **F1 Score**, **Precision**, and **Recall**.

5.5.1 Evaluation Metrics

The metrics are defined using the following formulas, where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives
- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Since our primary objective is to minimize false negative, as failing to identify a positive non-conformity could lead to significant user dissatisfaction. Therefore **Recall** is used as the primary metric for model selection during evaluation and fine-tuning.

5.5.2 CamemBERT Evaluation Results

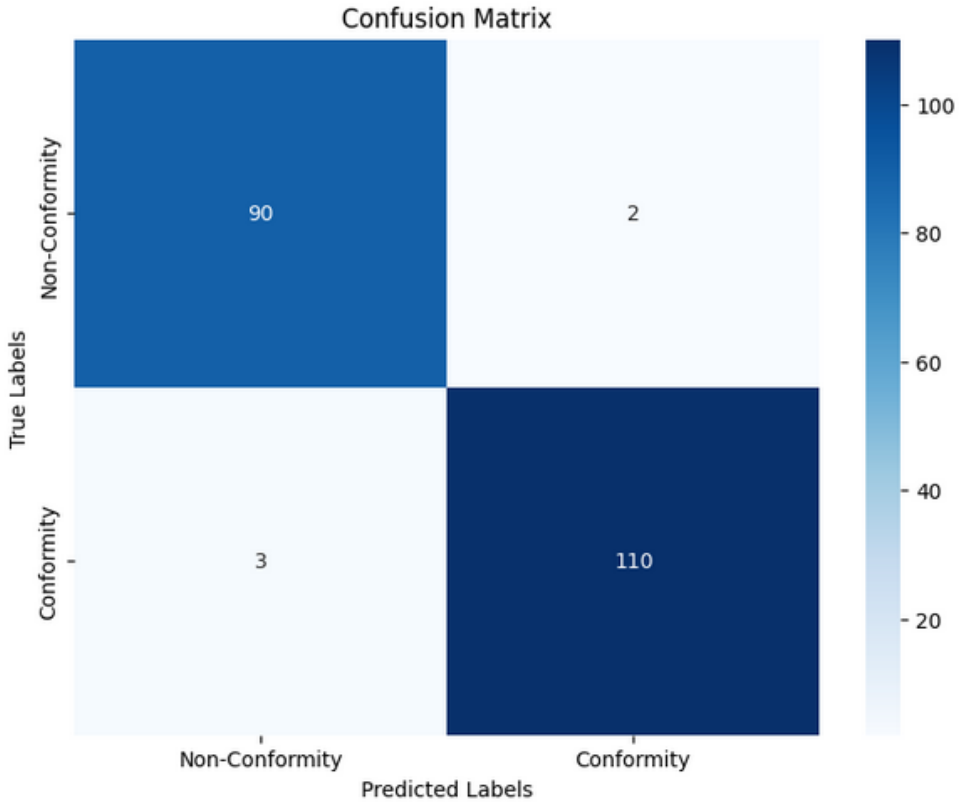


Figure 5.11: CamemBERT Evaluation Confusion Matrix

Figure 5.11 shows the confusion matrix obtained on the evaluation dataset, from which we calculated the evaluation metrics in Table 5.7:

Metric	Value
Accuracy	0.97
Precision	0.98
Recall	0.97
F1 Score	0.97

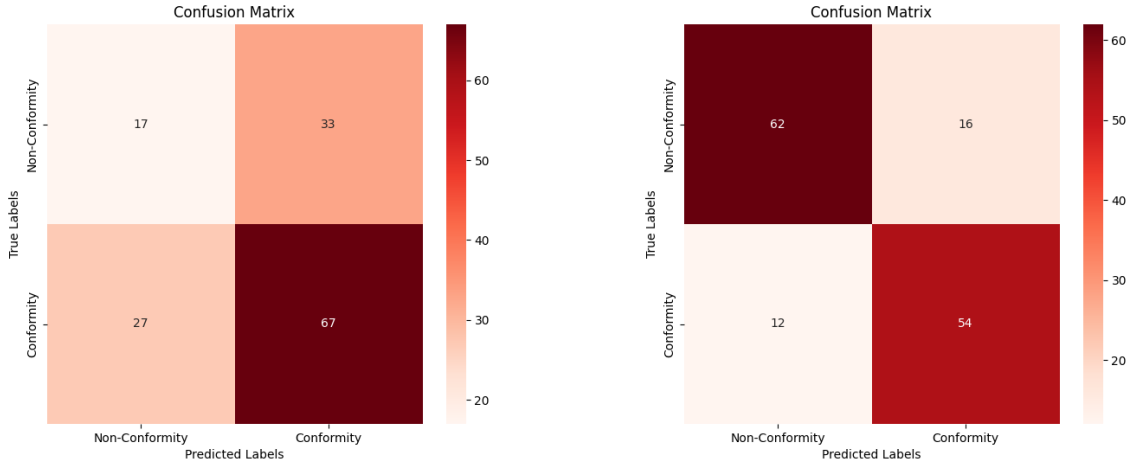
Table 5.7: Evaluation metrics after fine-tuning

