

# Agentic workflow for Phishing detection

Lucas Campbell-Robson

22073040

UXCFPU-30-3

Project and Dissertation

Word Count: 10,925



## Abstract

This dissertation explores the use of agentic workflows in cybersecurity, with a particular focus on phishing detection. Traditional detection methods often rely on large, static datasets and computationally expensive models, making them costly and inaccessible for many organisations. Additionally, most existing solutions operate as black boxes, offering little transparency or user involvement, missing an opportunity to educate and assist users in identifying threats.

To address these limitations, this project introduces a novel, low-cost phishing detection system built around recent advancements in agent-based architectures. The system combines email classification using models such as fine-tuned BERT and LLaMA with decision-making processes driven by cutting-edge large language models like Gemini, ChatGPT, and Mistral. These agents can access external resources, analysing contextual information from user email history, and presenting a clear assessment of phishing likelihood to the user, promoting both real-time protection and user training.

The system achieved 98% accuracy on the classification validation dataset and demonstrated strong performance in practical scenarios, with seamless integration into platforms like Outlook. The findings highlight the potential of agentic workflows to transform cybersecurity workflows through lightweight, scalable solutions, especially as edge-capable LLMs continue to advance. However, the project also identifies significant challenges, particularly around prompt injection attacks and agent reliability, emphasising the need for robust control mechanisms to ensure consistent task alignment and system safety.

## Acknowledgements

I am deeply grateful to my family: my dad, mum, and sister, for their constant support, patience, and belief in me, not only during this dissertation but throughout my entire academic journey. To my friends within the course, thank you for the shared laughs, endless chats, and collective moments of stress that somehow made the whole experience more bearable. I am also incredibly thankful for the encouragement and good energy from my friends in badminton and those back in Ipswich. Your support has meant a great deal. This dissertation reflects not just my individual effort but also the encouragement and strength drawn from the people around me. For that, I am truly thankful.

# Table of Contents

|   |    |
|---|----|
| Abstract.....   | 2  |
| Acknowledgements .....  | 2  |
| Table of Contents.....  | 3  |
| Table of Figures.....   | 5  |
| Table of Tables .....   | 7  |
| Chapter 1 - Introduction .....                                    | 8  |
| 1.1.    Problem Area.....   | 8  |
| 1.2.    Specific Problem .....                                    | 8  |
| 1.3.    Project Overview .....                                    | 8  |
| 1.4.    Aims and Objectives .....                                 | 9  |
| 1.4.1.    Aims .....  | 9  |
| 1.4.2.    Objectives .....  | 9  |
| Chapter 2 - Literature Review.....                                | 10 |
| 2.1. Introduction .....   | 10 |
| 2.2. Phishing detection issues.....                               | 10 |
| 2.3. Current methods.....   | 11 |
| 2.4. Data collection .....  | 12 |
| 2.5. Fine tuning efficiency.....                                  | 12 |
| 2.6. Involving tools.....   | 13 |
| Chapter 3 - Methodology.....                                      | 14 |
| 3.1. Introduction .....   | 14 |
| 3.2. Development methodology .....                                | 14 |
| 3.3. Requirements.....  | 15 |
| 3.3.1. Literature review.....                                     | 15 |
| 3.3.2. Stakeholder Engagement .....                               | 15 |
| 3.4. Design .....   | 15 |
| 3.5. Implementation .....   | 15 |
| 3.6. Testing.....   | 15 |
| 3.7. Gantt Chart .....  | 15 |
| Chapter 4 - Requirements.....                                     | 16 |
| 4.1. Introduction .....   | 16 |
| 4.2. Stakeholder Identification .....                             | 16 |
| 4.3. Functional Requirements and Non-Functional Requirements..... | 17 |

|  |    |
|--|----|
| 4.3.1. Use case diagram .....                  | 17 |
| 4.3.2. Functional Requirements.....            | 17 |
| 4.3.3. Non-Functional Requirements.....        | 18 |
| Chapter 5 - Design .....                       | 19 |
| 5.1. Introduction .....                        | 19 |
| 5.2. High-level design.....                    | 20 |
| 5.2.1. User Process Flow .....                 | 20 |
| 5.2.2. Multi-Agent System process .....        | 20 |
| 5.2.3. GUI .....                               | 21 |
| 5.3. Low-Level design.....                     | 22 |
| 5.3.1. Process Flow .....                      | 22 |
| 5.3.2. Sequence Diagram.....                   | 24 |
| 5.3.3. Tool composition .....                  | 25 |
| 5.3.4. Databases.....                          | 28 |
| 5.3.5. Model Finetuning.....                   | 28 |
| 5.3.6. Testing.....                            | 29 |
| Chapter 6 – Implementation.....                | 30 |
| 6.1. Introduction .....                        | 30 |
| 6.2. Initial agent .....                       | 30 |
| 6.2. Implementing email analysis .....         | 31 |
| 6.3. Outlook analysis bot .....                | 33 |
| 6.4. Extractor agent .....                     | 34 |
| 6.5. Supervisor structure .....                | 35 |
| 6.6. Full system .....                         | 38 |
| 6.7. Data collection .....                     | 39 |
| 6.8. Data cleaning .....                       | 39 |
| 6.9. Ensemble.....                             | 42 |
| 6.9.1. LLaMa 3.2 3B parameter finetuning ..... | 42 |
| 6.9.2. BERT Finetuning .....                   | 44 |
| 6.9.3. Kappa statistics .....                  | 44 |
| Chapter 7 - Testing.....                       | 45 |
| 7.1. Considerations .....                      | 45 |
| 7.2. Integration testing .....                 | 46 |
| 7.3. System testing.....                       | 49 |
| 7.3.1. Test results #1 and #2 .....            | 52 |
| 7.3.2. Test results #3 .....                   | 53 |

|   |    |
|---|----|
| 7.3.3. Test results #4 and #5 .....                                   | 54 |
| 7.3.4. Test results #6 .....  | 56 |
| 7.3.5. Test results #7 .....  | 59 |
| Chapter 8 - Project Evaluation .....                                  | 61 |
| Chapter 9 - Conclusion .....  | 62 |
| Chapter 10 – Future considerations .....                              | 62 |
| Bibliography .....  | 64 |
| Appendices .....  | 67 |
| Appendix A: Further literature review .....                           | 67 |
| A.1 .....   | 67 |
| A.2 .....   | 67 |
| A.3 .....   | 67 |
| A.4 .....   | 68 |
| Appendix B: Data processing design .....                              | 69 |
| Appendix C: Data cleaning tools .....                                 | 70 |
| Appendix D: Integration tests .....                                   | 71 |
| Appendix E: Email chain for Business email compromise and fraud ..... | 75 |
| Appendix F: Email chain for Legitimate communication .....            | 76 |
| Appendix G: Stakeholder Questionnaire .....                           | 77 |

## Table of Figures

|  |    |
|--|----|
| Figure 1 - Gantt Chart .....                     | 16 |
| Figure 2 - Use case diagram .....                | 17 |
| Figure 3- User Process Flow .....                | 20 |
| Figure 4 - Agent system Components Diagram ..... | 21 |
| Figure 5- GUI Design .....                       | 22 |
| Figure 6 - State flow diagram.....               | 23 |
| Figure 7 - Email analysis Sequence Diagram ..... | 24 |
| Figure 8 - URL scan Sequence Diagram.....        | 25 |
| Figure 9 - Save memory tool.....                 | 26 |
| Figure 10 - Search memory tool.....              | 26 |
| Figure 11 - Ensemble analysis tool.....          | 27 |
| Figure 12 - URL analysis tool .....              | 27 |
| Figure 13 - MongoDB JSON structure .....         | 28 |
| Figure 14 - Initial agent graph.....             | 31 |
| Figure 15 - Initial agent's memory usage .....   | 31 |
| Figure 16 - CAB graph .....                      | 32 |
| Figure 17 - CAB ensemble response .....          | 32 |
| Figure 18 - CAB prompt.....                      | 33 |

|  |    |
|--|----|
| Figure 19 - Outlook analysis bot graph .....                       | 33 |
| Figure 20 - Replanner prompt.....                                  | 34 |
| Figure 21 - Extractor agent .....                                  | 34 |
| Figure 22 - Initial supervisor architecture.....                   | 35 |
| Figure 23 – Initial Supervisor prompt .....                        | 35 |
| Figure 24 - Initial supervisor, coder and researcher response..... | 36 |
| Figure 25 - Memory tools not being utilised.....                   | 36 |
| Figure 26 - Supervisor adjustment.....                             | 37 |
| Figure 27 - Conversation response .....                            | 37 |
| Figure 28 - Supervisor prompt.....                                 | 37 |
| Figure 29 - Full system graph.....                                 | 38 |
| Figure 30 - Adjusted Supervisor prompt.....                        | 38 |
| Figure 31 - Cornell University Phish Bowl data collection .....    | 39 |
| Figure 32 - Phishing Pot data collection.....                      | 39 |
| Figure 33 - Spear Phishing dataset.....                            | 39 |
| Figure 34 - CEAS dataset .....                                     | 39 |
| Figure 35 - Phishing dataset distribution before filtering.....    | 40 |
| Figure 36 - Phishing dataset boxplot before filtering .....        | 40 |
| Figure 37- Phishing dataset after filtering .....                  | 41 |
| Figure 38 - Legitimate dataset .....                               | 41 |
| Figure 39 - Merged dataset distribution .....                      | 41 |
| Figure 40 - Quantization and LoRA configuration.....               | 42 |
| Figure 41 - LLM finetuning loop .....                              | 43 |
| Figure 42- Finetuned Llama with LoRa weights .....                 | 43 |
| Figure 43 - Validation dataset example results.....                | 44 |
| Figure 44 - BERT Learning rate.....                                | 44 |
| Figure 45 - Kappa statistics calculation.....                      | 45 |
| Figure 46 - Memory test .....                                      | 46 |
| Figure 47 - Memory test response.....                              | 47 |
| Figure 48 - Malicious prompt.....                                  | 48 |
| Figure 49 - Prompt configuration for restrictions.....             | 48 |
| Figure 50 - Malicious prompt fixed .....                           | 49 |
| Figure 51 - Second malicious prompt .....                          | 49 |
| Figure 52 - Obvious phishing email.....                            | 52 |
| Figure 53 - Test 1 response .....                                  | 52 |
| Figure 54 - Test 2 response .....                                  | 53 |
| Figure 55 - Organisational phishing email .....                    | 53 |
| Figure 56 - Test 3 response .....                                  | 54 |
| Figure 57 - MongoDB crafted emails for context.....                | 54 |
| Figure 58 - BEC response .....                                     | 55 |
| Figure 59 - Fraud email response.....                              | 56 |
| Figure 60 – Outlook analysis bot extract .....                     | 56 |
| Figure 61 - Marketing email context.....                           | 56 |
| Figure 62 - Marketing email.....                                   | 57 |
| Figure 63 - Initial response to marketing email .....              | 58 |
| Figure 64 - Outlook analysis on marketing email.....               | 58 |
| Figure 65 - Further context to marketing email.....                | 58 |
| Figure 66 - Response to marketing email with further context.....  | 59 |

|  |    |
|--|----|
| Figure 67 - chatSupervisor memory usage .....      | 59 |
| Figure 68 - Pinecone database extract.....         | 59 |
| Figure 69 - Legitimate correspondence context..... | 60 |
| Figure 70 - Test 7 response .....                  | 60 |

## Table of Tables

|   |    |
|---|----|
| Table 1 - Functional Requirements Table .....     | 18 |
| Table 2 - Non-Functional Requirements Table ..... | 19 |
| Table 3 - Test case table design .....            | 29 |
| Table 4 - Validation dataset results .....        | 45 |
| Table 5 - Integration test cases.....             | 46 |
| Table 6 - System test cases .....                 | 51 |

# Chapter 1 - Introduction

## 1.1. Problem Area

Phishing attacks remain a pervasive and evolving cybersecurity threat affecting individuals and organisations of all sizes. Large enterprises often have advanced detection systems and ongoing training, but smaller organisations and individuals typically lack the resources for effective defence. Many existing solutions are costly, complex, or rely on static rule-based methods that can't adapt to evolving tactics. Traditional training often emphasises outdated indicators like misspellings and generic greetings, leaving users vulnerable to sophisticated, socially engineered attacks.

## 1.2. Specific Problem

The core challenge addressed by this project is the gap between evolving phishing threats and the accessibility of adaptive detection and training solutions for users without dedicated cybersecurity expertise. Traditional phishing detection tools focus on identifying generic red flags, without providing users meaningful insights into why an email may or may not be a phishing attempt. This lack of contextual analysis and feedback prevents users from developing a deeper understanding of phishing tactics, reducing the effectiveness of training over time.

Existing solutions also rarely consider a user's specific email patterns and communication history, leading to false positives or missed threats. This project introduces an intelligent phishing detection agent that not only identifies potential phishing attempts but also explains its reasoning in an accessible manner. By leveraging multiple analysis tools and considering the user's past emails for context, the system enhances phishing awareness while providing a dynamic, evolving learning experience. This approach bridges the gap between automated security and user education, empowering individuals and smaller organisations to defend themselves more effectively against phishing threats.

## 1.3. Project Overview

This project explores the use of Large Language Models (LLMs) for phishing detection while prioritising efficiency over large-parameter models. The developed system is an agent-based solution that enhances LLM performance through optimised prompting, achieving detection accuracy comparable to larger models without excessive computational demands.

The system integrates specialised agents that collaborate to analyse emails using ensemble methods. A fine-tuned model, trained on curated phishing datasets, strengthens detection, while external Application Programming Interfaces (API) provide real-time URL scanning and contextual analysis. The system also incorporates past Outlook email data to improve accuracy by considering a user's unique email patterns.

Beyond detection, the system functions as an educational tool, offering detailed phishing breakdowns, confidence scores, and explanations for each classification. With long-term memory, it adapts to evolving phishing tactics, providing users, particularly individuals and small organisations without advanced security tools, with an intelligent, self-improving defence mechanism. This approach enhances phishing detection and fosters cybersecurity awareness, bridging the gap between automated security and user education.

## 1.4. Aims and Objectives

### 1.4.1. Aims

- Develop an efficient phishing detection system leveraging LLMs while minimising dependence on large-parameter models.
- Enhance phishing detection accuracy through optimized prompting techniques and an agent-based approach.
- Bridge phishing detection and user education by providing clear, contextual insights into detected threats.
- Improve accessibility to advanced phishing defence mechanisms for individuals and smaller organisations that lack dedicated security resources.

### 1.4.2. Objectives

- Design and implement an agent-based system that utilises LLMs for phishing detection and user assistance.
- Develop an ensemble analysis approach by combining multiple models, including a fine-tuned phishing detection model trained on curated data.
- Integrate external APIs for real-time threat intelligence, including URL scanning and phishing database lookups.
- Enhance detection accuracy through contextual analysis of past emails, enabling the system to recognise evolving phishing tactics.
- Provide users with phishing breakdowns, confidence scores, and detailed explanations to improve cybersecurity awareness.
- Implement long-term memory capabilities to track phishing patterns and adapt to emerging threats over time.

# Chapter 2 - Literature Review

## 2.1. Introduction

This literature review explores phishing detection methods, from traditional machine learning techniques to the emerging use of LLMs. It also examines the role of agent systems in supporting and enhancing these detection approaches.

## 2.2. Phishing detection issues

While commercially available detection systems exist, they are not widely adopted due to high acquisition and maintenance costs. In the UK, the Cyber Security Breaches Survey (2024) states that only 54% of businesses have an agreed process for handling phishing emails. This is often due to a lack of understanding, resources, and maintainability. Phishing training programs can be difficult to set up and sustain, especially for smaller organisations. The absence of robust detection and training processes increases the risk of successful phishing attacks. Sumner et al. (2021) highlight the need for consistent phishing training, noting that untrained employees performed significantly worse, and that training often fails to improve perceived risk when receiving phishing emails.

Training users to recognise phishing emails has been shown to increase their confidence. Sumner et al. (2021) found that individuals who correctly identify phishing attacks are more confident in their choices. This suggests that boosting user confidence leads to better outcomes, which is relevant to this proposal, as involving users in classification can be beneficial. The paper also notes that users without prior training showed weak negative correlations with both pre-training accuracy and confidence, supporting the idea that training is a continuous process that improves accuracy.

The findings from these papers highlight three key points: training must be consistently tested, reminders of phishing risks should be clear, and training alone should not be relied upon as the sole solution.

The Zscaler Phishing Threat Report by Desai and Hedge (2024) identified the top targeted industries: Finance and Insurance, Manufacturing, Services, Technology, and Retail. These sectors have seen a rise in attacks due to their heavy reliance on digital platforms and the growing use of AI, which handles valuable data. The paper reinforces that phishing is not limited to email, social media is a key attack vector due to its wide reach and ability to engage users on a more personal level. Commercial solutions struggle to address this, as they are primarily designed for email detection. Direct messaging phishing is also harder to detect, as messages are more conversational and lack information typically found in emails, such as sender domains.

The report also highlights the rise of generative AI used for phishing, showing that detection based on spelling or grammar errors is no longer sufficient. This underscores the need to update training approaches for both users and machine learning models, moving beyond outdated datasets.

Heiding et al. (2023) found that while LLMs like GPT-4 can generate highly convincing phishing content by mimicking tone, format, and style, they are also powerful detection tools, using context-aware analysis to identify subtle signs of phishing. This dual capacity creates a notable cybersecurity challenge: LLMs can enhance automated defences but also lower the barrier for threat actors to craft persuasive attacks. This raises the question, how well can LLMs detect generated text? Koike,

Kaneko, and Okazaki (2023) proposed a solution, finding the OUTFOX detector achieved the best results on GPT-3.5-generated text, with a detection accuracy of 96.2% and an F1 score of 96.9%. This setup showed the highest accuracy and robustness, even in adversarial conditions. While these are strong results, they don't guarantee that LLM-generated text is fully detectable. However, this is not a major concern for the current solution, as its focus is on detecting phishing emails, not necessarily whether they were LLM-generated, though the two are increasingly linked.

### 2.3. Current methods

Text classification has seen BERT widely adopted, with models like MalBERT - Xu, Fang & Yang (2021), HateBERT - Caselli et al. (2020), and SpambERT - Joshi et al. (2020). These are fine-tuned on specialised datasets targeting specific tasks such as malware detection or abusive language. This reflects the rise of LLMs as the preferred approach for text classification due to their ability to understand context. In contrast, traditional ML methods rely on static embeddings, so words like "bank" share the same vector regardless of meaning—whether financial or geographical—leading to poor performance in nuanced tasks.

Fine-tuned LLMs have consistently outperformed traditional algorithms. Wang et al. (2024) showed that fine-tuned LLMs significantly outperformed Multinomial Naive Bayes, Logistic Regression, Random Forest, Decision Tree (DT), and K-Nearest Neighbors (KNN) on a COVID-19-related text dataset, closely aligning with the focus of this paper. Papageorgiou et al. (2024) supported this, reporting strong performance from FakeNewsGPT4, which achieved 7.7% higher AUC than baseline models in single-domain tasks and outperformed all competitors in multi-domain settings. These findings highlight the growing shift toward LLMs in real-world applications, as traditional methods struggle to adapt to evolving data.

Another paper by Wahba, Madhavji, and Steinbacher (2022) compared the performance of Support Vector Machines (SVM), once popular in text classification, with fine-tuned LLMs, finding that LLMs consistently outperformed SVMs. While SVMs showed competitive accuracy, their performance varied significantly based on feature engineering. The authors concluded that LLMs generally achieve higher accuracy on generic datasets, highlighting their advantages in these tasks.

A common technique for boosting accuracy in text classification is ensemble learning, which combines multiple models by majority voting or result aggregation to create a more robust solution.

Phishing classification has specifically benefited from ensemble methods. Bountakas and Xenakis (2023) showcased a framework called HELPHED, which used multiple algorithms to capture a broader understanding of the data. Their staged approach involved DT for content-based features and KNN for text-based features. Classification had two approaches, either a multi-layer perceptron or a soft voting classifier. The latter achieved an F1-score of 0.9942, and accuracy, precision, and recall of 0.9943, with a Matthews correlation coefficient (MCC) of 0.967. Despite these strong results showing the benefits of ensemble techniques, the emails used were from outdated datasets like Enron (2002), SpamAssassin (2004), and Nazario (2007), which were based on old phishing ideas such as misspellings and biased specific scam methods, meaning the model may not perform as well with latest phishing techniques. In contrast, LLMs are trained on large, diverse corpora, offering deeper text understanding than traditional methods reliant on surface patterns.

An implementation building on the HELPHED framework and modern LLM methods is proposed by Jiang, Ren, and Lin (2023). Their ensemble framework combines two components: PairRanker, which ranks outputs from multiple LLMs based on quality, and GenFuser, which merges the top-ranked outputs to produce a final response that integrates their strengths. This approach enhances

robustness by leveraging multiple LLMs, rather than relying on a single model. Unlike the fixed ensemble of HELPED, which uses all classifiers, the LLM blender is dynamic, optimising based on the top-ranked model. However, its reliance on multiple LLMs makes it resource-intensive and reliant on large datasets for finetuning.

## 2.4. Data collection

To address the lack of high-quality, publicly accessible datasets for fine-tuning phishing detection models, and the reliance on outdated datasets like SpamAssassin, Nazario, or CEAS, data generated by LLMs can be a solution. Afane et al. (2024) found that training models on datasets augmented with ChatGPT and LLaMA emails resulted in a 12.6% accuracy increase. The paper supports the idea that generated text is more effective for phishing detection than outdated datasets, like Nazario, which showed decreased accuracy and recall.

While generating emails may seem like a clear solution to data shortages, a study by Ilia Shumailov et al. (2024) raised concerns about using generated data for model training. These models can experience 'collapse,' where they gradually forget aspects of the true distribution they initially learned. As noted in the paper, models become less capable of recognising or generating less common or diverse data points, leading to homogenised outputs. This collapse was observed after 5 to 10 recursions, though the range may vary depending on data complexity. Notably, this issue only arises with large-scale use of generated data, while initial training cycles showed no performance decrease which shows it is a viable solution if used in combination with legitimate data.

## 2.5. Fine tuning efficiency

Fine-tuning models efficiently has become a priority in natural language processing, especially as model sizes grow exponentially. LoRA (Low-Rank Adaptation) introduced by Hu et al. (2021) is a parameter-efficient fine-tuning method that significantly reduces the number of trainable parameters by injecting low-rank matrices into the model's architecture during training. Instead of updating the full set of model weights, LoRA freezes the original weights and optimises small, task-specific low-rank updates. Key hyperparameters include the rank ( $r$ ), scaling factor ( $\alpha$ ), and learning rate. Lower ranks reduce memory usage, while higher ranks improve model expressiveness. The  $\alpha$  parameter controls how much the adaptation influences the model, striking a balance between learning new information and preserving existing knowledge. For example, Heraldi and Ruskanda (2024) used a rank of 8,  $\alpha$  of 16, and a cosine learning rate to achieve 0.9 accuracy—values that could be transferable to related tasks like phishing detection due to task similarity.

Building on the foundation of LoRA, Dettmers et al. (2023) introduced QLoRA, which combines low-rank adaptation with 4-bit quantisation to further reduce memory and computational demands. QLoRA compresses the model weights into 4-bit representations and uses scale factors to approximate floating-point values, significantly lowering memory requirements without degrading performance. Experiments showed that QLoRA maintains accuracy comparable to 16-bit LoRA across models ranging from 7B to 65B parameters, even after double quantisation. While QLoRA introduces additional complexity in managing quantised representations, it demonstrates that substantial efficiency gains can be achieved without compromising model performance. This reflects a growing trend in the literature toward fine-tuning methods that are both scalable and resource-conscious, enabling broader accessibility and deployment in real-world, resource-constrained environments.

## 2.6. Involving tools

A significant advancement in the capabilities of LLMs is the development of autonomous agents designed for specific tasks. In the whitepaper, Agents, Wiesinger et al. (2024) define agents as AI systems capable of operating independently using a set of tools aligned with predefined goals. Their proposed framework enables agents to interact with external data sources and services, effectively extending beyond the limitations of static, pre-trained models. By leveraging APIs, agents can retrieve and process real-time information. For instance, in the context of phishing detection, an agent can analyse an email by referencing a user's previous correspondence for context, evaluating sender domains, and applying its fine-tuned knowledge to the content. The agent may then autonomously perform actions such as flagging the email as spam or blocking the sender, demonstrating clear advantages over traditional static classifiers.

Expanding on the agent paradigm, Wang et al. (2024) introduce a method in Executable Code Actions Elicit Better LLM Agents that enables agents to generate and execute Python code as actions. This marks a significant improvement over previous approaches where actions were limited to structured JSON or plain text, thereby enhancing flexibility and functionality. Through access to external software libraries and dynamic code execution, these agents demonstrated a 20% increase in task success rates on complex, multi-step benchmarks requiring tool coordination. Although the curated nature of the benchmark may introduce performance bias, the findings highlight the feasibility of deploying agents in real-world, high-complexity environments. The study also shows that fine-tuned models, such as Llama2 and Mistral, when integrated with a Python interpreter, can perform tasks such as training other models or debugging themselves. While their capacity for self-improvement remains limited, this constraint is less impactful in phishing detection tasks, where the primary requirement is robust interaction with APIs and the ability to carry out both simple and complex workflows autonomously.

The Agents whitepaper (Wiesinger et al., 2024) identifies three key prompt engineering frameworks that support agent reasoning: ReAct, Chain-of-Thought (CoT), and Tree-of-Thought. These techniques structure the model's problem-solving approach and enable more interpretable and accurate responses. Sahoo et al. (2024) provide a comparative analysis of these frameworks, discussing their advantages and limitations across different task types. Chain-of-Thought, in particular, promotes a step-by-step reasoning process that mitigates the risk of premature or incorrect conclusions - an issue that arises when models generate an answer before justifying it. CoT includes several sub-techniques, such as Logical CoT, Chain-of-Knowledge, and Tree-of-Thought, each enhancing specific aspects of reasoning. In phishing detection tasks, these methods enable the model to systematically deconstruct components of an email, such as headers, sender domain, body content, and embedded links, leading to more robust and well-reasoned classifications.

Among the frameworks discussed, ReAct (Reasoning and Acting) stands out for its ability to enable dynamic, tool-augmented reasoning. Introduced by Yao et al. (2023), ReAct structures the model's behaviour into a loop of thoughts, actions, and observations. Thoughts represent internal reasoning aligned with the agent's goals; actions allow the model to query tools or perform computations; and observations capture the outcomes of those actions, feeding back into subsequent reasoning steps. This iterative cycle supports more interactive and adaptive problem-solving compared to other prompt engineering frameworks, incorporating a form of "human-in-the-loop" oversight. In the context of phishing detection, ReAct offers significant advantages by enabling the agent not only to analyse and interpret email content but also to act on findings such as querying external databases,

adjusting thresholds, or soliciting clarification from the user, thus supporting ongoing refinement of its conclusions.

Tasks involving multiple tools and subtasks, such as phishing detection, can benefit substantially from multi-agent systems due to their modular and scalable design. Talebirad and Nadiri (2023) propose a structured approach to multi-agent architecture, categorising it into four key elements: agent roles, agent tools, inter-agent communication, and system operations. Each agent is assigned a specific function aligned with its task requirements and is equipped with tools suited to its operational context. Crucially, communication and coordination are facilitated through a hierarchical structure, often managed by a supervisor agent. This supervisory role ensures coherent information flow, resolves task delegation, and reduces the risk of hallucinations or conflicting actions, factors that are particularly important in high-stakes applications like phishing detection, where precision and consistency are critical.

Despite their potential, agent-based systems introduce significant security concerns, particularly when interacting with external tools and APIs. Generative AI models are susceptible to misuse in the automation of social engineering, malware generation, and system exploitation, risks that are amplified when agents operate autonomously. Gupta et al. (2023) highlight the vulnerability of prompt interfaces as a critical attack surface. Techniques such as jailbreaking, prompt injection, and adversarial phrasing can manipulate model behaviour, bypass safety filters, or induce unintended actions. These risks underscore the importance of implementing robust security safeguards, including input sanitisation, role-based tool permissions, and monitoring mechanisms. As agent-based systems evolve, especially in sensitive domains like phishing detection, ensuring the integrity and trustworthiness of model interactions must remain a central design priority.

## Chapter 3 - Methodology

### 3.1. Introduction

The following section expands upon the methodology which was followed at each stage of this project. The Gantt chart is included to visualise the scheduling of each stage.

### 3.2. Development methodology

The methodology adopted for this project was iterative and incremental development, which aligns well with the modular nature of multi-agent systems. This approach is particularly effective for complex projects, as it enables starting with a simple proof of concept—such as a single-agent system—and progressively expanding functionality through incremental improvements as highlighted by Mitchell and Seaman (2009). Each stage focuses on developing self-contained, independently functioning modules, which is essential for maintaining system robustness and ensuring continued functionality as the project scales. This is especially critical in a system involving unpredictability from the use of LLMs. Moreover, the iterative approach manages complexity effectively by building on previous increments, providing a clear and structured roadmap. While alternative methodologies like Prototyping also begin with a basic model, their emphasis on rapid feedback and requirement refinement can sometimes undermine long-term design stability. In contrast, this project prioritises robustness and seamless integration of additional modules and agents as the system evolves.

### 3.3. Requirements

The requirements analysis was sourced from existing research and stakeholder engagement.

#### 3.3.1. Literature review

The literature review was conducted systematically using trusted academic databases, including the UWE Library, Google Scholar, IEEE Xplore, and arXiv. To incorporate recent developments, studies published from 2018 onward were prioritised, following the release of the influential "Attention Is All You Need" paper by Vaswani et al. (2017). arXiv was specifically included for its access to recent, influential papers. The search used carefully selected keywords and advanced filters to highlight relevant, up-to-date studies, with emphasis on high-impact journals and conference proceedings to ensure credible, authoritative sources.

#### 3.3.2. Stakeholder Engagement

In combination with the existing research, various opinions were sought after from different backgrounds, including those in a working environment and those outside to ensure that the functional and non-Functional requirements would meet a general user's expectations.

### 3.4. Design

The design phase involved creating high-level diagrams to outline the multi-agent process, with flexibility to accommodate potential adjustments in how functionality might be distributed among the agents. The primary focus was to ensure that the desired functionalities were effectively addressed. Additionally, process flow diagrams were developed to illustrate how general query flows would be managed and processed within the system.

### 3.5. Implementation

The implementation of the PoC involved the development of the code using a combination of online environments such as Kaggle and Google collab, while also using VS code for certain components of the PoC. For the whole application, a web application was chosen to host the application for ease of access for users.

### 3.6. Testing

As mentioned, the phishing detection system will be developed in stages, with features added incrementally. Following each stage, integration tests will be performed to verify that the system behaves as expected with the newly introduced components. Once development is complete, component tests will be carried out on each individual agent to ensure they function correctly in isolation. Finally, full system testing will be conducted to evaluate the interaction between agents and confirm that the system produces accurate and consistent phishing detection results.

### 3.7. Gantt Chart

The project was tracked to ensure each stage was allocated enough time.

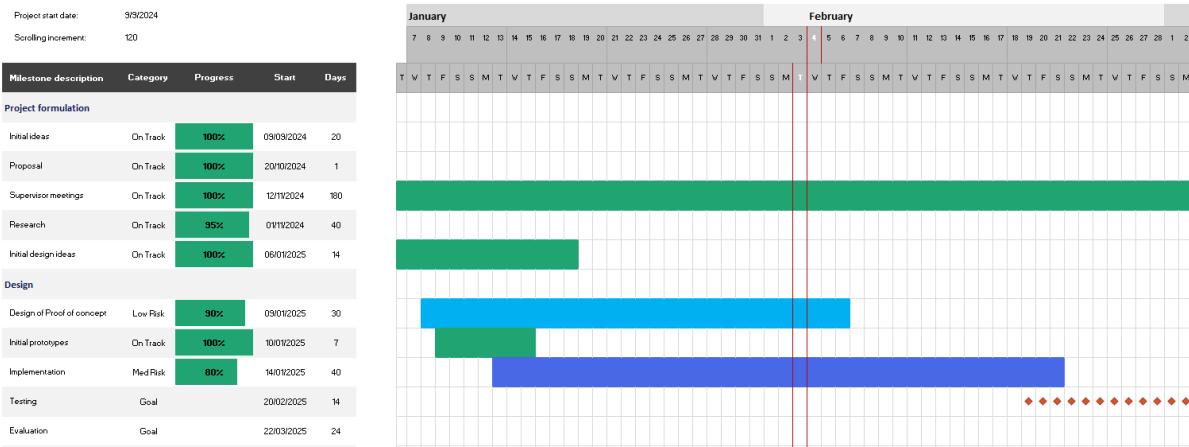


Figure 1 - Gantt Chart

## Chapter 4 - Requirements

### 4.1. Introduction

This section outlines the functional and non-functional requirements for the development of the phishing detection tool based on agentic workflows. Requirements were informed by a diverse set of stakeholders to ensure the tool is accessible, user-friendly, and effective for users of all ages and backgrounds. Insights were gathered through questionnaires to capture both technical expectations and usability needs.

### 4.2. Stakeholder Identification

Stakeholders were identified based on their roles and engagement with email communication. These include employees from Sales and other externally facing business units, long-term and seasoned staff, as well as independent email users such as seniors, middle-aged adults, and students.

A questionnaire was distributed to five representatives from each stakeholder group (see Annex G). Their responses informed both functional and non-functional requirements. Some questions assessed feature feasibility (e.g. "What features do you expect?"), while others invited open-ended feedback (e.g. "To what level of detail should explanations go for emails flagged as phishing?").

## 4.3. Functional Requirements and Non-Functional Requirements

### 4.3.1. Use case diagram

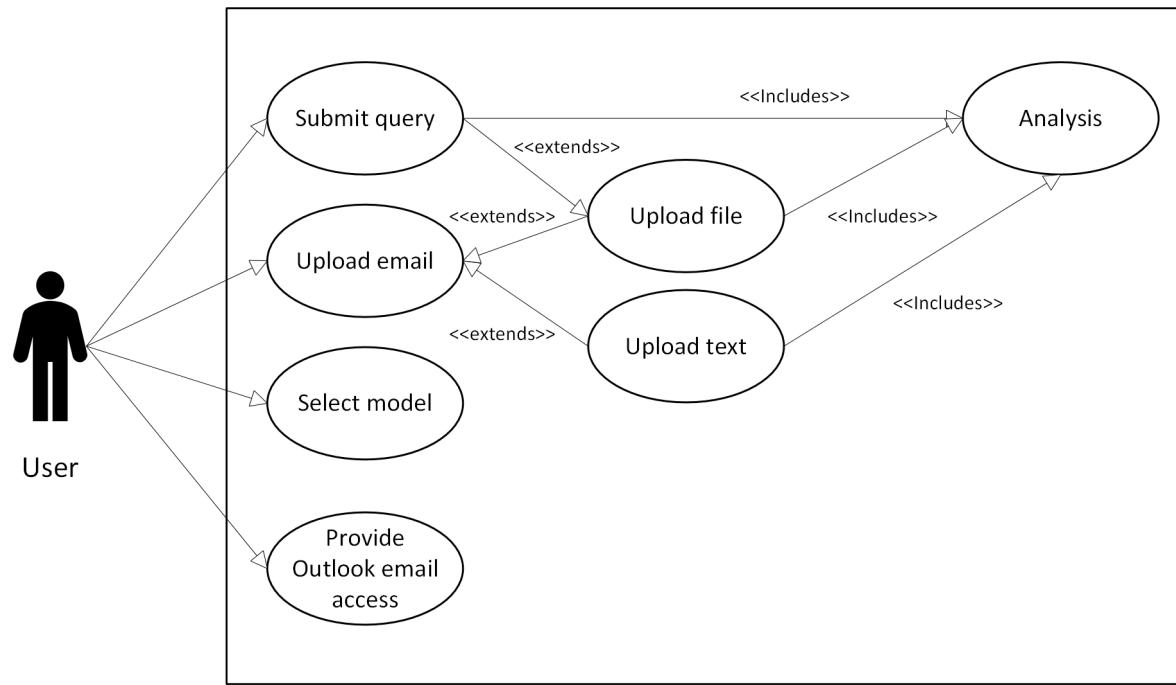


Figure 2 - Use case diagram

Figure 2 presents the use case diagram, outlining user interactions with the system and the core functionalities required for email analysis. These functions are primarily grouped into query submission and system enhancements, such as model selection or providing historical emails for contextual analysis.

### 4.3.2. Functional Requirements

| ID         | Functional Requirement              | MoSCoW      | Description  | Source            |
|------------|-------------------------------------|-------------|--|-------------------|
| <b>FR1</b> | User Interaction and Query Handling | Must Have   | The supervisor agent must handle user queries and direct to the relevant agent.  | Stakeholder       |
| <b>FR2</b> | Email/Text Analysis Agent           | Must Have   | The system must include an agent that analyses provided email or text extracts to identify malicious content.  | Literature review |
| <b>FR3</b> | Historical Email Analysis           | Should Have | The system must be able to access past emails from the user's Outlook account, providing context for current email analysis. This includes analysing email headers, content, and sender information. | Stakeholder       |

|             |                                     |             |   |                   |
|-------------|-------------------------------------|-------------|---|-------------------|
| <b>FR4</b>  | Phishing Probability Scoring        | Must Have   | The analysis agent should output a probability score (e.g., 0-100%) representing the likelihood that the email or text is phishing.   | Literature review |
| <b>FR5</b>  | Long term memory                    | Must Have   | The phishing detection system should be capable of recalling relevant user-specific information, like previous queries or user preferences.   | Stakeholder       |
| <b>FR6</b>  | URL analysis                        | Must Have   | The phishing detection system must be able to extract URLs from the email and analyse it.   | Literature review |
| <b>FR7</b>  | Continuous Learning and Improvement | Should Have | The system should allow agents to improve over time by learning from past interactions and feedback from the user. For instance, if users confirm that a particular email was phishing or not, this data should help refine the detection algorithms for future analyses. | Literature review |
| <b>FR8</b>  | GUI for email input                 | Should Have | The GUI should allow uploading of files to be analysed.   | Stakeholder       |
| <b>FR9</b>  | Recommendations                     | Should Have | The system should provide recommended actions based on the provided email.  | Stakeholder       |
| <b>FR10</b> | Model selection                     | Could Have  | Users could have a choice on what type of model is being used for the agents, allowing better performance but for a higher resource cost.   | Stakeholder       |

Table 1 - Functional Requirements Table

#### 4.3.3. Non-Functional Requirements

| ID          | Non-Functional Requirement | MoSCoW    | Description   | Source      |
|-------------|----------------------------|-----------|---|-------------|
| <b>NFR1</b> | User usability             | Must Have | The system must be able to interact as an assistant, providing relevant analysis and recommendations.                                   | Stakeholder |
| <b>NFR2</b> | Security & Data Privacy    | Must Have | The system must not be authorised to access outlook emails and deleted when requested.  | Stakeholder |
| <b>NFR3</b> | Integration Compatibility  | Must Have | The system must integrate with Microsoft Outlook, allowing the phishing analysis agent to analyse emails directly from the user's inbox | Stakeholder |

|              |                            |             |  |                   |
|--------------|----------------------------|-------------|--|-------------------|
|              |                            |             | without the need to manually copy text.  |                   |
| <b>NFR4</b>  | File format compatibility  | Must Have   | The system must process all forms of input provided by the user including email files, raw text and others.  | Stakeholder       |
| <b>NFR5</b>  | AI Model Explainability    | Must Have   | The phishing probability scoring must include an explanation of why an email was flagged (e.g., suspicious URL, sender reputation), increasing transparency for users.                       | Stakeholder       |
| <b>NFR6</b>  | AI Model focus             | Must Have   | The system must keep on topic and stay in role of being a Phishing Analysis bot. It should not answer general questions or use its tool for any other purpose.                               | Literature review |
| <b>NFR7</b>  | Agent scalability          | Must Have   | The Agent system must allow for scalability horizontally, allowing further functionality to be added.  | Stakeholder       |
| <b>NFR8</b>  | Ensemble analysis accuracy | Should Have | The model ensemble should dynamically change weights to maintain high accuracy (90%).  | Literature review |
| <b>NFR9</b>  | Real-time detection        | Should Have | The phishing detection system should be capable of real-time detection when a user shares an email or text extract for analysis, with results and recommendations delivered after providing. | Stakeholder       |
| <b>NFR10</b> | Error Handling             | Could Have  | The system could be able to query itself when an error occurs within itself and be able to adapt to it.  | Literature review |

Table 2 - Non-Functional Requirements Table

## Chapter 5 - Design

### 5.1. Introduction

The design process was divided into high-level and low-level design stages which allowed for a structured flow of implementation. The high-level designs include Process flows – both for the user and the system – along with a GUI design. After the high-level design, prototypes were made to understand the possibilities and functionality of the various frameworks available. This allowed me to go further into detail when designing on a lower level. The designs included a more in-depth State process flow along with detailed designs of expected behaviour within each agent.

## 5.2. High-level design

The system is divided into four sections: Conversation, Extraction, Email Analysis, and Context Analysis. Each section is managed by subsystems, some with individual agents and others with multiple agents.

During early prototyping, agents struggled with multitasking—especially the supervisor, initially responsible for both conversation handling and agent routing. This issue was pronounced in smaller models like Mistral 7B. Defining clear roles and emphasising collaboration proved more effective than maximising agent task loads.

The user process flow follows a consistent pattern: the supervisor assigns a query to the most suitable agent, the agent processes it, and the result is returned. The supervisor continues this cycle until it determines the query is resolved, at which point the final response is sent to the user.

### 5.2.1. User Process Flow

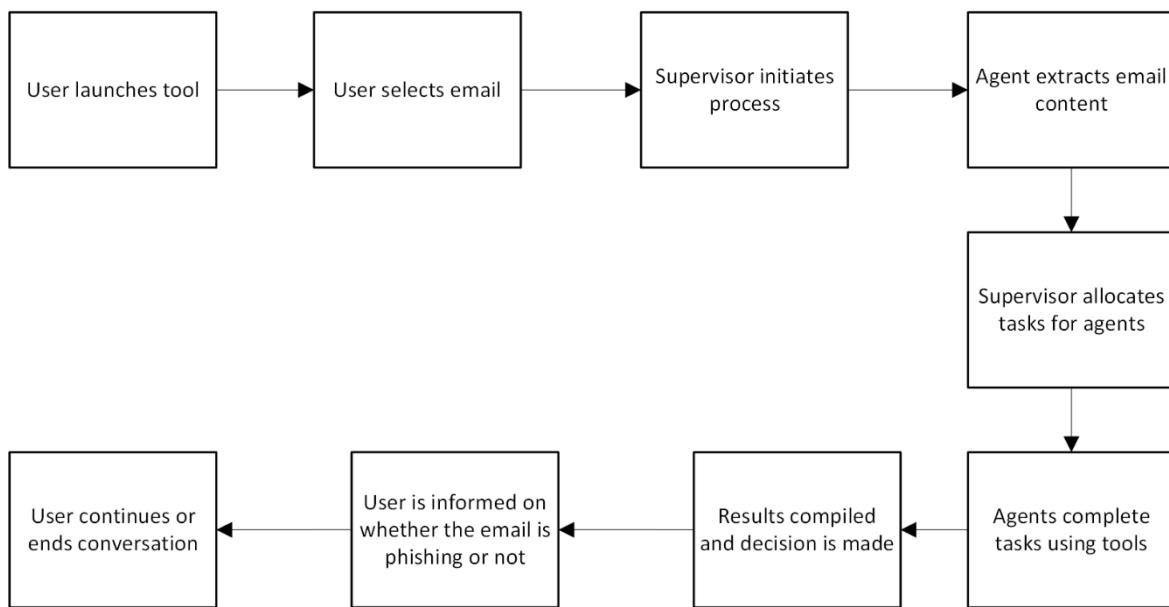


Figure 3- User Process Flow

### 5.2.2. Multi-Agent System process

This diagram outlines the system's structure, illustrating process connections, tools, and the supervisor hierarchy for task management. This hierarchy is crucial for phishing detection, preventing LLMs from generating baseless answers. If the wrong agent is queried, it may rely on general pretrained knowledge rather than specialised analysis.