These results demonstrate that the Fine-Tuned CamemBERT model achieves high performance, particularly in minimizing false negatives. This aligns with our objective of prioritizing the correct identification of non-conformities to ensure optimal user satisfaction.

5.5.3 Gemma3-4B Evaluation Results

To evaluate **Gemma3-4B's** performance after fine-tuning on our custom dataset, we used the same benchmark dataset that was employed prior to fine-tuning. Importantly, this benchmark

data was not included in the training set during fine-tuning, ensuring an unbiased evaluation of the model's improvement.



Confusing Matrix before Fine-Tuning **Confusing Matrix after Fine-Tuning**

Figure 5.12: Gemma3-4B performance before VS after Fine-Tuning

Figure 5.12 shows the confusion matrix obtained on the evaluation dataset, from which we can calculate the evaluation metrics :

Metric	Before Fine-Tuning	After Fine-Tuning
Accuracy	0.56	0.67
Precision	0.65	0.77
Recall	0.62	0.82
F1 Score	0.60	0.79

Table 5.8: Gemma3-4B Evaluation metrics before VS after Fine-Tuning

Quantitative Performance Comparison

Based on the results shown in Table 5.8, the performance of **Gemma3-4B** improved significantly. We observe that:

- **Accuracy** increased from 0.65 to 0.67, indicating better overall correctness of compliance detection.
- **Precision** improved from 0.65 to 0.77, showing fewer false positives.
- **Recall** improved from 0.62 to 0.82, which is important for detecting Positive cases (Non-Conformity) and lowering the False Negatives.

- **F1-score** rose from 0.60 to 0.79, suggesting a better balance between precision and recall.

Noting that these results were achieved after fine-tuning the model for only 2 epochs. In future iterations, with a larger number of epochs and more extensive parameter tuning, we expect the model's performance to improve further.

5.6 Conclusion

In this chapter, we covered the architecture and implementation of the proposed MAS through fine-tuning various models on our custom datasets. The comprehensive evaluation demonstrated promising performance and served as a proof of concept for the effectiveness of our approach.

Chapter 6

Deployment and Integration

6.1 Introduction

In this chapter we cover the deployment environment of the MAS as well showcase its integration within Winshot's platform.

6.2 Deployment Strategy

In this section, we present the available deployment options, evaluates their respective advantages and disadvantages, and explains the rationale behind the selected strategy.

6.2.1 Deployment Environments

Environment	Pros	Cons
On-Premise Servers	• Full control over infrastructure. • High data security. • No dependency on internet availability.	• High upfront costs. • Requires in-house maintenance • Limited scalability
Cloud (AWS, Azure, GCP, etc ...)	• High scalability and flexibility. • Pay-as-you-go pricing. • Managed infrastructure and services.	• Ongoing operational costs. • Vendor lock-in risks. • Potential data privacy concerns.
Serverless (AWS Lambda, Azure Functions, etc ...)	• No server management required. • Automatic scaling. • Cost-effective for low traffic workloads.	• Limited execution time. • Cold start latency. • Tightly coupled to cloud vendor.

Table 6.1: Comparison of Common Deployment Environments

Table 6.1 shows some of the available deployment environments.

6.2.2 Infrastructure Architecture

Given the high demand for GPU capabilities and computational power, especially for inferencing our VLM model, Gemma3-4B, our preferred deployment option for hosting the MAS was Amazon Web Services. As AWS offer a broad range of GPU-accelerated virtual machines for a very cost-effective price, making it well-suited to meet our performance requirements.

Our decision to use AWS was not solely based on their competitive pricing and extensive instance options. We also considered our existing use of AWS for various services such as **EC2** for servers and **S3** for storage. Staying within the same cloud ecosystem allows for seamless integration and reduced operational complexity.

6.2.3 MAS Serving

To support the high-performance requirements of the MAS, we deployed it on a dedicated GPU-based Virtual Machine, **ml.g6.4xLarge**, hosted on AWS. The Machine comes pre-configured with the following specifications:

Component	**Specification**
CPU	16 vCPUs (AMD EPYC 7R13 Processor)
GPU	1 x NVIDIA L4
VRAM	24 GB
Memory	64 GB RAM
Storage	600GB SSD
Operating System	Ubuntu 22.04 LTS
Network Performance	Up to 25 Gibps
Max Bandwidth on EBS	8000 Mbps

Table 6.2: VM Specifications for MAS Hosting

This VM is designated as the `MCP Server`, responsible for hosting and serving the MAS components, including our core Supervisor Model.

6.3 Integration Plan

Deploying the MAS is only part of the challenge, without proper integration into the existing operational workflow, we can't fully harness its potential or assess strengths, or weaknesses. This section details the plan for integrating the MAS into **Winshot's platform**, focusing on its interaction with the `Winshot AI Agent` and the data flow.

6.3.1 MAS Integration with Winshot AI Agent

As discussed earlier in subsection 5.2.2, the communication between the client and server is facilitated through the MCP protocol. The MAS server receives incoming MCP requests from a Winshot AI Agent, which acts as the MCP client. Each request is encapsulated in an MCP message containing attributes such as:

- **Sender** – The identifier of the AI agent initiating the request (Winshot AI in our case).
- **Recipient** – The designated server endpoint.
- **Target Agent** – The target Agent to invoke.
- **Payload** – A raw checklist requiring validation and processing.

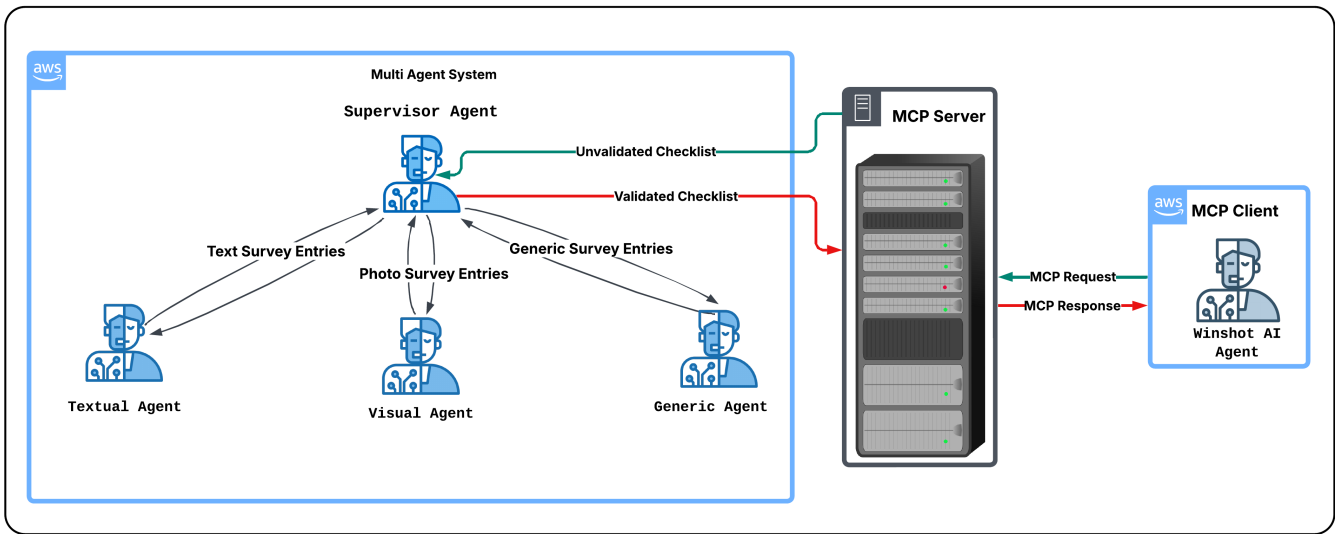


Figure 6.1: Model Serving Architecture

As described in Figure 6.1, upon receiving the message, the **MCP Server** forwards the payload to the Recipient Agent, Supervisor Agent in our case. This last, as illustrated in Figure 5.2 , interprets the checklist, invokes the MAS workflow to perform validation and reasoning tasks, and reconstructs a fully validated checklist.