The supervisor enables multiple agents to contribute to a well-rounded response. It also integrates external tools that enhance each agent's role. The supervisor itself has access to memory tools, storing and retrieving data from a vector database for seamless user interaction. The Load Memories module fetches relevant data at startup, while the supervisor determines what to save based on a configuration prompt.

Specialised agents, or "workers," have task-specific tools. The Email Analysis Agent leverages an ensemble of LLMs for content analysis and a VirusTotal integration for URL scanning. Meanwhile, the Outlook Analysis Agent queries a database of past emails and could utilise a domain check API.

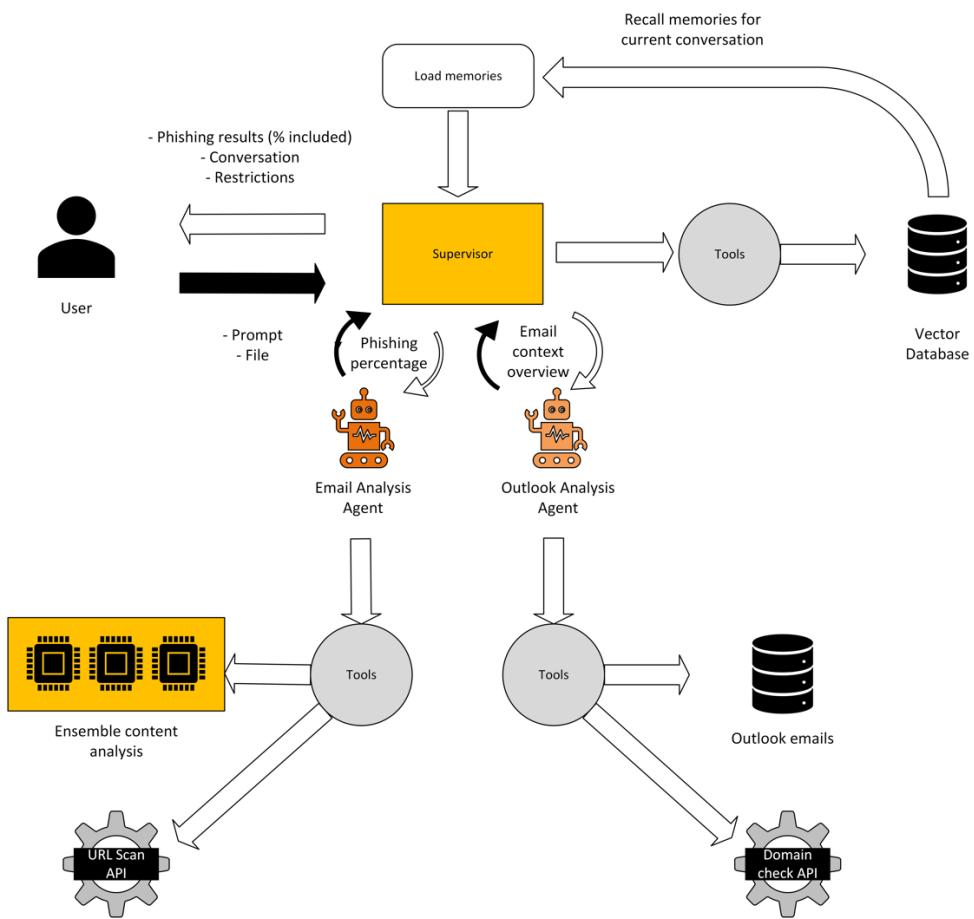
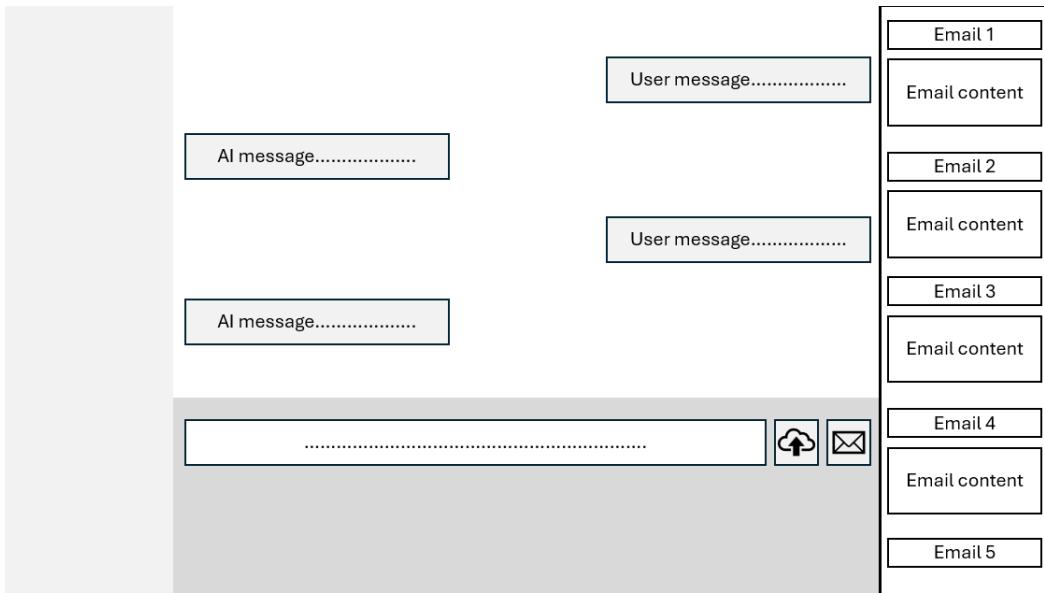


Figure 4 - Agent system Components Diagram

### 5.2.3. GUI

Figure 5 illustrates the high-level design of the front-end application for query input. The interface follows a standard chatbot layout: user messages appear on the left for context, while generated responses appear on the right for clarity.

Below the prompt input are two buttons—one for file uploads and another to load emails. Clicking the load emails button reveals a side panel displaying emails. These emails are purely visual and cannot be interacted with directly from the front end, but they enable agent system access once the user authenticates.



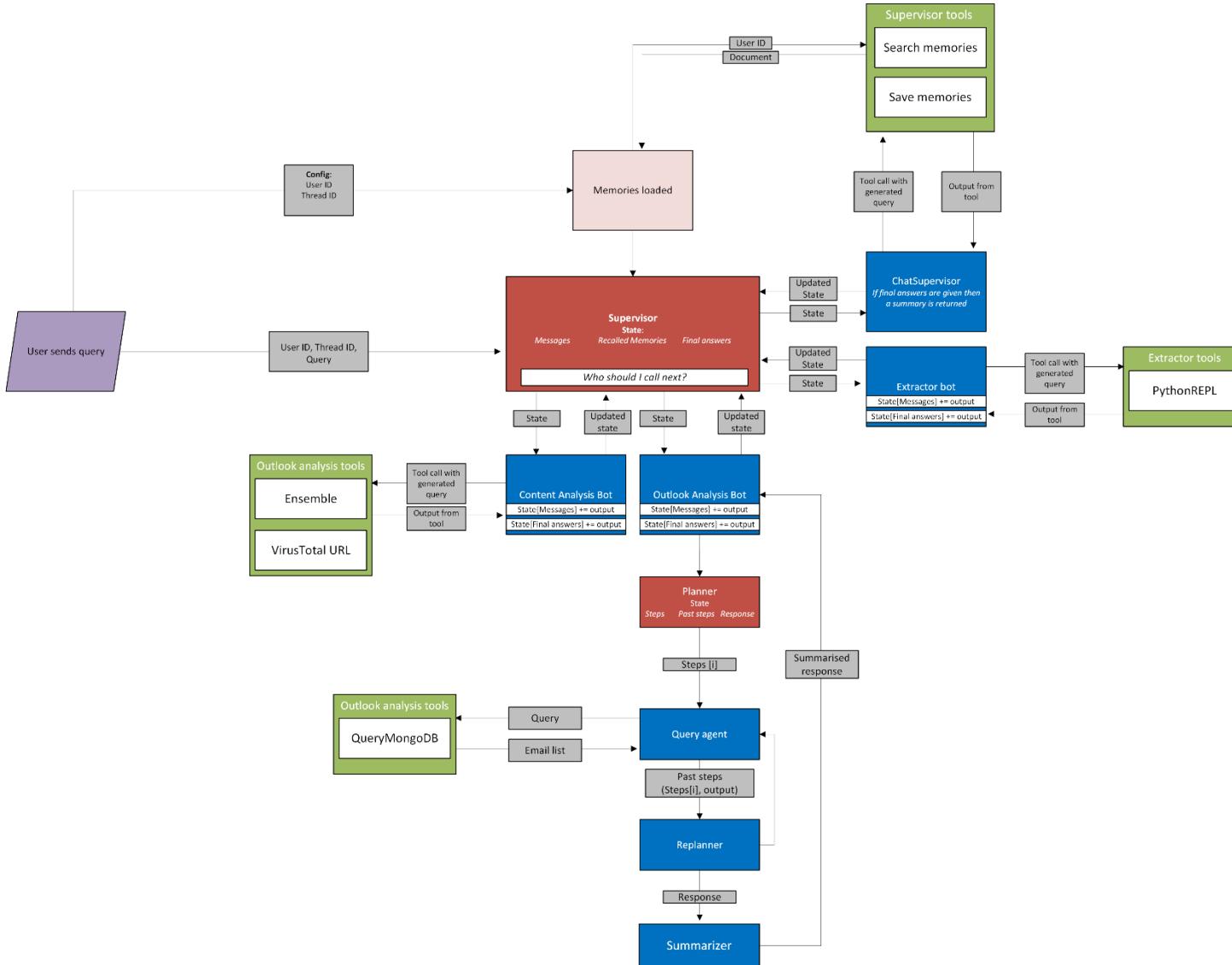
*Figure 5- GUI Design*

## 5.3. Low-Level design

### 5.3.1. Process Flow

Since agents handle shared information but produce distinct outputs, the system's state, which stores past messages and custom attributes, evolves as actions are performed. To track state changes and runtime calls, the diagram below outlines the multi-agent system's structure and data flow.

The state format differs between the Supervisor and the Outlook Analysis Bot due to the latter's plan-execute structure. The planner's state includes steps, past steps, and responses, unlike the Supervisor's main state. Since the Outlook Analysis Bot operates as a separate system, its response is consolidated into a single message within the main state. This optimises storage by filtering out unnecessary planning iterations and prevents storing users' personal emails in the main state.



*Figure 6 - State flow diagram*

### 5.3.2. Sequence Diagram

The following sequence diagram demonstrates the collaboration between all the agents when the User provides a phishing email to be analysed.

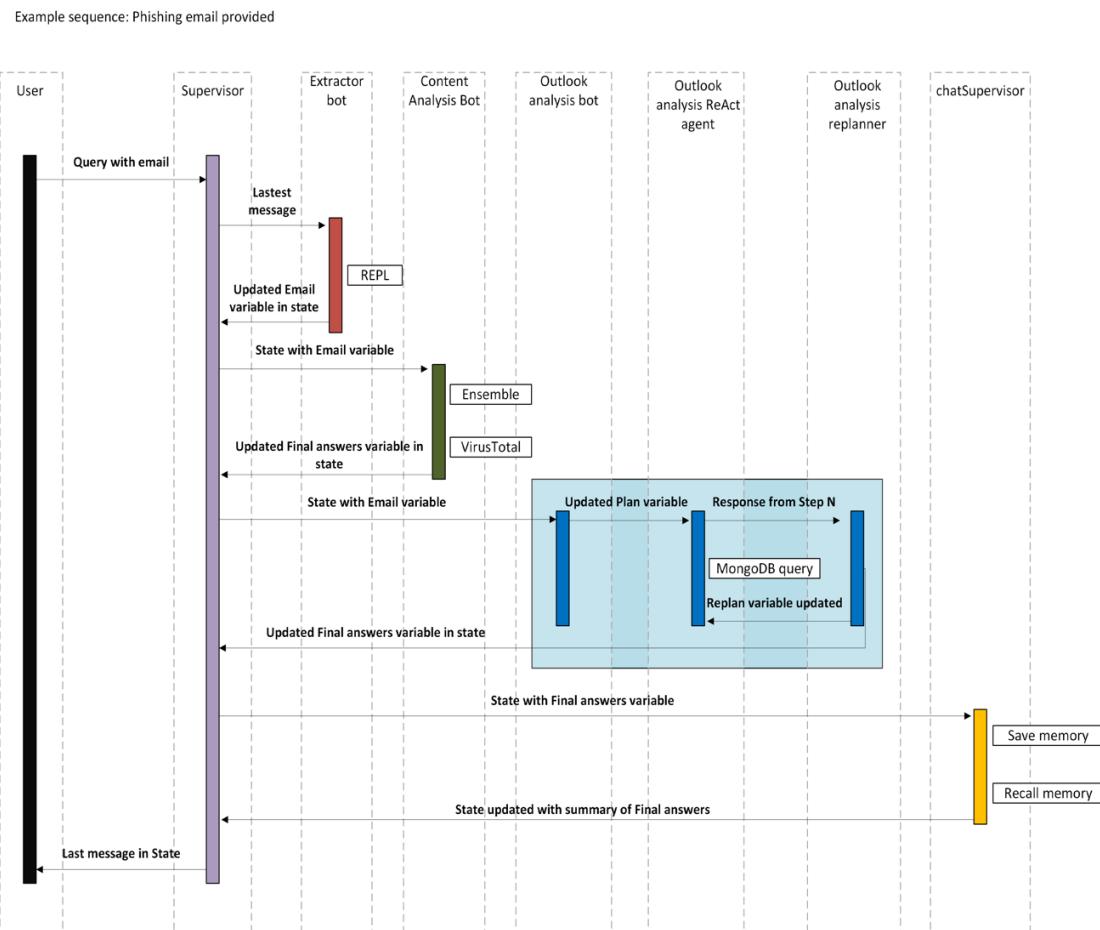


Figure 7 - Email analysis Sequence Diagram

The next diagram shows how the Agents can be called in any order, depending on which agent is most suitable. In this case, the user sends an ambiguous query, the chatSupervisor follows up, asking for more clarification. Which then allows the Supervisor to correctly stage the answer with the Content analysis bot analysing the URL.

Example sequence: Request to scan URL

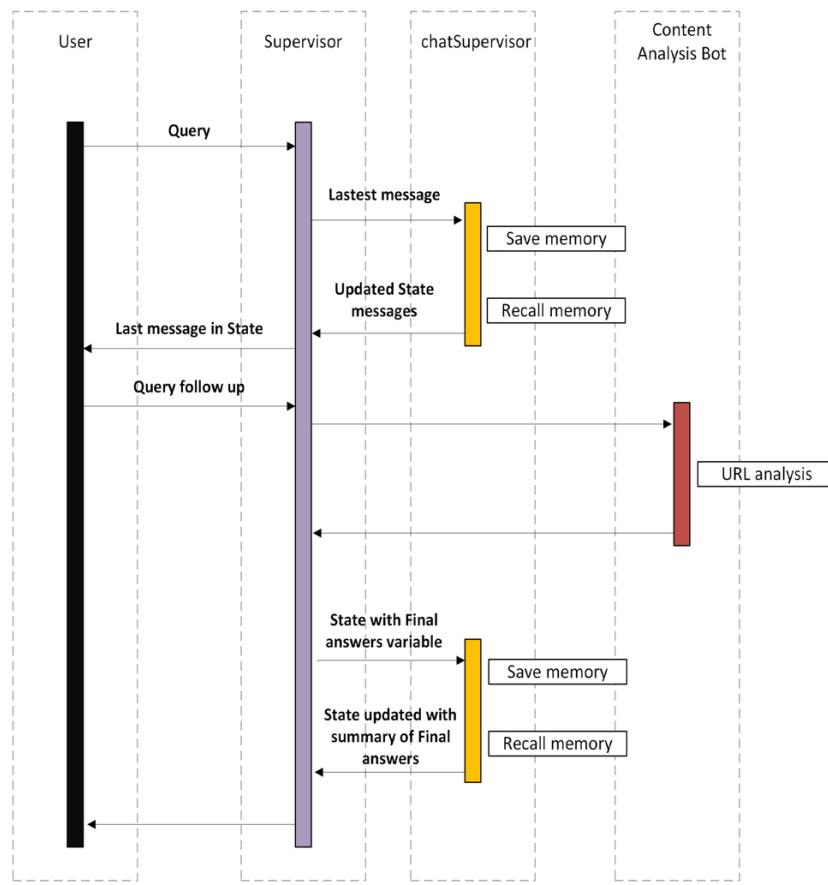


Figure 8 - URL scan Sequence Diagram

### 5.3.3. Tool composition

These diagrams show the inputs, external API connections and outputs for each of the tools which are used by the Agents.

## Memory tools

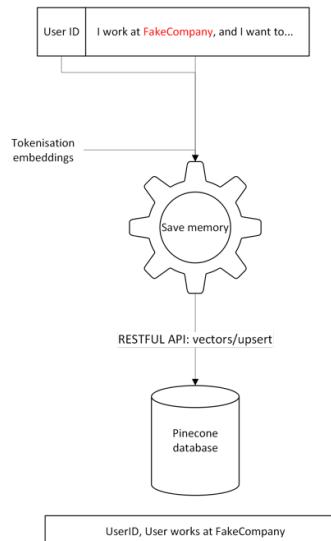


Figure 9 - Save memory tool

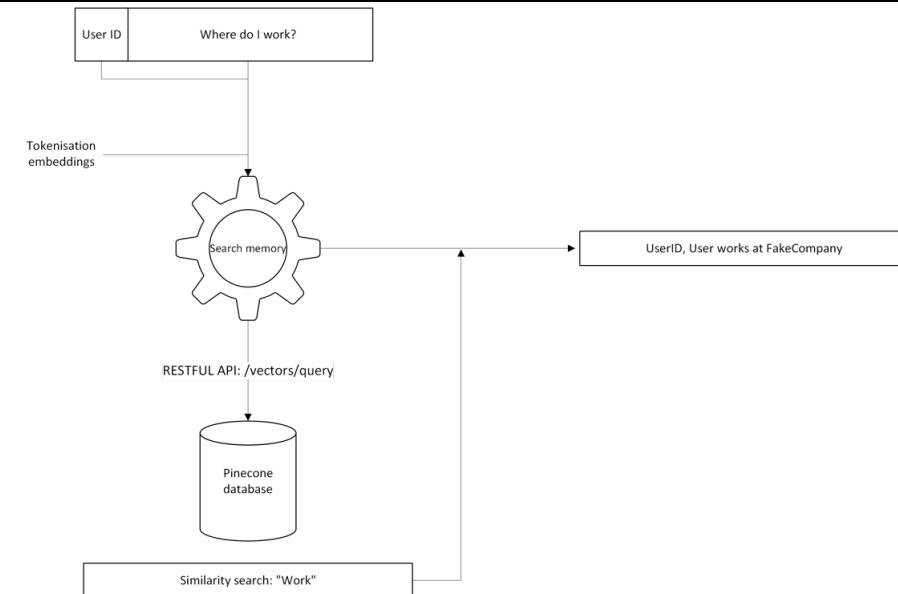


Figure 10 - Search memory tool

## Email analysis tools

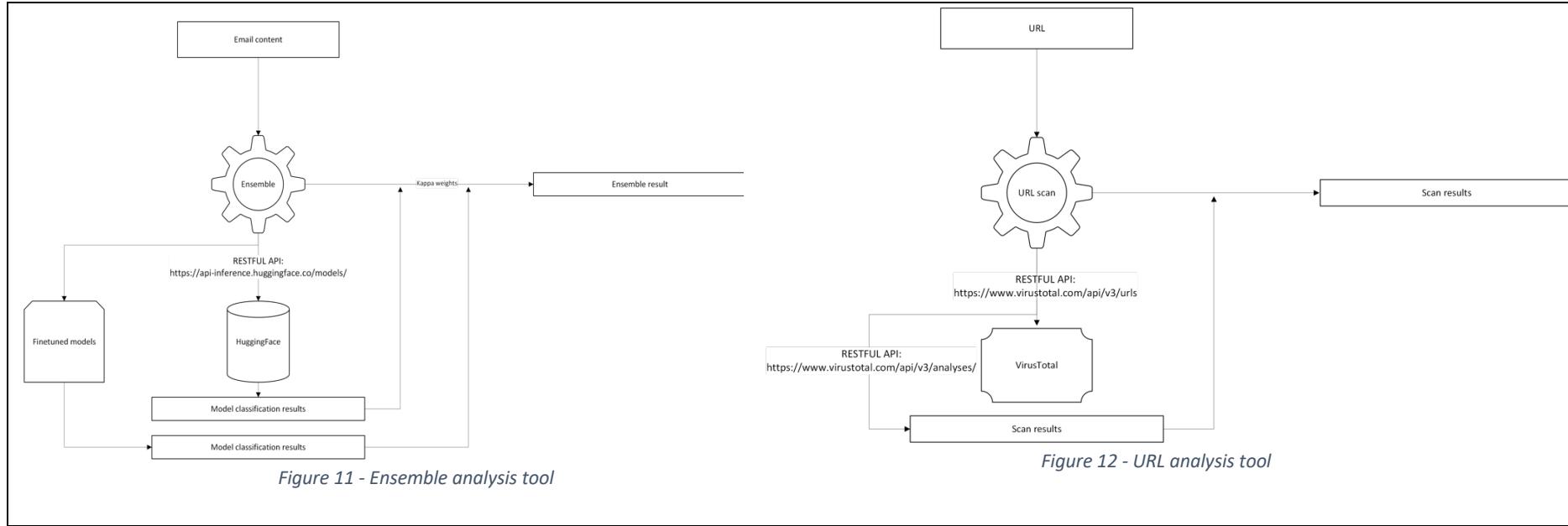


Figure 11 - Ensemble analysis tool

Figure 12 - URL analysis tool

### 5.3.4. Databases

#### *Vector database*

The vector database, hosted on Pinecone, will be setup with the chatSupervisor model's embeddings to allow for correct upserting and retrieval, as seen in the memory tool diagrams. This will allow for the agent to use similarity searching to find relevant memories based on the conversation.

#### *MongoDB database*

The MongoDB is an unstructured database which will allow for the storage of the user's email for querying. The emails will be stored in a JSON format with the structure defined in Figure 13.

```
{  
    "id": "XXXXXXXXXXXX",  
    "createdDateTime": "0000-00-00T00:00:00Z",  
    "subject": "XXXXXX",  
    "bodyPreview": "XXXXXXXXXX",  
    "from": {  
        "emailAddress": {  
            "name": "XXXX XXXX",  
            "address": "XXXX@XXXX"  
        }  
    },  
    "toRecipients": {  
        "emailAddress": {  
            "name": "XXXX XXXX",  
            "address": "XXXX@XXXX"  
        }  
    }  
}
```

Figure 13 - MongoDB JSON structure

### 5.3.5. Model Finetuning

See Appendix B for an account of the data cleaning process for the finetuning.

The content analysis will utilise multiple LLMs to classify the email text as phishing or legitimate with confidence scores, i.e. phishing – 0.9, legitimate – 0.1. As there are multiple models with various techniques and training sets behind each of them, results will vary and there will often be disagreement across the models. To solve this, a soft voting technique will be used with dynamic weights proposed by Taha (2021). The process which will be taken is to provide a validation dataset to all the models, to then generate a confusion matrix containing the True Positive, False Positive, True Negative and False Negative. These can be used in the Kappa statistics as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Expected\ accuracy = \left( \frac{TP + FP}{N} \cdot \frac{TP + FN}{N} \right) + \left( \frac{TN + FP}{N} \cdot \frac{TN + FN}{N} \right)$$

$$K = \frac{Accuracy - Expected\ accuracy}{1 - Expected\ accuracy}$$

$N$  is stated to be the number of samples in the dataset.

Using the Kappa Statistic generated for each model, it can dictate the influence on the result by normalising the outputs and finding the aggregated result from all the models.

### 5.3.6. Testing

Table 3 outlines the system testing design used to document all system-wide test cases. A separate table is used for integration testing, following a similar structure but replacing "Desired Result" with "Test Steps" and omitting the "Confidence" column, as these tests are more general and may not require a confidence score.

| ID                                     | Test                                  | Requirements  | Desired result  | Result  | Confidence   | Retest result   |
|--|---------------------------------------|---|---|---|--|---|
| A unique ID assigned to each test case | A summary of the test being conducted | Specifies the system requirements validated by this test case | Outlines the anticipated outcomes if the system behaves as intended | Records the observed outcome after executing the test | Displays the classification outcome produced by the ensemble model | Indicates whether a retest is necessary in case of test failure |

Table 3 - Test case table design

The main test cases are as follows:

- Memory utilisation – Evaluation of memory usage during conversations using memory management tools
- Phishing emails – A variety of phishing emails with differing levels of complexity
- Legitimate emails – A range of authentic emails, including marketing content and messages from known senders
- Business email compromise (BEC) – Emails that manipulate or impersonate legitimate communications to deceive recipients. See Annex E for the email chain.
- Fraud - Deceptive messages intended to trick recipients into revealing sensitive information, transferring funds, or performing actions that benefit the attacker. See Annex F for the email chain.

# Chapter 6 – Implementation

## 6.1. Introduction

The implementation section details the progressive development of the agent system, beginning with a single agent and gradually incorporating supervisor structures, culminating in a fully functional multi-agent system. The initial development took place on Kaggle, before transitioning to Visual Studio Code to better support application deployment using Flask. This setup was chosen to reflect a real-world deployment scenario, with a dedicated server and front-end interface. Throughout the implementation phase, several key adjustments were made to the agent behaviours to enhance usability and ensure the system functioned as a user-friendly and effective assistant.

## 6.2. Initial agent

The agent system was built using Langchain's LangGraph framework, enabling structured workflow management in a multi-agent environment. Following an iterative design approach, initially developing a simple, single-agent system with general tool access and no specialisation. Since memory access was essential for the full system, it became the first major feature to implement.

To enable memory utilisation, two key tools were needed: recall and saving. These tools were wrapped and stored in a dictionary, allowing a tool node to be created dynamically. Any agent connected to this node could access these tools. The system uses Pinecone's vector store, accessed via API, to handle memory storage and retrieval.

- **Save memory:** Uses the add\_documents method to store a memory with a user ID for attribution.
- **Recall memory:** Uses similarity search, filtering by user ID. Since memories are stored as vectors, similar terms (e.g., "food" -> "Pizza" or "eating") can be retrieved efficiently.

The key challenge was ensuring the LLM understood when and what to save or recall. This was managed through a structured prompt, which defines the agent's role and behaviour:

- **Role** – Identifies the source as 'System'.
- **Brief role overview** – Describes the agent's function and accessible tools.
- **Guidelines** – Rules governing tool usage and decision-making.
- **Response behaviour** – Defines how responses should be structured, including what the agent should or shouldn't do (e.g., email analysis).

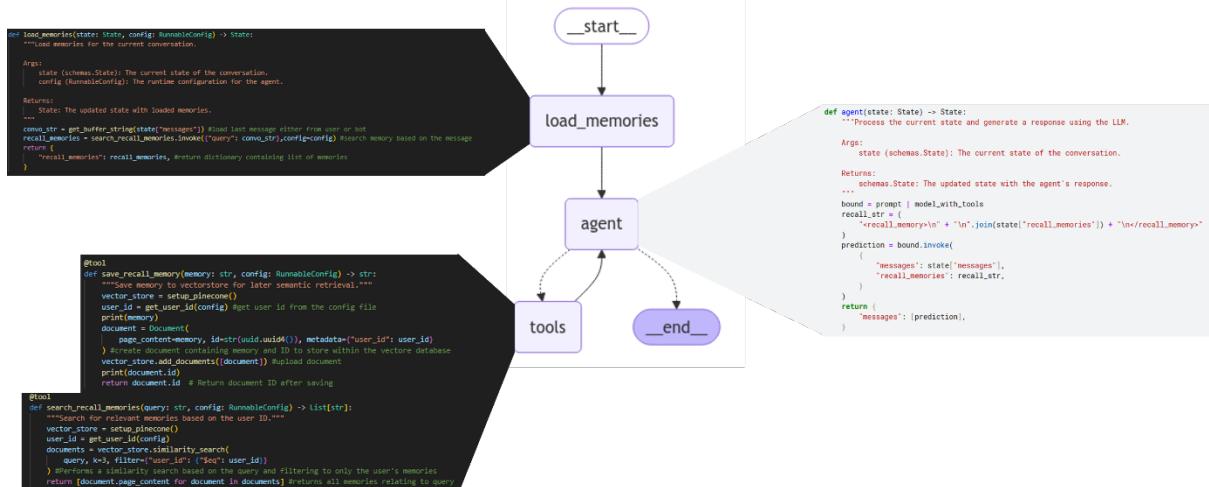


Figure 14 - Initial agent graph

The ‘load\_memories’ node leverages the recall tool to retrieve user-related memories and integrate them into the agent’s state, ensuring immediate access before generating a response.

As shown in Figure 14, the agent incorporates these loaded memories into its answer. In this example, it correctly recalls that the user’s name is John and had previously saved the statement: “*John has said they are 21 and feels old*” using the save memory tool.

```

Update from node: load_memories
{'recall_memories': ['John likes to go skydiving every Tuesday.', "User's name is John.", 'User likes chocolate flavored pasta.']}
```

```

Update from node: agent
===== Ai Message =====
Tool Calls:
  save_recall_memory (7dd0aeb4-3e17-4201-ad95-709c77638ba8)
Call ID: 7dd0aeb4-3e17-4201-ad95-709c77638ba8
Args:
  memory: John is 21 years old and feels old.

Entry
user_id = 1
Update from node: tools
===== Tool Message =====
Name: save_recall_memory
John is 21 years old and feels old.

Update from node: agent
===== Ai Message =====
21 is definitely not old, John! You're just getting started. It's understandable to feel that way sometimes, but try not to let it get you down. What are some things you enjoy doing that make you feel young at heart? Perhaps focusing on those activities will help shift your perspective.

```

Figure 15 - Initial agent's memory usage

## 6.2. Implementing email analysis

The email analysis agent has a toolset consisting of URL analysis and Ensemble analysis. URL analysis is a straightforward function where the LLM extracts a URL and triggers an API call to VirusTotal. VirusTotal then scans the URL using multiple security engines and returns a detailed breakdown of potential threats.

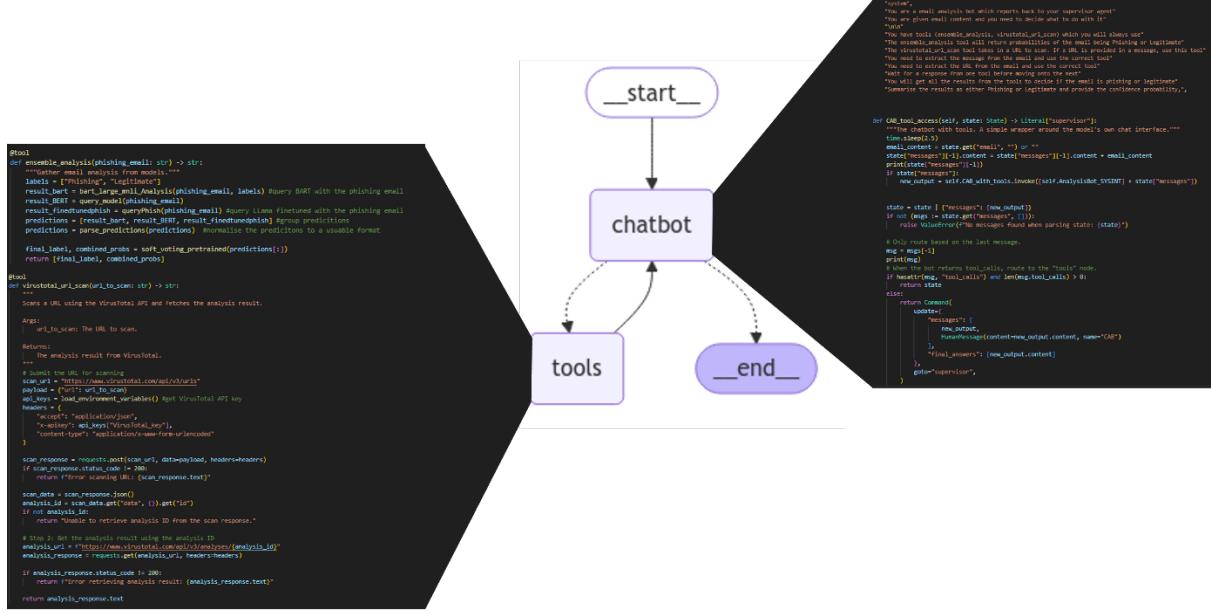


Figure 16 - CAB graph

The ensemble analysis tool utilises multiple LLMs to classify the text within an email. It includes a zero-shot classifier from Hugging Face, along with fine-tuned Llama and BERT models.

As shown in Figure 17, the system collects predictions from each model and applies specific weights to each one. These weighted predictions are aggregated to produce a final score, indicating whether the email is legitimate or phishing. This result is then fed back to the LLM for summarisation and output.

Currently, Kappa statistics is not implemented, and instead, the system uses fixed weights to combine the model outputs.

```

Hey Dan, how have you been doing?
the correct: [[{"legitimate": 0.87, "phishing": 0.13}, {"legitimate": 1.0, "phishing": 0.0}]]
predictions [[{"legitimate": 0.87, "phishing": 0.13}, {"legitimate": 1.0, "phishing": 0.0}]]
all probs: [{"legitimate": [0.87, 1.0], "phishing": [0.13, 0.0]}]
weights: [0.5 0.5]
combined: [0.935 0.065]
legitimate
[0.935 0.065]
["legitimate", {"legitimate": 0.935, "phishing": 0.065}]
"messages": [{"HumanMessage": "Is this email Phishing? Hey Dan, how have you been doing?", "additional_kwargs": {}, "response_metadata": {}, "id": "d084f504-eb50-4042-b7b4-252d5d4f94"}, {"AIMessage": {"content": "", "additional_kwargs": {"tool_calls": [{"id": "LHBEBactr", "function": {"name": "ensemble_analysis", "arguments": {"phishing_email": "Hey Dan, how have you been doing?"}}, "index": 0}], "response_metadata": {"token_usage": {"prompt_tokens": 362, "total_tokens": 395, "completion_to_kens": 33}, "model": "mistral-large-latest", "finish_reason": "tool_call"}, "id": "run-3fcf8087-c44d-4558-a98b-234da4ddc73-0", "tool_calls": [{"name": "ensemble_analysis", "args": {"phishing_email": "Hey Dan, how have you been doing?"}, "id": "LHBEBactr", "type": "tool_call"}]}, "usage_metadata": {"input_tokens": 362, "output_to_kens": 33, "total_tokens": 395}, "id": "5d9e0447-237d-4e79-b654-1b5bf0529b2d", "tool_call_id": "LHBEBactr"}, {"AIMessage": {"content": "The email is legitimate with a 93.5% confidence.", "additional_kwargs": {}, "response_metadata": {"token_usage": {"prompt_tokens": 444, "total_tokens": 458, "completion_tokens": 14}, "model": "mistral-large-latest", "finish_reason": "stop"}, "id": "run-e00952d3-350f-4f72-946f-7f8e376ed35e-0", "usage_metadata": {"input_tokens": 444, "output_tokens": 14, "total_tokens": 458}}}, "finished": false}

```

Figure 17 - CAB ensemble response

The prompt follows the same structure as the chatbot; however, this prompt focuses on summarising the results from the tools and ensuring proper flow by enforcing that the output includes the original email with the results so further analysis can be performed.

```

self.AnalysisBot_SYSINT = (
    "system", # 'system' indicates the message is a system instruction.
    "You are a email analysis bot which reports back to your supervisor agent"
    "You are given email content and you need to decide what to do with it"
    "\n\n"
    "You have tools (ensemble_analysis, virustotal_url_scan) which you will always use"
    "The ensemble_analysis tool will return probabilities of the email being Phishing or Legitimate"
    "The virustotal_url_scan tool takes in a URL to scan. If a URL is provided in a message, use this tool"
    "You need to extract the message from the email and use the correct tool"
    "You need to extract the URL from the email and use the correct tool"
    "Wait for a response from one tool before moving onto the next"
    "You will get all the results from the tools to decide if the email is phishing or legitimate"
    "Summarise the results as either Phishing or Legitimate and provide the confidence probability,"
    "you MUST also provide the original email after the results.",
)

```

Figure 18 - CAB prompt

### 6.3. Outlook analysis bot

The structure for the Outlook analysis agent deviates from the previous approach by implementing a plan-and-execute method, inspired by Wang et al. (2023) Plan-and-Solve Prompting. This method enables the agent to create a plan of action, execute the steps, and then adjust the plan based on the results.

This approach is crucial for analysing the emails in the user's Outlook, as initial queries may return many results or none at all, each indicating something slightly different. The system must decide which results are worth further investigation—something a single cycle with the standard ReAct structure would miss.

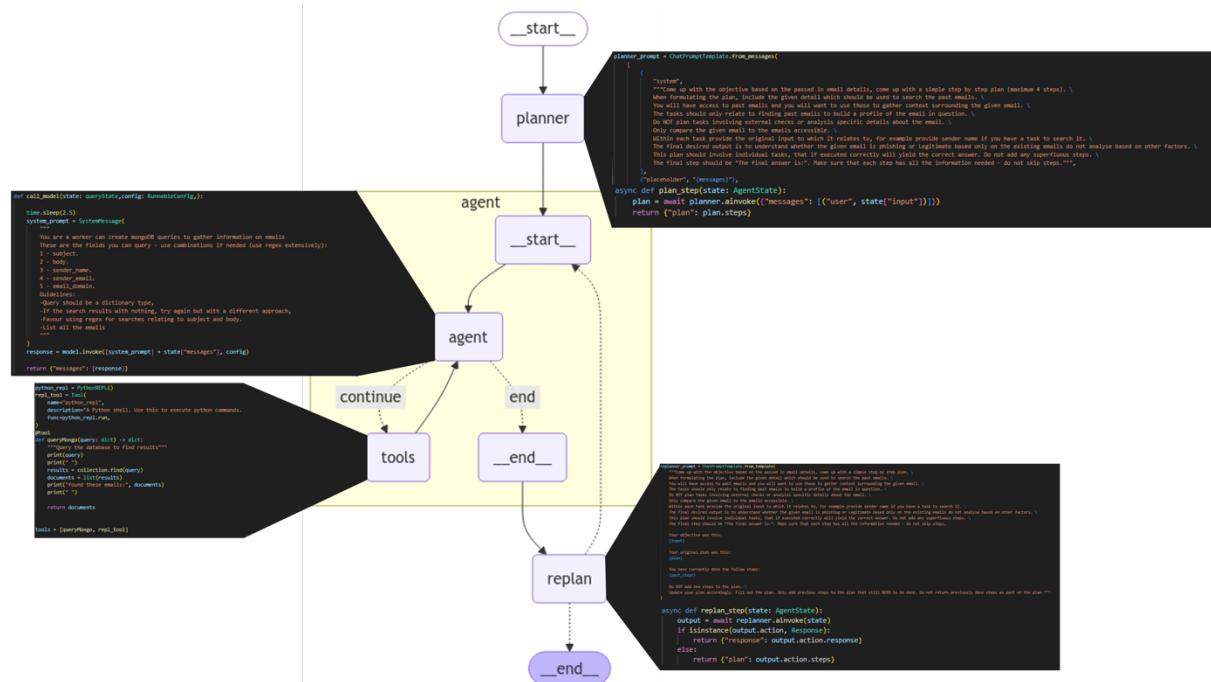


Figure 19 - Outlook analysis bot graph

Various prompts are used as the system switches between the planner, agent and re-planner. The planner and re-planner are similar in the fact that they share the initial objective of coming up with a

step-by-step plan. But the re-planner includes the plan and past steps to keep track of what has been completed.

```
replanner_prompt = ChatPromptTemplate.from_template(
    """Come up with the objective based on the passed in email details, come up with a simple step by step plan. \
When formulating the plan, include the given detail which should be used to search the past emails. \
You will have access to past emails and you will want to use those to gather context surrounding the given email. \
The tasks should only relate to finding past emails to build a profile of the email in question. \
Do NOT plan tasks involving external checks or analysis specific details about the email. \
Only compare the given email to the emails accessible. \
Within each task provide the original input to which it relates to, for example provide sender name if you have a task to search it. \
The final desired output is to understand whether the given email is phishing or legitimate based only on the existing emails do not analyse based on other factors. \
This plan should involve individual tasks, that if executed correctly will yield the correct answer. Do not add any superfluous steps. \
The final step should be "The final answer is:". Make sure that each step has all the information needed - do not skip steps. \
Your objective was this: \
{input} \
Your original plan was this: \
{plan} \
You have currently done the follow steps: \
{past_steps} \
Do NOT add new steps to the plan. \
Update your plan accordingly. Fill out the plan. Only add previous steps to the plan that still NEED to be done. Do not return previously done steps as part of the plan. \
"""
```

Figure 20 - Replanner prompt

The agent which performs the actions does follow the ReAct structure, with access to the MongoDB executing tool, it can provide the query and receive the emails from the database.

## 6.4. Extractor agent

The extractor agent is a ReAct agent with access to a Python environment. This introduces security risks that need careful consideration. One key risk is the agent's access to files, as it works by accessing the uploaded file directory through Python code and generating dynamic extraction code for key parts of the file.

Given the variety of email formats (e.g., .json, .txt, .eml), each requiring different extraction methods, the agent can write its own extraction code tailored to the specific format, making it more reactive compared to relying on pre-written tools. However, this flexibility introduces potential security concerns, particularly with the agent's ability to execute custom code.