Once the validation process is completed, the server wraps the validated survey into a structured MCP response and sends it back to the Winshot AI Agent, thereby completing the request-response cycle.

6.3.2 End-to-End MAS Integration within Winshot's Platform

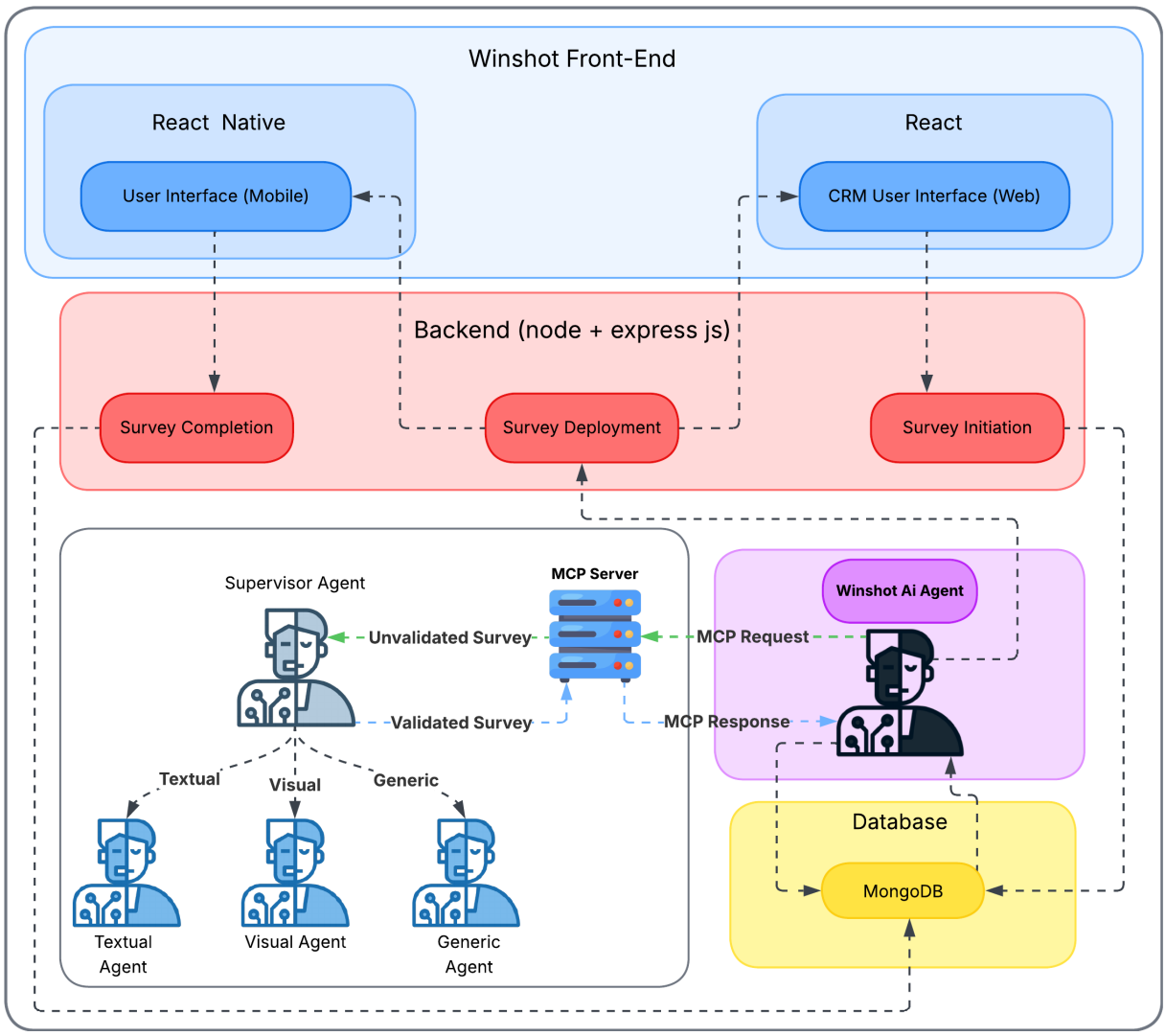


Figure 6.2: MAS Integration Architecture

Figure 6.2 describes the End-to-End MAS integration into **Winshot's Platform**, the workflow is as follows:

1. Auditors create a Survey from Winshot's Web Front-End.
2. Survey is saved into the Database.
3. Survey is completed by Store Staffs through Winshot's Mobile Front-End.
4. Survey answers saved into the Database.
5. Winshot AI server pulls the Survey from the Database.
6. Winshot AI server Wraps the Survey in **MCP Request**, in which it specifies:

- Its identity as `Sender`.
- The MCP server address as `Recipient`.
- The Supervisor Agent as Target Agent.
- The Survey as `Payload`.

7. Winshot AI Agent initiates the Request-Response cycle by sending an **MCP Request** to the MCP Server, previously discussed in subsection 6.3.1.

8. Winshot AI Agent saves the validated report into the Database, and updates it into Winshot's Front-Ends.

6.3.3 User Interface and Reporting Integration

This subsection presents examples of the adapted user interfaces from the original Winshot platform, modified as part of our MAS integration.

6.4 Limitations

Despite the advantages offered by the proposed MAS, several technical limitations were encountered during development and deployment, some of which are:

- Inference Latency: The orchestration of agents and sequential model inferences can introduce notable delays, affecting real-time performance.
- High Computational Demand: The system's architecture involves multiple specialized agents operating in parallel. This design, while modular and intelligent, significantly increases computational load, especially when processing large datasets for either training or inferencing.
- Inconsistent Model Output: The models sometimes face challenges with subjective questions, leading to variability and occasional inconsistency in their responses.

6.5 Perspectives

To mitigate the limitations mentioned in section 6.4, future versions of the MAS can include:

- Performance improvements for the Fine-Tuned models through the exploration of Domain Adaptation[17] and Continual Learning[16].

- Infrastructure Scaling: Improve response times by deploying models on more powerful VMs with better GPUs and leveraging parallel processing across multiple agents to reduce bottlenecks during inference.
- User Feedback Loop: Integrate user feedback mechanisms to continuously collect data on model performance, enabling dynamic updates and iterative improvements.

6.6 Conclusion

By leveraging AWS infrastructure as a deployment infrastructure, and through the integration of the Multi Agent System into Winshot's platform, we were able to provide a working proof-of-concept for our MAS that is both scalable and high performing.

General Conclusion

This report serves as a proof of concept for the development of a Multi Agent System aimed at automating retail store compliance assessment. Our work strategy was inspired from the **CRISP-DM** methodology. We began by understanding the business and its requirements, analyzing the current status of the collected data for this project, and identifying weakness points in the traditional compliance evaluation workflows. Then we fine-tuned models for text and image analysis and combined them with specialized agents allowing for automated detection of non-conformities in survey responses across multiple data formats. Prompt engineering techniques were also applied to enhance explainability, enabling the generation of human-readable justifications and actionable recommendations, useful for resolving the detected non-conformities. The preliminary results demonstrate the potential of agentic AI in real-world retail environments, not only in terms of performance but also in modularity and scalability.

Bibliography

[1] Amazon Web Services. Amazon elastic container registry, 2025.

[2] Amazon Web Services. Amazon sagemaker, 2025.

[3] Bryan Clark. What is quantization?, 2024.

[4] Google Cloud. Multimodal ai, 2024.

[5] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[6] Google. Introducing gemma: Open models built from the same research and technology used to create gemini, 2024.

[7] Google Cloud. Prompt engineering for ai guide, 2025.

[8] IBM. Vision-language models: Combining the power of sight and language, 2023.

[9] IBM. What is a multi-agent system?, 2024.

[10] IBM. What is fine-tuning in machine learning?, 2024.

[11] IBM. Low-rank adaptation (lora) fine tuning, 2025.

[12] LLaVA Team. Llava: Large language and vision assistant, 2024.

[13] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[14] Gemma Team. Gemma 3. 2025.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[16] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

[17] Yufeng Zhang, Jianxing Yu, Yanghui Rao, Libin Zheng, Qinliang Su, Huaijie Zhu, and Jian Yin. Domain adaptation for subjective induction questions answering on products by adversarial disentangled learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9074–9089, Bangkok, Thailand, August 2024. Association for Computational Linguistics.