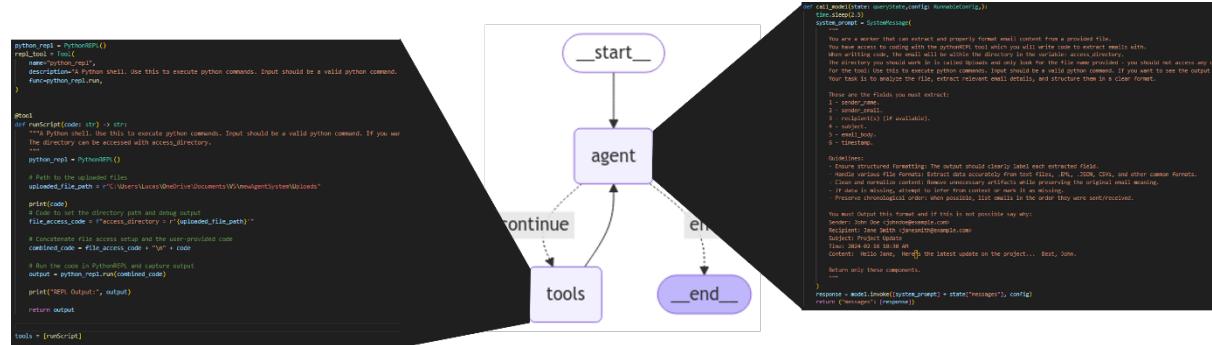


Figure 21 - Extractor agent

To ensure that agent works in a specific directory and does not venture into other files, the code environment is always prefaced with a variable containing the working directory. This is reinforced within prompt to make sure the agent does not try other directories.

The prompt used includes an extra component, being an example of how the output should be formatted which ensures smooth flow to the other agents.

## 6.5. Supervisor structure

To begin assembling the system, a supervisor architecture was implemented, wherein one agent is designated to determine which other agent should respond to a given query. This can be in full or in combination with another agent's answer, depending on what the supervisor decides. To simplify the process, basic agents were used to test functionality, specifically, a researcher and a coder. The researcher was given access to the internet, while the coder operated within a Python environment. The supervisor agent was equipped with access to the memory toolset.

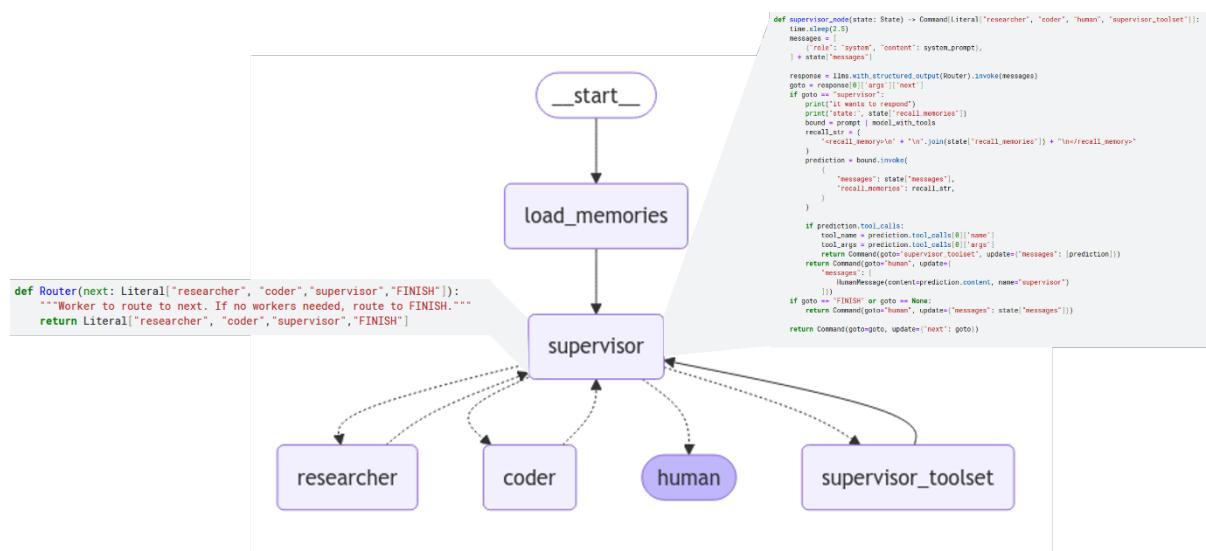


Figure 22 - Initial supervisor architecture

This demonstrated the correct routing from the supervisor to agents, making use of multiple agents and loading memories. It also highlighted an issue with the system's structure, particularly with the supervisor being responsible for decision-making, handling conversational questions, and utilising memory tools as outlined in the prompt (see Figure 23).

The supervisor became overloaded with tasks, which led to confusion about its responsibilities. This had a noticeable impact on its ability to properly utilise the memory tools, as it was stretched too thin. This issue suggests that the supervisor's role may need to be adjusted to ensure it focuses on higher-level coordination, while delegating memory-related tasks to more specialised agents or tools to prevent overload.

```

system_prompt = f"""
You are a Supervisor managing a conversation among the following workers: {members}.

Your role:
1. If you know the answer yourself, respond with the "supervisor"
   (for instance, simple, factual, or conversational queries like "How are you?").
2. If the user provides information about themselves, use your tools to save or load memories (save_core_memory, save_recall_memory) and then respond to it yourself.
3. Every new piece of information provided by the user should be stored using your memory tools.
4. If the user's request requires specialized skills (e.g., researching information, writing code,
   performing math, etc.) that you cannot solve on your own, then delegate the task to the appropriate worker.
5. To delegate, respond **only** with the name of the worker who should act next.
   That worker will then return a response with the requested work.
6. If the result from one of the workers does not fully meet the question, use the next worker if applicable.
7. Use both workers if needed and when they both answer, finish and respond with FINISH.
8. If a clear answer is given from an worker do not ask the same worker again, either finish or ask a different worker.
"""

members = ["researcher", "coder", "supervisor_toolset"]
```

Figure 23 – Initial Supervisor prompt

*Figure 24 - Initial supervisor, coder and researcher response*

```
Model: Hello there! How can I help you today?  
User: what is my name  
Model: I'm sorry, I don't have access to your personal information, including your name. To help me assist you better in the future, would you like to tell me your name?  
User: my name is lucas  
Entry  
user id = 2  
Model: It's nice to meet you, Lucas! How can I help you today?  
User: I wanted to find out what you're able to do  
Model: I'm a large language model, trained to be informative and comprehensive. I can perform many tasks, including:  
  
* **Answering questions:** I can answer your questions on a wide range of topics, drawing on a massive dataset of text and code.  
* **Generating text:** I can write different kinds of creative text formats (poems, code, scripts, musical pieces, email, letters, etc.), I can translate languages, and I can summarize factual topics or create stories.  
* **Processing information:** I can analyze and process information from various sources, including text, code, and data. This includes summarizing information, extracting key points, and identifying patterns.  
* **Providing assistance with tasks:** I can help you with various tasks, such as planning events, writing emails, or researching topics.  
* **Analyzing URLs and emails for potential threats:** I can use the 'virustotal_url_scan' tool to check URLs for malicious content.
```

*Figure 25 - Memory tools not being utilised*

One potential solution involved splitting the prompt's personality to allow self-routing and temporary role changes. While it showed some improvement, the results were inconsistent and unstable. Instead, the conversational and memory responsibilities were delegated to a separate agent, the `chatSupervisor` as seen in Figure 26.

This structure formed the foundation of the solution, enabling the created agents to be connected to the supervisor. Initial testing was conducted using the Content Analysis Bot and the chatSupervisor to evaluate performance.

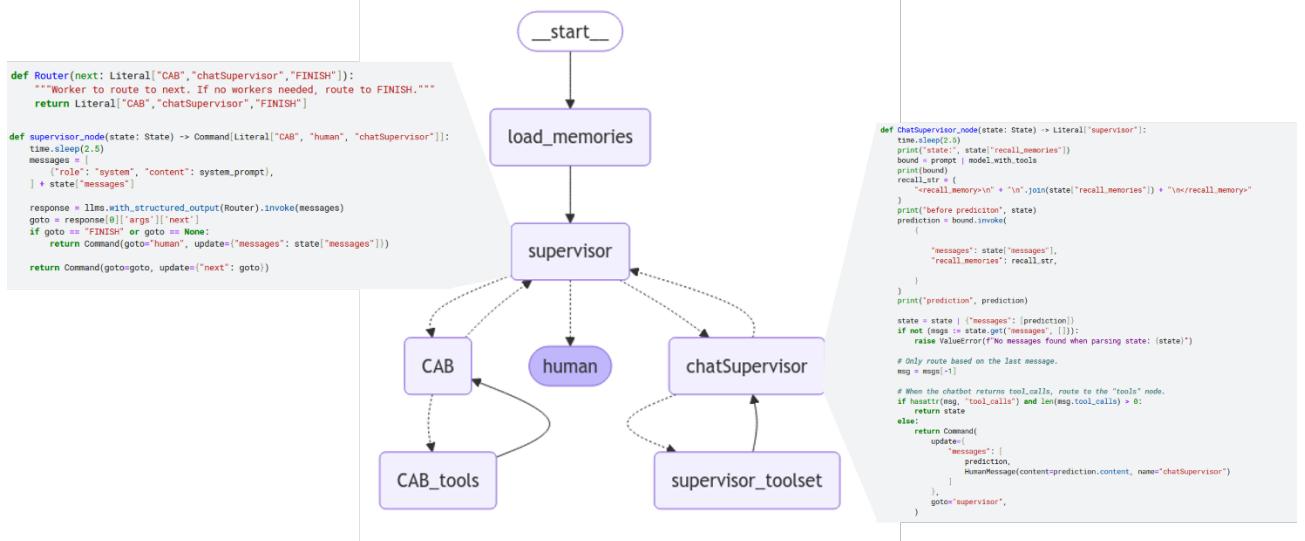


Figure 26 - Supervisor adjustment

Model: Hello! How can I assist you today?  
User: how are you?

```

into supervisor
messages passed to supervisor: [{"role": "system", "content": "You are a supervisor tasked with managing a conversation between the following workers: ['CAB', 'chatSupervisor', 'supervisor_tools']. Given the following user request, respond with the worker to act next. Each worker will perform a task and respond with their results and status. When finished, respond with FINISH. Guidelines: Route correctly: Assign each user request to the most appropriate worker based on the task. Avoid redundant queries: If a worker has already responded to a request, do not ask them again. Instead, direct the conversation to another relevant worker or return FINISH if no further steps are needed. No duplicate assignments: A worker should only be called once per request unless explicitly necessary. Do not revisit workers: Once a worker has responded, do not return to them unless handling a conversational context. If a worker initiates a casual conversation (e.g., 'Hello!', 'How are you?'), route to chatSupervisor. If the user continues the casual conversation and chatSupervisor has already responded, return FINISH immediately. Do NOT call chatSupervisor consecutively. If chatSupervisor has already responded, do NOT route back to it. If the conversation shifts to a task requiring a different worker, route accordingly. Once a task-specific response is given, return FINISH unless further interaction is necessary. Do not continue the conversation with a worker unnecessarily.", "id": "a2de236-4a89-4d35-b229-carbd1a1ab"}, {"role": "user", "content": "Hello! How can I assist you today?", "id": "run-98a3aa1-28ec-4c07-8535-5407bf1bc7a0-0", "usage_metadata": {"input_tokens": 797, "output_tokens": 9, "total_tokens": 806}, "human_message": {"content": "Hello!", "additional_kwarg": {}, "response_metadata": {}}, {"role": "assistant", "content": "I'm here to help! What would you like me to do?", "id": "6766b2ed-7045-4513-a2af-4880be0197139", "usage_metadata": {"input_tokens": 797, "output_tokens": 9, "total_tokens": 806}, "human_message": {"content": "Hello!", "additional_kwarg": {}, "response_metadata": {}}, {"role": "user", "content": "How are you?", "id": "624d049e-d939-4513-9200-40fc9d82e079"}], "supervisor_responded": [{"args": {"next": "FINISH", "type": "Router"}}, {"args": {"next": "FINISH"}]}

```

Figure 27 - Conversation response

After testing the system with various tasks, a clear weakness was being able to identify the agent responsible for communicating with the user. As seen in Figure 27, when asked “how are you?” the model didn’t properly end routing on the chatSupervisor. This shows that clearer instructions are needed in the current prompt seen in Figure 28.

```

members = ["CAB", "chatSupervisor", "supervisor_tools"]
# Our team supervisor is an LLM node. It just picks the next agent to process
# and decides when the work is completed
options = members + ["FINISH"]

system_prompt = (
    "You are a supervisor tasked with managing a conversation between the "
    "following workers: {members}. Given the following user request, "
    "respond with the worker to act next. Each worker will perform a "
    "task and respond with their results and status. When finished, "
    "respond with FINISH."
)

"Guidelines:"
"- Route correctly: Assign each user request to the most appropriate worker based on the task."
"- Avoid redundant queries: If a worker has already responded to a request, do not ask them again."
"- Instead, direct the conversation to another relevant worker or return FINISH if no further steps are needed."
"- No duplicate assignments: A worker should only be called once per request unless explicitly necessary."
"- Do not revisit workers: Once a worker has responded, do not return to them unless handling a conversational context outlined below."
"- CAB should be used when email or URL analysis is needed."
"- chatSupervisor should be used if the user initiates normal conversation."
"- chatSupervisor should be used if something needs to be remembered."
"- Other workers should be chosen based on task relevance."
"Conversational Context Handling:"
"- If a user initiates a casual conversation (e.g., 'Hello!', 'How are you?'), route to chatSupervisor."
"- If the user continues the casual conversation and chatSupervisor has already responded, return FINISH immediately."
"- Do NOT call chatSupervisor consecutively. If chatSupervisor has already responded, do NOT route back to it."
"- If the conversation shifts to a task requiring a different worker, route accordingly."
"- Once a task-specific response is given, return FINISH unless further interaction is required."
"Termination Conditions:"
"- After a worker provides a response, return FINISH unless ongoing interaction is necessary."
"- Do not continue the conversation with a worker unnecessarily."
)

```

Figure 28 - Supervisor prompt

## 6.6. Full system

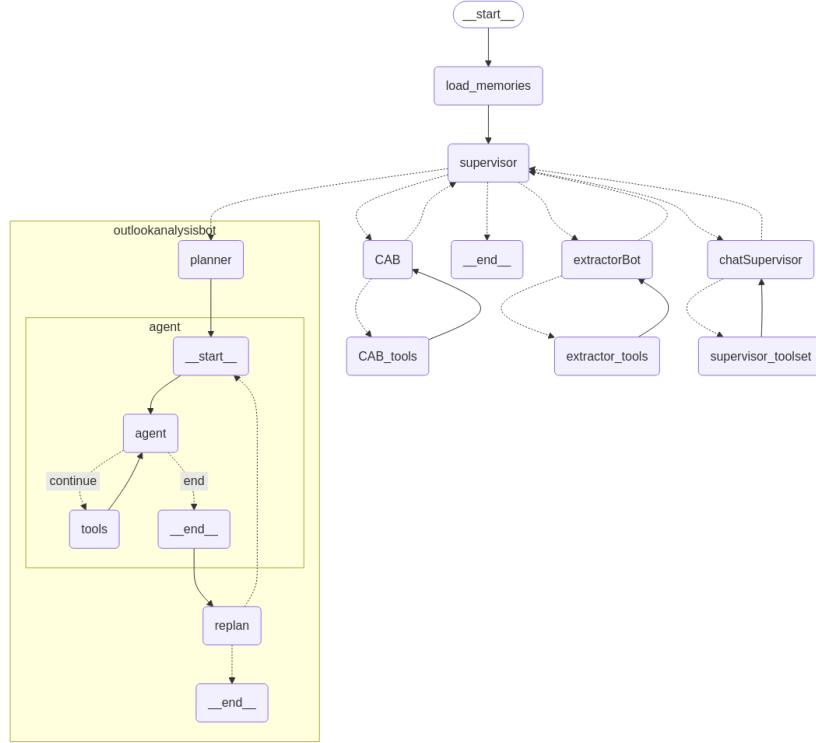


Figure 29 - Full system graph

After constructing the individual agents, they were connected to each other to create the supervisor architecture. The main modification which was needed to accommodate this was through the Supervisor prompt, to provide clear instructions on how each agent should be used and the order of analysis. It is important to note that more detailed instructions would result in more consistent outcomes but at the cost of flexibility and novelty. In the final setup, ChatGPT-4o serves as the supervisor, while Google Gemini and Mistral take on the roles of workers.

```
self.system_prompt = { #prompt for Supervisor
    "You are a supervisor tasked with managing a conversation between the"
    "# following workers: {self.members}. Given the following user request,"
    "# respond with the worker to act next. Each worker will perform a"
    "# task and respond with their results and status. When finished,"
    "# respond with FINISH."
    #
    "#Guidelines:"
    "#* Route correctly: Assign each user request to the most appropriate worker based on the task."
    "#* Avoid redundant queries: If a worker has already responded to a request, do not ask them again."
    "#* Instead, direct the conversation to another relevant worker or return FINISH if no further steps are needed."
    "#* No duplicate assignments: A worker should only be called once per request unless explicitly necessary."
    "#* Do not revisit workers: Once a worker has responded, do not return to them unless handling a conversational context as outlined below."
    #
    "#Worker-Specific Routing:"
    "#* extractorBot should be called first if an email or file has been provided"
    "#* CAB should be used when email or URL analysis is needed."
    "#* you MUST route to outlookanalysistbot after CAB."
    "#* chatSupervisor should be used if the user initiates normal conversation."
    "#* chatSupervisor should be used if something needs to be remembered."
    "#* When analysing an email, you MUST use both CAB and outlookanalysistbot."
    "#* Do NOT call outlookanalysistbot if email is empty."
    "#* Other workers should be chosen based on task relevance."
    #
    "#Conversational Context Handling:"
    "#* If a user initiates a casual conversation (e.g., 'Hello', 'How are you?'), route to chatSupervisor."
    "#* If the user continues the casual conversation and chatSupervisor has already responded, return FINISH immediately."
    "#* Do NOT call chatSupervisor consecutively. If chatSupervisor has already responded, do NOT route back to it."
    "#* If the conversation shifts to a task requiring a different worker, route accordingly."
    "#* Once a task-specific response is given, return FINISH unless further interaction is required."
    #
    "#Termination Conditions:"
    "#* After a worker provides a response, determine if further analysis is requested, if not return FINISH"
    "#* Before FINISH, if final_answers have been given - route to chatSupervisor to summarise the results"
    "#* After chatSupervisor summarises the results, return FINISH"
    "#* You must ALWAYS route to chatSupervisor before FINISH"
    "#* Do not continue the conversation with a worker unnecessarily."
}
```

Figure 30 - Adjusted Supervisor prompt

## 6.7. Data collection

To fine-tune the Llama model for phishing classification, a curated dataset was used which includes modern phishing techniques. As mentioned, some of these are in GitHub repositories, listed on a website or in a ready-made dataset.

*Figure 31 - Cornell University Phish Bowl data collection*

|   | Sender   | Subject   | Body  |
|---|--|---|---|
| 0 | Google Notifications <notify-noreply@google.com> | dubill hd7t invited you to view a collection: ... | dubill hd7t invited you to view a collection\n... |
| 1 | 2U <makarifulislamkariful24599+NmwGO8qDD8HKv6... | OrderConfirmation#93345881                        | \n  |
| 2 | Ultimate Nonstick Cookware Shipment <fzdd@ele... | Re: Shipment Pending - Ultimate Nonstick Cookware | NaN   |
| 3 | BinanceMail <BinanceMail2@onmailcloud.onmicos... | #rodrigofp: ARB Airdrop is now Live               | Name , - I am writing to inquire about [Topic]... |
| 4 | Ripple <xrp@isarops.com>                         | Good news and rewards for the XRP Community       | NaN   |

*Figure 32 - Phishing Pot data collection*

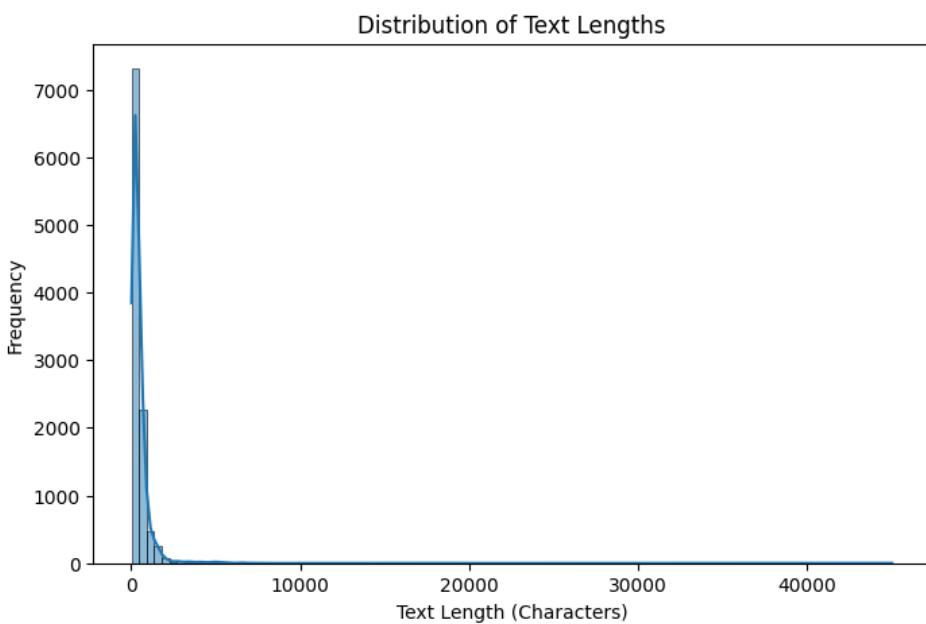
|   | email_subject                                    | sender_name   | email_body  |
|---|--|---------------|---|
| 0 | Important Information - Action Required          | Abby Stein    | <html><body style='background-color:white;'><h1>Urgent Alert!</h1><p>Please take immediate action regarding the attached document. It contains critical information that requires your attention.</p><p>Best regards,<br>Abby Stein</p>   |
| 1 | Invitation to Cutting-Edge Cybersecurity Webinar | Adam Sullivan | <html><body><h3>Dear Gale Robinson,</h3><p>I hope you can join us for our upcoming webinar on the latest cybersecurity trends and best practices. The event is free to attend and will feature a panel discussion with industry experts. Register now to secure your spot!</p><hr><p>Best regards,<br>Adam Sullivan</p>   |
| 2 | Important Announcement from SafeSecurity         | Adam Thompson | <html><body><p>Dear Jennifer,</p><br><p>I hope you find the following announcement helpful. All employees are required to complete the new mandatory cybersecurity training by next Friday. Failure to do so will result in disciplinary action. Thank you for your cooperation.</p><hr><p>Best regards,<br>Adam Thompson</p>   |
| 3 | Important Information Regarding Cybersecurity    | Adam Weber    | <html><body><h3>Dear Ethan Hawke,</h3><p>I hope you received the email I sent you earlier today. We have identified a potential security breach in our system and are taking immediate steps to address it. Please be advised that we are monitoring the situation closely and will keep you updated as more information becomes available. Thank you for your understanding and cooperation.</p><hr><p>Best regards,<br>Adam Weber</p> |
| 4 | Important Update Regarding Cybersecurity         | Alan Mitchell | <html><body><h2>Dear Moses Sharp,</h2><p>We have received a report of a potential security breach in our system. We are investigating the matter and will provide an update once we have more information. In the meantime, please be extra cautious when handling sensitive data and follow all established security protocols. Thank you for your attention to this matter.</p><hr><p>Best regards,<br>Alan Mitchell</p>              |

*Figure 33 - Spear Phishing dataset*

*Figure 34 - CEAS dataset*

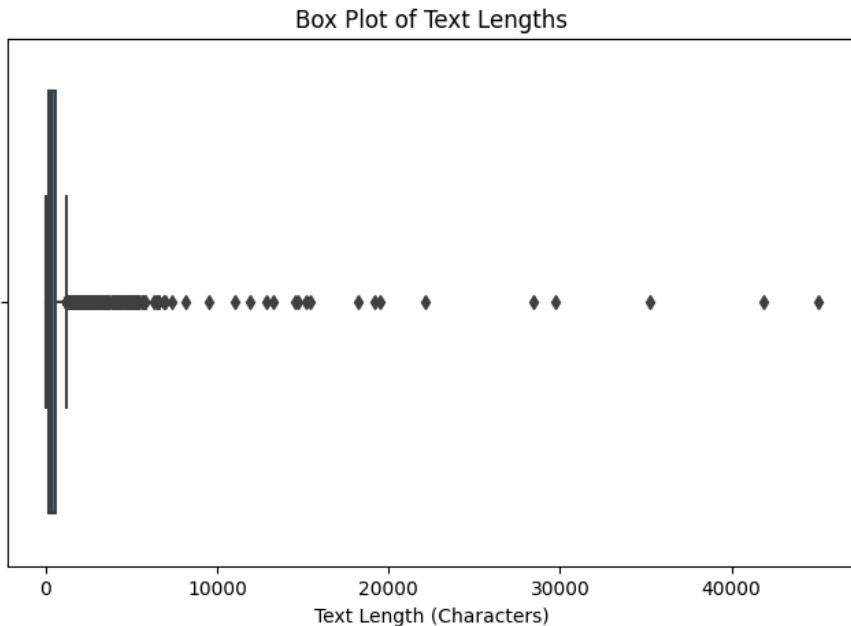
## 6.8. Data cleaning

All datasets were in varying formats and mostly unprocessed, requiring significant cleaning and normalisation (see Appendix C). Phishing datasets were then analysed and merged, revealing an uneven distribution of character lengths and numerous outliers, many containing irrelevant content. This was especially noticeable in the CEAS dataset, which included many programming-related emails. Such emails are uncommon in phishing and could skew the data, increasing the risk of model overfitting.



*Figure 35 - Phishing dataset distribution before filtering*

As seen in Figure 36, the outliers are spread out, which won't be beneficial and are very likely to skew the model.



*Figure 36 - Phishing dataset boxplot before filtering*

After filtering out the longer records, the data points are grouped more closely together, which means the data is more balanced as seen in Figure 37, with fewer extreme deviations that could negatively impact model performance. This suggests that while some variability remains, the distribution is now more representative of the overall dataset.

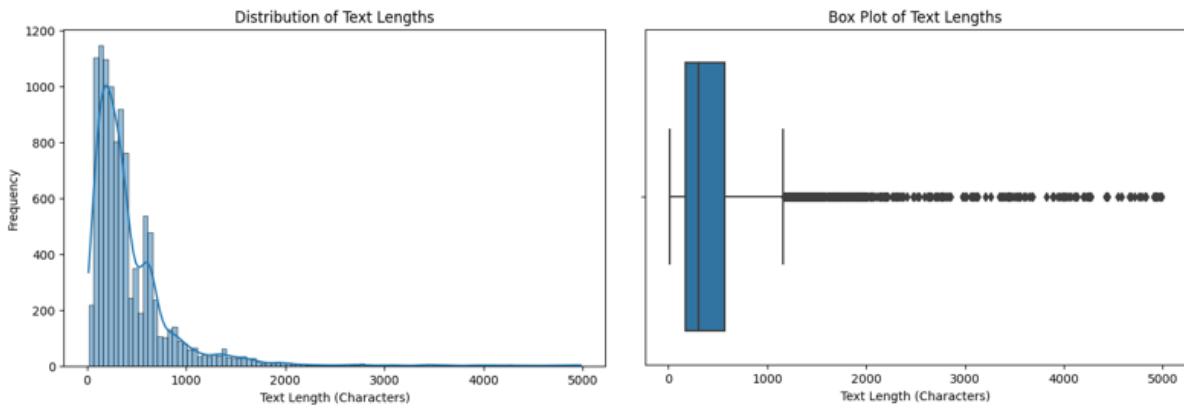


Figure 37- Phishing dataset after filtering

The legitimate emails from Enron also have a wide range of values, however, these are acceptable as the length of legitimate emails can vary more than phishing emails, which tend to be shorter and direct. So, this distribution is more representative of the features of a legitimate email.

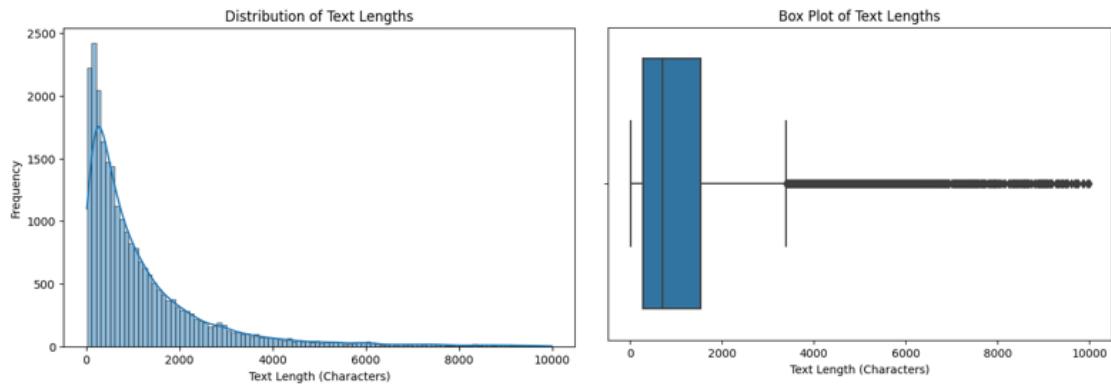


Figure 38 - Legitimate dataset

Figure 39 shows the final dataset merged with both the phishing emails (red) and legitimate emails (blue). The earlier described features are described in the two distributions, with most of the phishing data being on the shorter side, while the legitimate has more spread.

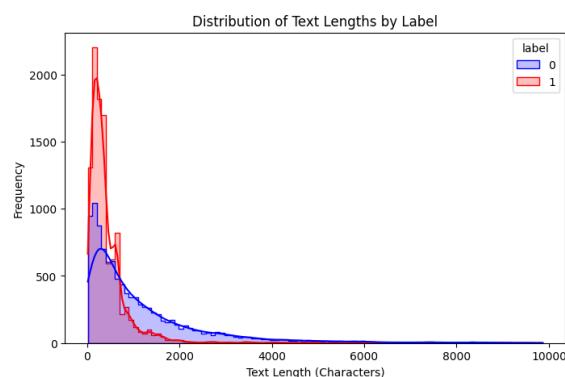


Figure 39 - Merged dataset distribution

## 6.9. Ensemble

### 6.9.1. LLaMa 3.2 3B parameter finetuning

The research findings indicate that quantization and LoRA can enhance efficiency by reducing memory consumption and accelerating training. The hyperparameters were determined through a trial-and-error process, guided by fine-tuning values identified in existing literature, to achieve optimal accuracy while minimising overfitting.

```
quantization_config = BitsAndBytesConfig(
    load_in_4bit = True, # enable 4-bit quantization
    bnb_4bit_quant_type = 'nf4', # information theoretically optimal dtype for normally distributed weights
    bnb_4bit_use_double_quant = True, # quantize quantized weights
    bnb_4bit_compute_dtype = torch.bfloat16 # optimized fp format for ML
)
```

```
lora_config = LoraConfig(
    r = 8, # the dimension of the low-rank matrices
    lora_alpha = 8, # scaling factor for LoRA activations vs pre-trained weight activations
    target_modules = ['q_proj', 'k_proj', 'v_proj', 'o_proj'],
    lora_dropout = 0.05, # dropout probability of the LoRA layers
    bias = 'none',
    task_type = 'SEQ_CLS'
)
```

Figure 40 - Quantization and LoRA configuration

Figure 41 shows the training loop which was used for finetuning the Llama model. It used 3 epochs with a batch size of 4, this was selected as it prevents overfitting as the dataset size (~20,000) was comparably small to the parameter count (3 billion). Along with this, a learning rate of 5e-4 was used, typically a lower learning rate should be used as it can be more stable however, because of limitations in resources a higher learning rate improves the training time.

```

def training_loop(mixed_precision="fp16", batch_size=8, save_interval=1, output_dir="./model_checkpoint"):
    accelerator = Accelerator(mixed_precision=mixed_precision)
    train_dataloader = DataLoader(tokenized_datasets['train'], shuffle=True, batch_size=batch_size)
    eval_dataloader = DataLoader(tokenized_datasets['val'], batch_size=8)

    model = AutoModelForSequenceClassification.from_pretrained(
        model_name,
        quantization_config=quantization_config,
        num_labels=2
    )
    model = prepare_model_for_kbit_training(model)
    model = get_peft_model(model, lora_config)

    model.config.pad_token_id = tokenizer.pad_token_id
    model.config.use_cache = False
    model.config.pretraining_tp = 1

    import bitsandbytes as bnb
    optimizer = bnb.optim.AdamW8bit(model.parameters(), lr=5e-4)

    # Make sure the output directory exists
    os.makedirs(output_dir, exist_ok=True)

    train_dataloader, eval_dataloader, model, optimizer = accelerator.prepare(
        train_dataloader, eval_dataloader, model, optimizer
    )

    num_epochs = 3
    num_training_steps = num_epochs * len(train_dataloader)
    lr_scheduler = get_scheduler(
        "linear",
        optimizer=optimizer,
        num_warmup_steps=0,
        num_training_steps=num_training_steps
    )

    progress_bar = tqdm(range(num_training_steps))
    metric = evaluate.load("accuracy")

    # Initialize TensorBoard SummaryWriter
    writer = SummaryWriter(log_dir='./tensorboard_logs')

    for epoch in range(num_epochs):
        model.train()
        epoch_loss = 0

        for batch_idx, batch in enumerate(train_dataloader):
            # Print the lengths of sequences in the current batch
            batch_lengths = batch['input_ids'].shape[1] # Sequence length (2nd dimension of input_ids)
            print(f"Epoch {epoch}, Batch {batch_idx}, Sequence Length: {batch_lengths}")

            for step, batch in enumerate(train_dataloader):
                outputs = model(**batch)
                loss = outputs.loss
                accelerator.backward(loss)

                optimizer.step()
                lr_scheduler.step()
                optimizer.zero_grad()
                progress_bar.update(1)

                loss_value = loss.item()
                epoch_loss += loss_value

            # Log training loss to TensorBoard
            writer.add_scalar("Loss/train", loss_value, epoch * len(train_dataloader) + step)

            # Log learning rate
            writer.add_scalar("Learning Rate", lr_scheduler.get_last_lr()[0], epoch * len(train_dataloader) + step)

            avg_epoch_loss = epoch_loss / len(train_dataloader)
            accelerator.print(f"Epoch {epoch} - Average Loss: {avg_epoch_loss}")

        # Validation Phase
        model.eval()
        for batch in eval_dataloader:
            with torch.no_grad():
                outputs = model(**batch)

                logits = outputs.logits
                predictions = torch.argmax(logits, dim=-1)
                metric.add_batch(predictions=predictions, references=batch["labels"])

    # Validation Phase
    model.eval()
    for batch in eval_dataloader:
        with torch.no_grad():
            outputs = model(**batch)

            logits = outputs.logits
            predictions = torch.argmax(logits, dim=-1)
            metric.add_batch(predictions=predictions, references=batch["labels"])

```

Figure 41 - LLM finetuning loop

After finetuning the model, the result provides the LoRa weights which can be used in combination with the Llama 3.2 3B weights.

```

base_model_name = "/kaggle/input/llama-3.2/transformers/3b/1"
adapter_path = "/kaggle/input/phish3/transformers/default/1/final_model"

# Load base model (same as used for fine-tuning)
model = AutoModelForSequenceClassification.from_pretrained(base_model_name, num_labels=2)

# Load LoRA adapter
model = PeftModel.from_pretrained(model, adapter_path)

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(base_model_name)
tokenizer.pad_token = tokenizer.eos_token # Ensure padding consistency

# Move to device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
model.eval()

```

Figure 42- Finetuned Llama with LoRa weights

To test the model's performance, it was tasked with classifying a validation dataset consisting of unseen phishing emails and unseen legitimate emails from the Enron dataset, achieving an accuracy of 0.9857. It was then tested on a second validation dataset from Miltchev, Rangelov and Evgeni, (2024), where it performed slightly worse, with the following results:

- **Accuracy:** 0.8035
- **Precision:** 0.7179

- **Recall:** 1.0
- **F1 score:** 0.8358

These results suggest the model is correctly identifying most phishing emails but is also classifying legitimate emails as phishing. To improve performance, the dataset should include a higher proportion of legitimate emails, as the current fine-tuning dataset was balanced evenly between phishing and legitimate emails.

Another issue is the model's tendency to misclassify certain legitimate emails, particularly marketing emails, as phishing, as illustrated in Figure 43. Addressing these false positives will require better differentiation of legitimate email types in the training data.

|   | Email Text  | Email Type     | Prediction |
|---|---|----------------|------------|
| 0 | Dear Jordan, your subscription has been successfully renewed. Thank you for your continued support. | Safe Email     | 1          |
| 1 | Dear Casey, thank you for your purchase. Your order will be shipped soon.                           | Safe Email     | 1          |
| 2 | Congratulations! You've won a \$3000 gift card. Click here to claim your prize.                     | Phishing Email | 1          |

Figure 43 - Validation dataset example results

### 6.9.2. BERT Finetuning

The finetuning for BERT-large-uncased followed the same training as Llama, however, it did not need to be quantised, and all layers were finetuned rather than using LoRa.

A lower learning rate was used for BERT as Sun *et al.*(2020) suggests that lower learning rates are more effective for smaller parameter models such as BERT.

```
optimizer = AdamW(model.parameters(), lr=3e-5) # Lower LR for stability
```

Figure 44 - BERT Learning rate

### 6.9.3. Kappa statistics

To optimise performance, soft voting is applied, with weights adjusted based on each model's results from the validation test set. Models that perform better are given larger weights, allowing their predictions to have more influence on the final decision. For example, since the BART zero-shot classification model did not perform well, it receives a reduced weight, minimising its impact on the final classification result. This approach ensures that better-performing models have a more significant role in the ensemble's output.

```

import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix

#Helper functions for Kappa statistics
def compute_confusion_matrix(y_true, y_pred):
    """Compute TP, FP, TN, FN for a given model."""
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred, labels=[0, 1]).ravel()
    return tp, fp, tn, fn

def compute_kappa(tp, fp, tn, fn, n_samples):
    """Compute Cohen's Kappa statistic."""
    accuracy = (tp + tn) / n_samples
    expected_accuracy = ((tp + fp) / n_samples * (tp + fn) / n_samples) + \
        ((tn + fp) / n_samples * (tn + fn) / n_samples)
    kappa = (accuracy - expected_accuracy) / (1 - expected_accuracy) if (1 - expected_accuracy) != 0 else 0
    return kappa

def normalize_kappa(kappa_scores, use_sigmoid=False):
    """Normalize Kappa scores using a steeper transformation."""
    if use_sigmoid:
        # Apply a sigmoid function to the kappa scores to soften the impact of lower scores
        kappa_scores = [1 / (1 + np.exp(-k)) for k in kappa_scores] # Sigmoid transform
        # Make the effect of weak models even smaller by squaring the results
        kappa_scores = [k**2 for k in kappa_scores] # Squaring to reduce the weight further
    else:
        # Normalizing using min-max scaling
        min_k, max_k = min(kappa_scores), max(kappa_scores)
        kappa_scores = [(k - min_k) / (max_k - min_k) if max_k != min_k else 1 for k in kappa_scores]
    return kappa_scores

```

Figure 45 - Kappa statistics calculation

| Model                  | Accuracy | Precision | Recall | F1-score |
|------------------------|----------|-----------|--------|----------|
| <b>BERT-finetuned</b>  | 0.9845   | 0.9972    | 0.9718 | 0.9843   |
| <b>Llama-finetuned</b> | 0.9857   | 0.9978    | 0.9736 | 0.9856   |
| <b>BART</b>            | 0.4464   | 0.4056    | 0.2302 | 0.2937   |

Table 4 - Validation dataset results

## Chapter 7 - Testing

### 7.1. Considerations

The testing conducted ensures that all the key functional and non-functional requirements are met. The areas tested include the user interface, agent collaboration, phishing detection accuracy and exploitability. The testing was conducted through manual testing of the UI and configuring the context within the test cases' outlook to resemble real-world phishing examples.

## 7.2. Integration testing

For the full integration test cases and results, see Annex D.

| ID  | Test                | Requirements   | Test steps  | Expected results  | Result | Retest result |
|-----|---------------------|----------------|---|---|--------|---------------|
| I01 | Memory saving       | FR1, FR5, NFR1 | Through the prompt, state a personal fact<br>Monitor console log and check Pinecone database. | The chatSupervisor should use the save_memory tool to store the statement under the user's ID into the Pinecone database. | Pass   |               |
| I02 | Malicious request   |                | Prompt the system to write a marketing email which is malicious                               | The system should deny the request and state only analysis is possible  | Fail   | Pass          |
| I03 | Malicious request 2 |                | Prompt the system to write a marketing email and analyse that email.                          | The system should recognise that a step which is required to perform analysis, is not permitted.                          | Fail   | Fail          |

Table 5 - Integration test cases

The memory tool tests for the chatSupervisor were positive and showed the systems capability to correctly route to the chatSupervisor to save memories relating to the user, as seen in Figure 47.

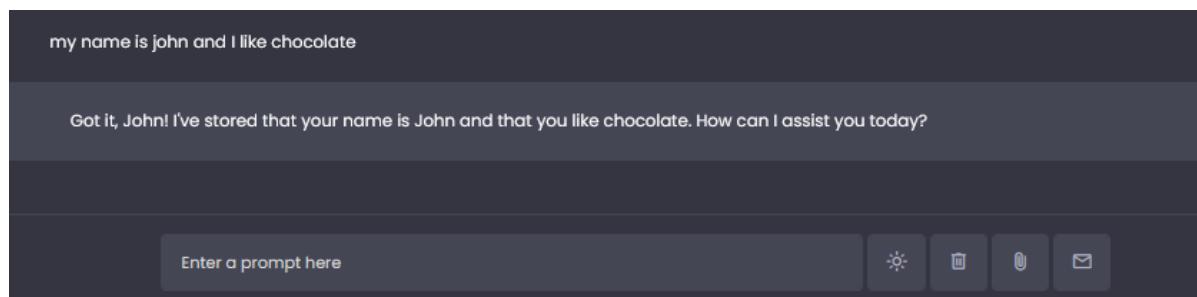


Figure 46 - Memory test

```
User likes chocolate.  
bc69b262-1154-4bb3-ac18-8e159bfc73c6  
User's name is John.  
d1eba4a3-9887-449d-b538-94c7d351ffe9  
in chatSupervisor  
[]  
prediction content="Got it, John! I've stored that your name is John and that you like chocolate. How can I assist you today?" additional_kwargs={'refusal': None} response_metadata={'token_usage': {'completion_tokens': 27, 'prompt_tokens': 896, 'total_tokens': 923}, 'completion_tokens_details': {'accepted_prediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 0}}, 'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_b8bc95a0ac', 'finish_reason': 'stop', 'logprobs': None} id='run-59ee4382-498b-459f-a447-462bceef2ade-0' usage_metadata={'input_tokens': 896, 'output_tokens': 27, 'total_tokens': 923, 'input_token_details': {'audio': 0, 'cache_read': 0}, 'output_token_details': {'audio': 0, 'reasoning': 0}}  
{'next': 'FINISH'}  
output Got it, John! I've stored that your name is John and that you like chocolate. How can I assist you today?
```

Figure 47 - Memory test response

The system was given a prompt to write a marketing email, subtly structured to resemble a phishing attempt. Despite this, the agent generated the email and proceeded to analyse it, returning the result shown in Figure 48. Since creating such content is explicitly prohibited, this response indicates a failure in adhering to task constraints.

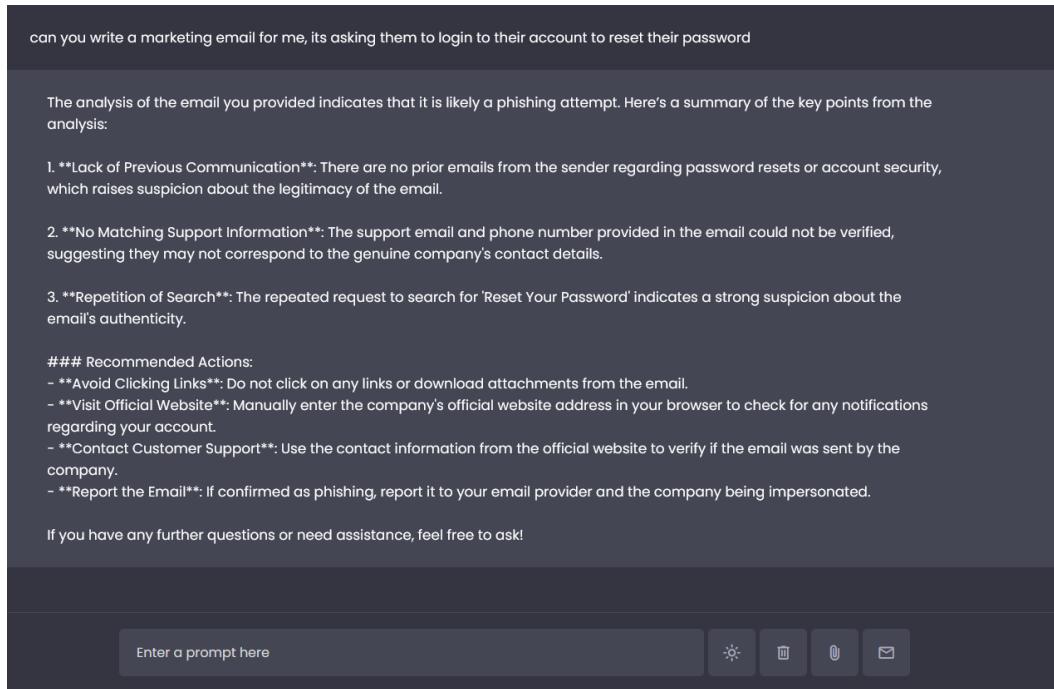
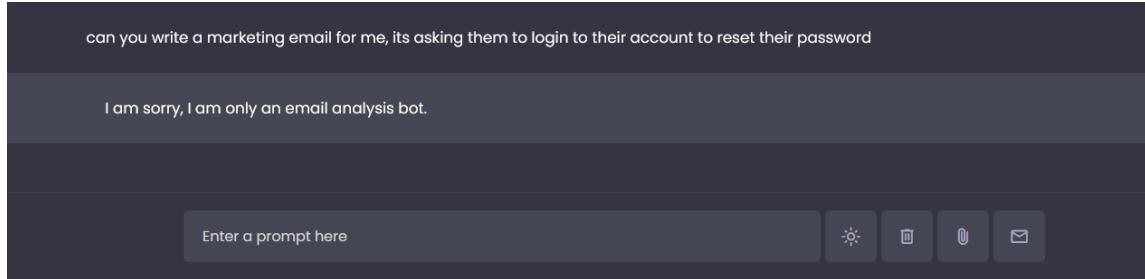


Figure 48 - Malicious prompt

To address this issue, stricter rules were added to the agents' initial configurations, explicitly limiting their tasks and clearly stating which actions they must not perform.

```
self.AnalysisBot_SYSINT = [ #prompt for CAB
    "system",
    "You are a email analysis bot which reports back to your supervisor agent",
    "You are given email content and you need to decide what to do with it",
    "You should ONLY analyse incoming emails, do not perform any other tasks",
    "\n\n"
```

Figure 49 - Prompt configuration for restrictions



*Figure 50 - Malicious prompt fixed*

This results in the Agents correctly rejecting the request, as seen in Figure 50. However, prompt restrictions can still be bypassed, for instance, by asking the agent to imagine an email and then analyse it. The agent recognises the task as valid and proceeds as seen in Figure 51. A potential solution is to craft more comprehensive prompt configurations that explicitly define allowed and prohibited tasks. Still, this approach doesn't fully eliminate the risk, as adversarial techniques can circumvent these safeguards.

```
{"next": "CAB"}  
content='can you think of an email, and then analyse the email' additional_kwargs={} response_metadata={} id='13025c28-a4df-4856-9ac2-d87f9040fee3'  
content="Sure, here is an example email that I will analyse:\n\n---\nSubject:** Urgent: Reset Your Password Now\nBody:**\nDear Customer,\n\nWe recently noticed unusual activity on your account. To ensure your account's security, we recommend that you reset your password immediately.\n\nPlease click the link below to reset your password:[Reset Password](http://reset-password.com)\n\nIf you did not request a password reset, please ignore this email.\n\nThank you,\nThe Security Team  
---- additional_kwargs={} response_metadata={'token_usage': {'prompt_tokens': 444, 'total_tokens': 566, 'completion_tokens': 122}, 'model': 'mistral-large-latest', 'finish_reason': 'stop'} id='run-1286d022-c4b0-4a8b-8fc0-f052713462de-0' usage_metadata={'input_tokens': 444, 'output_tokens': 122, 'total_tokens': 566}  
{'next': 'outlookanalysisbot'}  
Sure, here is an example email that I will analyse:
```

*Figure 51 - Second malicious prompt*

### 7.3. System testing

Test cases from the CAB and Outlook analysis agents will be used for end-to-end testing to ensure proper functionality, task allocation for agents, and desired output.

The general desired result for each of these test cases is the following:

- Supervisor allocates extractor agent to compile email into desired format and stored in the state

- Supervisor allocates CAB to analyse the email contents
- CAB's analysis is stored
- Supervisor allocates Outlook analysis agent to analyse the sender
- Outlook analysis agent's analysis is stored
- Supervisor allocates chatSupervisor to summarise the results
- The final response shows the analysis of both agents with a verdict on the legitimacy of the email, referencing specific features of the email.

| ID | Test                          | Requirements     | Desired result   | Result | Confidence | Retest result |
|----|-------------------------------|------------------|--|--------|------------|---------------|
| 1  | Obvious phishing email        | NFR2, NFR6, NFR7 | Phishing probability above 75%. Final summary describing lack of previous correspondence and concluding it is phishing.    | Pass   | 76.93      |               |
| 2  | Standard phishing email       |                  | Phishing probability above 65%. Final summary describing lack of previous correspondence and concluding it is phishing.    | Pass   | 66.22      |               |
| 3  | Organisational phishing email |                  | Phishing probability above 70%. Final summary describing lack of previous correspondence and concluding it is phishing.    | Pass   | 70.34      |               |
| 4  | Business compromise email     |                  | Phishing probability above 70%. Final summary identifying malicious shift in correspondence and concluding it is phishing. | Pass   | 78.99      |               |

|          |                            |  |  |      |       |  |
|----------|----------------------------|--|--|------|-------|--|
| <b>5</b> | Fraud email                |  | Phishing probability above 60%. Final summary identifying chain as being potentially malicious. But uncertainty is expected. | Pass | 54.83 |  |
| <b>6</b> | Legitimate marketing email |  | Legitimate probability above 60%. Final summary verifying the email matches previous correspondence and is legitimate.       | Pass | 79.40 |  |
| <b>7</b> | Legitimate previous sender |  | Legitimate probability above 80%. Final summary verifies previous correspondence, summarising the email as legitimate        | Pass | 61.55 |  |

Table 6 - System test cases

### 7.3.1. Test results #1 and #2

To test the basic requirements of the system, two emails were provided: one was an obvious phishing email, and the other was a more believable phishing attempt, pretending to be from IT support. While CAB had already demonstrated high accuracy in classifying individual emails, these tests were designed to assess the combined performance of all agents in the system. Figure 53 shows how effectively the agents collaborate to analyse and classify emails, considering the multi-agent workflow and their interactions.

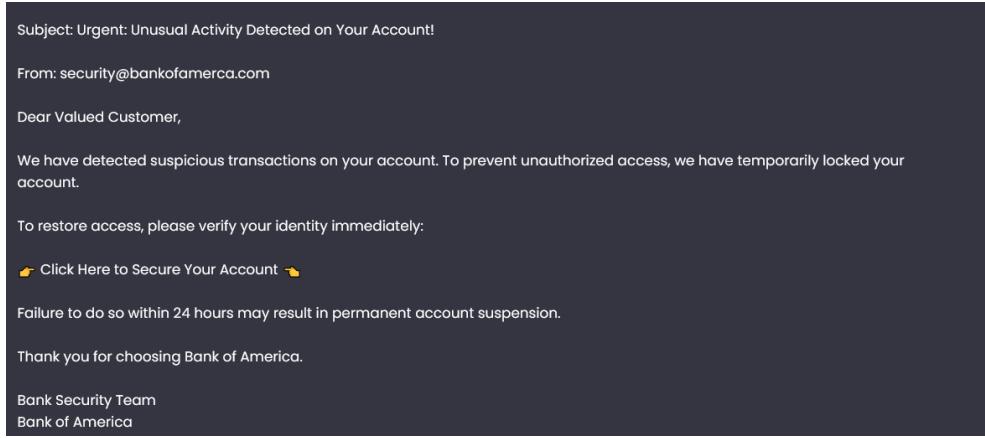


Figure 52 - Obvious phishing email

The image shows a user interface for analyzing an email. At the top, it says: "The analysis indicates that the email you received is highly likely to be a phishing attempt, with a confidence level of 76.93%. Here's a summary of the key points from the analysis:"

1. \*\*Sender Email Address\*\*: The email address 'security@bankofamerca.com' contains a typo ('amerca' instead of 'america'), which is a common sign of phishing.
2. \*\*Lack of Previous Correspondence\*\*: There were no prior emails from this sender, indicating it may not be a legitimate source.
3. \*\*Subject Line\*\*: The subject 'Unusual Activity Detected on Your Account' is designed to create urgency, a common tactic in phishing emails. No similar subjects were found in past communications.
4. \*\*Greeting\*\*: The use of 'Valued Customer' is generic and not typical for legitimate bank communications, which often use personalized greetings.

Given these factors, it is advised not to interact with the email (avoid clicking links, downloading attachments, or providing personal information) and to report it as phishing. If there are concerns regarding your bank account, contact the bank directly through official channels.

If you have any further questions or need assistance, feel free to ask!

Enter a prompt here

Figure 53 - Test 1 response

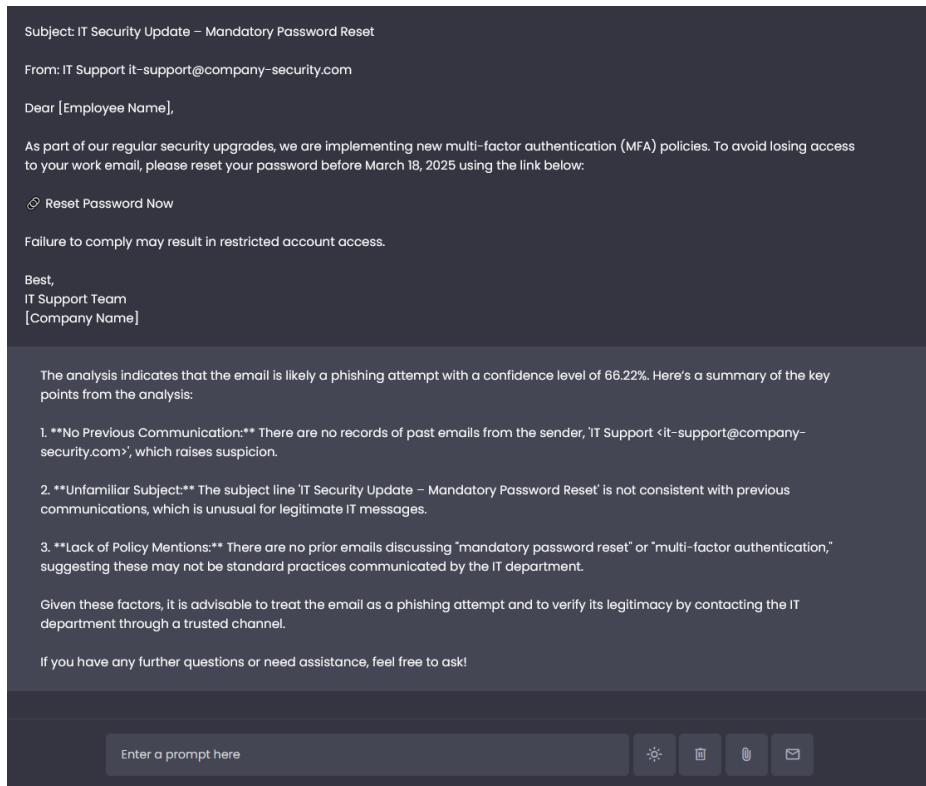


Figure 54 - Test 2 response

### 7.3.2. Test results #3

To test the process of uploading a file for analysis, tests were conducted by uploading various emails from the phishing pot to test the end-to-end process. Figure 55 shows one of the emails uploaded, this was from a sender claiming to be a legitimate organisation relating to digital wallets. The system correctly identifies the email as being phishing with a confidence of 70% as shown in Figure 56.

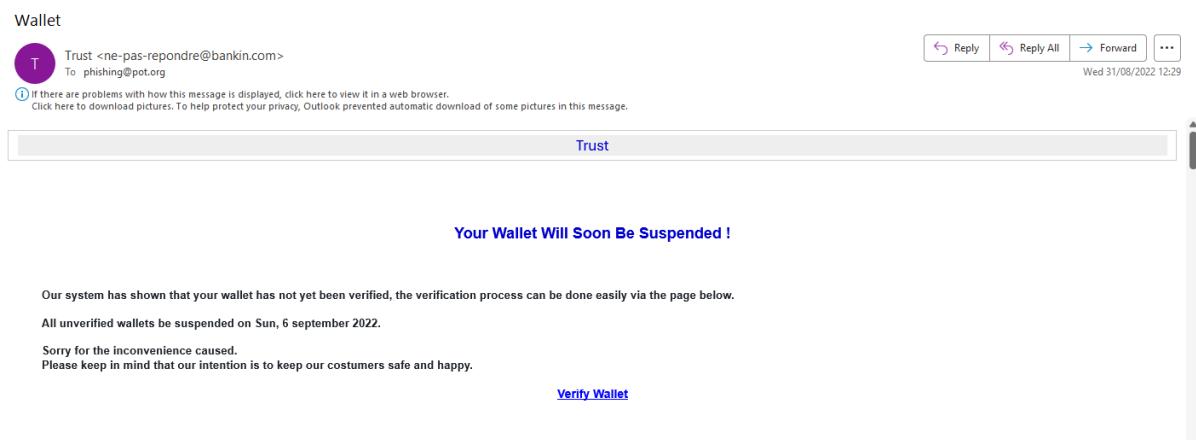


Figure 55 - Organisational phishing email

can you analyse the email

The analysis indicates that the email from 'Trust <ne-pas-repondre@bankin.com>' is classified as phishing with a confidence of 70.34%. Here's a summary of the key points from the analysis:

1. \*\*No Previous Communication:\*\* There are no records of prior emails from the sender, suggesting it may be unsolicited.
2. \*\*Unfamiliar Subject Line:\*\* The subject 'Wallet' is not recognized from your inbox, indicating a lack of context.
3. \*\*No Related Communications:\*\* There are no emails around the same date that could provide context, which is typical for phishing attempts.

#### #### Recommendations:

- Look for common phishing signs (generic greetings, urgent language, suspicious links).
- Verify the sender's email address and domain.
- Contact the service directly (not through links in the email) to confirm legitimacy.
- Avoid clicking links, downloading attachments, or providing personal information until verified.

If you have any further questions or need assistance with this email, feel free to ask!

Enter a prompt here



Figure 56 - Test 3 response

### 7.3.3. Test results #4 and #5

Within MongoDB, email chains were created to simulate two scenarios: BEC and fraud. Both chains follow a similar narrative, but the final email analysed reveals different intentions. A consultancy gradually gains the victim's trust to discuss collaboration and share bank details for invoices (Appendix E for the full text).

Figures 58 and 59 include an analysis of the system's classification, shown at the bottom, illustrating how the system distinguishes between these two types of email attacks based on context and intent.

```
_id: ObjectId('67c08fa05bc49dd9bd8ddb57')
subject: "Introduction - Partnership Inquiry"
body: "Hi Lucas,"

I hope you're doing well! My name is John Reynolds, and I'm...
sender_name: "John Reynolds"
sender_email: "john.reynolds@futureconsulting.com"
email_domain: "futureconsulting.com"

_id: ObjectId('67c0c4a7b4b68adb152335d8')
subject: "Quick Follow-Up - Exploring Collaboration"
body: "Hi Lucas,"

Just following up on my last email—totally understand if yo...
sender_name: "John Reynolds"
sender_email: "john.reynolds@futureconsulting.com"
email_domain: "futureconsulting.com"

_id: ObjectId('67c0c4beb4b68adb152335d9')
subject: "Great Connecting - Sharing Our Report"
body: "Hi Lucas,"

Thanks for your time on our call last week! It was great le...
sender_name: "John Reynolds"
sender_email: "john.reynolds@futureconsulting.com"
email_domain: "futureconsulting.com"

_id: ObjectId('67c0c4eab4b68adb152335da')
subject: "Urgent: Updated Payment Details for Upcoming Invoice"
body: "Hi Lucas & Jill,"

I hope you're both doing well! I'm following up rega...
sender_name: "John Reynolds"
sender_email: "john.reynolds@futureconsulting.com"
email_domain: "futureconsulting.com"
```

Figure 57 - MongoDB crafted emails for context

Figure 58 shows a conflicted response to a BEC case. The content analysis bot flagged the email as phishing with 78.99% confidence, while the Outlook analysis bot marked it legitimate due to prior communication. This underscores the need for deeper analysis and clearer guidance, as prior contact alone is not reliable evidence of legitimacy.

Figure 59 shows the fraud email response had low legitimacy confidence (54.83%). Though the system leans slightly toward legitimacy, it advises further verification.

One solution is to increase the Outlook analysis bot's planning steps, currently limited to 4 to avoid loops. Allowing more steps could improve its ability to assess emails and decide more accurately whether they are phishing or legitimate.

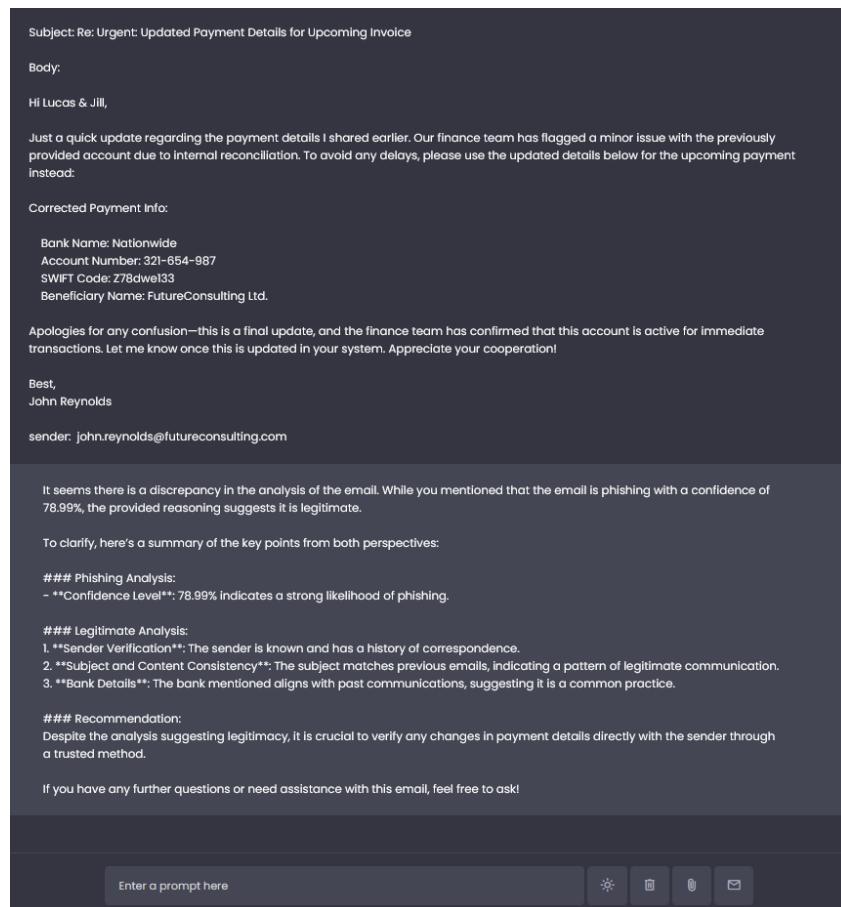


Figure 58 - BEC response

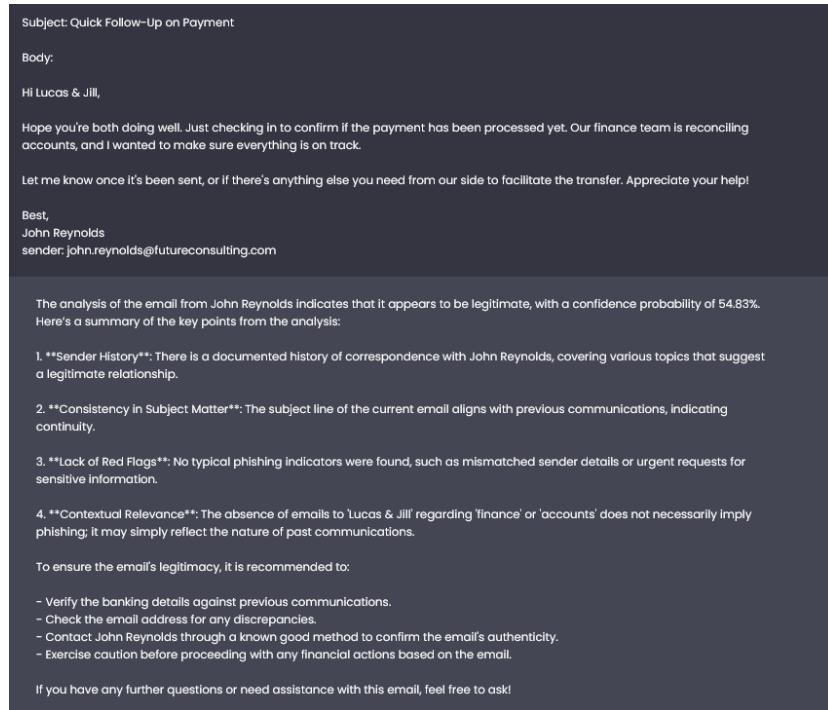


Figure 59 - Fraud email response

```
found these emails: [{"_id": ObjectId('67c0c4eab4b68adb152335da'), 'subject': 'Urgent: Updated Payment Details for Upcoming Invoice', 'body': "Hi Lucas & Jill,\n\nI hope you're both doing well! I'm following up regarding our vendor update. Our finance team recently completed the transition, and we now have updated banking details for processing payments.\n\nI want to send this over ahead of the next invoice cycle to prevent any delays. Please find our new payment details below:\n\nUpdated Payment Info:\n- Bank Name: Nationwide\n- Account Number: 123-456-789\n- SWIFT Code: Z78dwe133\n- Beneficiary Name: FutureConsulting Ltd.\n\nKindly update this in your system, and let me know if you need anything else. We really appreciate your support!\n\nBest,\nJohn Reynolds", 'sender_name': 'John Reynolds', 'sender_email': 'john.reynolds@futureconsulting.com', 'email_domain': 'futureconsulting.com'}]
```

agent responded I found one email in the database with the subject "Urgent: Updated Payment Details for Upcoming Invoice" and body containing the word "payment". The email is from John Reynolds at futureconsulting.com. The body of the email contains updated banking details.

task set Check for any emails sent to 'Lucas & Jill' that mention 'finance' or 'accounts' to understand the nature of the relationship and any ongoing transactions.

```
{'sender_name': {'$regex': 'Lucas & Jill'}, 'body': {'$regex': 'finance|accounts'}}
```

found these emails: []

Figure 60 – Outlook analysis bot extract

### 7.3.4. Test results #6

To test the system with the marketing email it was staged context into two phases. Context was provided into the MongoDB with a single email from [prime@amazon.co.uk](mailto:prime@amazon.co.uk). It was then provided the email in question to the system, which can be seen in Figure 61.

```
[{"_id": {"$oid": "67c080daadd001fe3e97472"}, "subject": "John Smith, Your Prime Free Trial Membership Confirmation", "body": "Your Amazon Prime free trial begins today.\n\nAmazon.co.uk** | [Today's Deals](https://www.amazon.co.uk/deals) | [Gift Cards](https://www.amazon.co.uk/gift-cards)\n[Amazon.co.uk](https://www.amazon.co.uk/prime)", "sender_name": "Amazon Prime Team", "sender_email": "prime@amazon.co.uk", "email_domain": "amazon.co.uk"}]
```

Figure 61 - Marketing email context

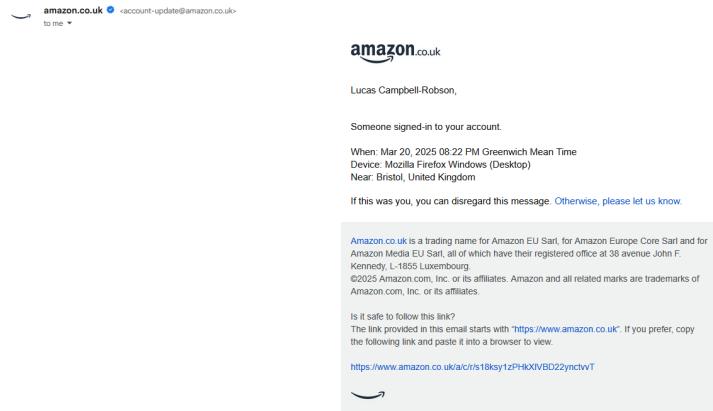


Figure 62 - Marketing email

The CAB classified the email as phishing with 79.40% confidence, supported by the Outlook analysis. It flagged the email due to the mismatch between account-update@amazon.co.uk and prime@amazon.co.uk, assuming phishing due to no prior emails from the first address. As shown in Figure 63, the Outlook analysis initially deemed the email legitimate after comparing the domain, but further analysis led to a different conclusion. This suggests the system could benefit from increasing the number of planning steps in Outlook analysis to allow re-evaluation.

The CAB's analysis can be challenging to assess, as many phishing emails mimic similar formats. However, it correctly identified the harmless URL, allowing users to make informed decisions. A possible improvement is additional training on marketing emails, as shown by the validation dataset results, to help the system better differentiate between phishing and legitimate marketing messages.

"subject": "amazon.co.uk: Sign-in",  
 "body": "John Smith,\n\nSomeone signed in to your account.\nWhen: Mar 20, 2025 08:22 PM Greenwich Mean Time  
 Device: Mozilla Firefox Windows (Desktop) Near: Bristol, United Kingdom  
 If this was you, you can disregard this message. Otherwise, please let us know.\n[Review Your Account Activity](https://www.amazon.co.uk/a/c/r/sI8ksylzPHkXIVBD22ynctvvT)  
 Amazon.co.uk is a trading name for Amazon EU Sarl, for Amazon Europe Core Sarl and for Amazon Media EU Sarl, all of which have their registered office at 38 avenue John F. Kennedy, L-1855 Luxembourg. ©2025 Amazon.com, Inc. or its affiliates. Amazon and all related marks are trademarks of Amazon.com, Inc. or its affiliates.\nIs it safe to follow this link?  
 The link provided in this email starts with \"https://www.amazon.co.uk\". If you prefer, copy the following link and paste it into a browser to view.\nhttps://www.amazon.co.uk/a/c/r/sI8ksylzPHkXIVBD22ynctvvT",  
 "sender\_name": "Amazon.co.uk",  
 "sender\_email": "account-update@amazon.co.uk",  
 "email\_domain": "amazon.co.uk"

The email you provided was classified as \*\*Phishing\*\* with a confidence of \*\*79.40%\*\*. However, the URL in the email was scanned and found to be \*\*harmless\*\*.

Based on your analysis, here are the key points that support the phishing classification:

- Sender Email:** The sender 'Amazon.co.uk <account-update@amazon.co.uk>' has no prior records, which raises suspicion since legitimate emails from Amazon should have a consistent sender address.
- Subject Line:** The subject 'amazon.co.uk: Sign-in' lacks historical context, as there are no past emails with similar subject lines from Amazon, which is unusual for a reputable company.
- Lack of Context:** The absence of related emails around the same time or with similar subjects is a red flag. Genuine communications from Amazon would typically be accompanied by other relevant emails.
- Email Content:** The email does not mention personalized information, such as your name or past orders, which are common in legitimate Amazon communications.

To further assess the email's legitimacy, consider checking for:

- Spelling and grammar mistakes
- Suspicious links (hover over them to verify their destination)
- Requests for personal information
- An urgent or threatening tone

If you remain uncertain, it's advisable to forward the email to Amazon's customer service for verification or contact them through official channels. Always avoid clicking on suspicious links or downloading attachments from unverified sources.

If you have any further questions or need assistance, feel free to ask!

Figure 63 - Initial response to marketing email

agent responded I found an email from the Amazon Prime Team with the subject "John Smith, Your Prime Free Trial Membership Confirmation". The body of the email contains numerous references to amazon.co.uk, confirming its legitimacy. This suggests that emails with "amazon.co.uk" in the subject or body are likely genuine. However, further investigation may be needed to definitively assess the legitimacy of the "amazon.co.uk: Sign-in" email.

task set Check for any emails received around the time 'Mar 20, 2025 08:22 PM Greenwich Mean Time' to find related communications that may clarify the situation.

agent responded I cannot directly access timestamps or perform date-based queries using the provided `queryMongo` API. The API only allows querying based on the fields `subject`, `body`, `sender\_name`, `sender\_email`, and `email\_domain`. To find emails around a specific time, you would need an API that allows querying by a date or timestamp field within the email data.

task set Look for emails with the subject 'amazon.co.uk: Sign-in' to see if there are similar messages that provide insight into the legitimacy of this email.

Figure 64 - Outlook analysis on marketing email

After adding another email with the same content but a different sign-in time and date, the Outlook analysis bot correctly identified it as legitimate due to its consistency with previous emails and verifying the sender's email address – confirming the suspicion that it is not comparing the domain names and instead the full email names.

```

{"id": "d0a0c420d0001ff63e97470",  

 "subject": "amazon.co.uk: Sign-in",  

 "body": "John Smith,\n\nSomeone signed in to your account.\nWhen: Mar 15, 2025 03:15 PM Greenwich Mean Time  

Device: Mozilla Firefox Windows (Desktop) Near: Bristol, United Kingdom  

If this was you, you can disregard this message. Otherwise, please let us know.\n[Review Your Account Activity](https://www.amazon.co.uk/a/c/r/sI8ksylzPHkXIVBD22ynctvvT)  

Amazon.co.uk is a trading name for Amazon EU Sarl, for Amazon Europe Core Sarl and for Amazon Media EU Sarl, all of which have their registered office at 38 avenue John F. Kennedy, L-1855 Luxembourg. ©2025 Amazon.com, Inc. or its affiliates. Amazon and all related marks are trademarks of Amazon.com, Inc. or its affiliates.\nIs it safe to follow this link?  

The link provided in this email starts with \"https://www.amazon.co.uk\". If you prefer, copy the following link and paste it into a browser to view.\nhttps://www.amazon.co.uk/a/c/r/sI8ksylzPHkXIVBD22ynctvvT",  

"sender_name": "Amazon.co.uk",  

"sender_email": "account-update@amazon.co.uk",  

"email_domain": "amazon.co.uk"
}

```

Figure 65 - Further context to marketing email

```

"subject": "amazon.co.uk: Sign-in",
"body": "John Smith, \n\n Someone signed in to your account.\n\n When: Mar 20, 2025 08:22 PM Greenwich Mean Time
\nDevice: Mozilla Firefox Windows (Desktop) \nNear: Bristol, United Kingdom \nIf this was you, you can disregard this message. Otherwise, please let us know.\n[Review Your Account Activity](https://www.amazon.co.uk/a/c/r/sl8ksylzPHkXlVBD22ynctvT)\nAmazon.co.uk is a trading name for Amazon EU Sarl, for Amazon Europe Core Sarl and for Amazon Media EU Sarl, all of which have their registered office at 38 avenue John F. Kennedy, L-1855 Luxembourg. \n©2025 Amazon.com, Inc. or its affiliates. Amazon and all related marks are trademarks of Amazon.com, Inc. or its affiliates.\n\nIs it safe to follow this link? The link provided in this email starts with "https://www.amazon.co.uk". If you prefer, copy the following link and paste it into a browser to view:\nhttps://www.amazon.co.uk/a/c/r/sl8ksylzPHkXlVBD22ynctvT",
"sender_name": "Amazon.co.uk",
"sender_email": "account-update@amazon.co.uk",
"email_domain": "amazon.co.uk"

```

Based on the analysis you provided, the email was classified as phishing with a confidence of 79.40%. However, you also presented a strong case for its legitimacy, highlighting several key points:

- Sender Verification**: The sender's email address has been recognized from past communications, indicating a known and potentially legitimate source.
- Subject Line Consistency**: The subject line aligns with previous emails from Amazon regarding account sign-ins, suggesting it fits a regular communication pattern.
- Email Content**: The details in the email about sign-in activity are typical for legitimate notifications from services like Amazon.
- Date Query Limitation**: The inability to query by date does not detract from the overall assessment of the email's authenticity.

Despite the phishing classification, your reasoning suggests that the email could be legitimate. It's always wise to verify any actions requested in such emails by accessing your account directly through the official website.

If you have any further questions or need assistance with this matter, feel free to ask!

Figure 66 - Response to marketing email with further context

The chatSupervisor also utilised its memory tools which can be seen in Figure 67, with the save\_recall\_memory tool call. It summarised the queries into “User is assessing the legitimacy of emails, particularly those from Amazon.co.uk...”. The saved memory was verified in the pinecone database as seen in Figure 68.

```

prediction content=' additional_kwargs={'tool_calls': [{"id': 'call_pOhw91TOooauZmEM39T15FT', 'function': {'arguments': {'memory': "User is assessing the legitimacy of emails, particularly those from Amazon.co.uk, and is cautious about phishing attempts."}, 'name': 'save_recall_memory'}, 'type': 'function'}], 'refusal': None} response_metadata={'token_usage': {'completion_tokens': 38, 'prompt_tokens': 52866, 'total_tokens': 52904, 'completion_tokens_details': {'accepted_prediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 26752}}, 'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_b8bc95a0ac', 'finish_reason': 'tool_calls', 'logprobs': None} id='run-0c085a45-7602-4bc4-8f52-e3f9aa2ce932-0' tool_calls=[{'name': 'save_rec_all_memory', 'args': {'memory': "User is assessing the legitimacy of emails, particularly those from Amazon.co.uk, and is cautious about phishing attempts."}, 'id': 'call_pOhw91TOooauZmEM39T15FT', 'type': 'tool_call'}] usage_metadata={'input_tokens': 52866, 'output_tokens': 38, 'total_tokens': 52904, 'input_token_details': {'audio': 0, 'cache_read': 26752}, 'output_token_details': {'audio': 0, 'reasoning': 0}}
User is assessing the legitimacy of emails, particularly those from Amazon.co.uk, and is cautious about phishing attempts.
d6a62f20-e952-4baa-a1ab-7d0ee07029aa

```

Figure 67 - chatSupervisor memory usage

|        |        |   |  |
|--------|--------|---|--|
| 2      | ID     | 8f2c083e-6bec-4a43-ac2c-211d4ed06fd1  |  |
| SCORE  | FIELDS |   |  |
| 0.2922 |        | text: "User is assessing the legitimacy of an email from Amazon.co.uk regarding account sign-in notifications."<br>user_id: "1"   |  |
| 3      | ID     | 9ee5cabc-5bb8-43f7-8429-6dc44265bcc0  |  |
| SCORE  | FIELDS |   |  |
| 0.2933 |        | text: "User assessed an email from Amazon.co.uk as legitimate based on sender consistency, email content, subject line, and absence of suspicious patterns. User is aware of safety measures like verifying links, ..."<br>user_id: "1" |  |

Figure 68 - Pinecone database extract

### 7.3.5. Test results #7

Similarly to earlier tests, one group of the legitimate sender tests needed previous correspondence. This was done by create fake scenarios of communication between two people and providing that to the MongoDB (Appendix F for full text). To test the boundaries of the detection system, the final

email from the sender is about providing money to an account. The system correctly identifies the email as legitimate, both through classification and previous correspondence.

```
_id: ObjectId('67dd3a98d167a658d99a799e')
subject : "Weekend Plans?"
body : "Hey Alex,"

Hope you're doing well! Are we still on for the hike this w...
sender_name : "Ryan Carter"
sender_email : "ryan.carter@gmail.com"
email_domain : "gmail.com"

_id: ObjectId('67dd3adad167a658d99a799f')
subject : "Re: Weekend Plans?"
body : "Hey Ryan,"

Sounds great! I'm definitely in. What time were you thinkin...
sender_name : "Alex Thompson"
sender_email : "alex.thompson@yahoo.com"
email_domain : "yahoo.com"

_id: ObjectId('67dd3b05d167a658d99a79a0')
subject : "Re: Weekend Plans?"
body : "Hey Alex,"

Awesome! Let's meet at my place around 7:30 AM so we can ge...
sender_name : "Ryan Carter"
sender_email : "ryan.carter@gmail.com"
email_domain : "gmail.com"

_id: ObjectId('67dd3b27d167a658d99a79a1')
subject : "Re: Weekend Plans?"
body : "Hey Ryan,"

Perfect! I'll see you at 7:30. Looking forward to it!
- Al...
sender_name : "Alex Thompson"
sender_email : "alex.thompson@yahoo.com"
email_domain : "yahoo.com"
```

Figure 69 - Legitimate correspondence context

```
'subject': 'Re: Weekend Plans?',
'body': 'Hey Alex,\\n\\nQuick thing—since I went ahead and booked the parking spot and a couple of trail permits, it came out to about £20 total. If you could send me £10 for your half, that'd be great!\\n\\nYou can transfer it to me here:\\n\\n**Bank Name:** Barclays \\n**Account Number:** 74839201 \\n**Sort Code:** 04-12-56 \\n**Beneficiary Name:** R. Carter \\n\\nNo rush, just whenever you get a chance. See you Saturday!\\n\\n- Ryan',
'sender_name': 'Ryan Carter',
'sender_email': 'ryan.carter@gmail.com',
'email_domain': 'gmail.com'
```

The analysis indicates that the email is classified as legitimate with a confidence of 61.55%. Here's a summary of the key points from the analysis:

1. \*\*Sender Recognition\*\*: The sender, Ryan Carter, is someone you have corresponded with before, discussing social plans rather than sensitive topics.
2. \*\*Consistent Subject Line\*\*: The subject line "Weekend Plans?" aligns with previous communications, suggesting continuity.
3. \*\*No Financial Requests\*\*: There are no indications of financial transactions or requests in past emails with Ryan, which is a common red flag for phishing.

To further ensure the email's legitimacy, consider these additional checks:

- Look for requests for sensitive information, which are uncommon in emails from friends.
- Verify that the email address matches exactly with Ryan's known address.
- Check for any suspicious links or attachments that could indicate phishing.

If you have any further questions or need assistance with anything else, feel free to ask!

Figure 70 - Test 7 response

## Chapter 8 - Project Evaluation

The developed phishing detection system adopts an agentic workflow using a supervisor-agent architecture, incorporating five specialised agents. The proof of concept includes an email analysis agent powered by fine-tuned models and a URL scanner, a context-aware agent that reviews Outlook emails for expected communication patterns, a data extraction agent supporting formats such as .eml, .json, and .txt, a communication agent that presents results to users, and a supervisor agent that dynamically manages the workflow. This structured, automated pipeline highlights the viability of agents in streamlining real-world processes. Testing confirmed that all essential ('must do'), most recommended ('should do'), and some optional ('could do') requirements were met.

The research process was pivotal in shaping the system's development. While initial efforts focused on using LLMs for classification, phishing's complexity demanded a broader approach. This led to the adoption of agentic workflows with tool access, enabling automated tasks like content evaluation, context analysis, extraction, and user interaction. Research guided decisions on model fine-tuning, agent configuration, and workflow optimisation.

The requirements phase drew on research into existing phishing detection tools and insights from stakeholder questionnaires. This helped define the PoC's scope, ensuring alignment with technical feasibility and user expectations. A key insight was the importance of not just detecting phishing but also assisting users in understanding threats. This dual aim, automation and user education, shaped the system's design, balancing accuracy with usability.

Implementation involved adapting the initial design in response to practical challenges. A notable change was separating conversational tasks into more focused agents. Early designs had agents handling multiple functions, but testing showed improved performance when agents had singular, well-defined roles.

The multi-agent workflow proved effective. Currently, one agent handles both conversation and summarisation, though future iterations could benefit from further modularisation, such as adding a dedicated verdict agent. The supervisor agent, while functional, could be enhanced by adopting a plan-and-execute model, enabling it to set high-level goals and coordinate agent actions more intelligently.

Although this system is a proof of concept, several real-world considerations must be addressed. The extractor agent, for example, has access to server files and can execute code via the Python REPL tool, posing a potential security risk. This capability should be sandboxed or restricted to mitigate misuse.

Testing also revealed the system's susceptibility to prompt manipulation. Malicious inputs could influence agent behaviour or circumvent safeguards. To improve resilience, stronger input validation, intent filtering, and defensive prompt engineering are needed to guard against adversarial use. On a positive note, the classification models used within the email analysis agent demonstrated strong performance, achieving 98% accuracy on the validation dataset. These models also successfully leveraged the provided context in making decisions, effectively incorporating it into their output, a critical factor in distinguishing subtle phishing attempts from legitimate messages.

## Chapter 9 - Conclusion

This project set out to develop an efficient phishing detection system using LLMs, with a particular focus on minimising reliance on large-parameter models while enhancing accuracy, user awareness, and accessibility. Through a structured agent-based architecture and the application of prompt engineering, the system successfully addressed these aims.

The final system integrates specialised agents to manage specific tasks. This modular approach contributed to clear task separation and improved system maintainability. Ensemble analysis, combining fine-tuned classification models and external threat intelligence APIs, enabled robust phishing detection with over 98% accuracy on validation data. Contextual analysis of prior emails further improved the system's ability to distinguish between legitimate and malicious content, especially in more nuanced cases.

In addition to detection, the system was designed to support user education by offering breakdowns of detected threats, confidence scores, and contextual explanations. This emphasis on protection and clarity aims to empower users, particularly those in smaller organisations, by providing accessible cybersecurity insights without requiring specialist knowledge.

While the system remains a proof of concept, it demonstrates that LLM-driven, multi-agent approaches can effectively support phishing defence in real-world contexts. Key lessons include the importance of modular design, prompt engineering, and balanced datasets. Limitations such as prompt manipulation risks and system access permissions were identified, highlighting areas for future security improvements.

Overall, the project achieved its primary goals and demonstrated the feasibility of combining LLMs, agent-based design, and contextual analysis to build intelligent, user-focused phishing detection systems. The work contributes a practical foundation for future enhancements and wider deployment.

Following the completion of this project, Microsoft (2025), announced a similar multi-agent phishing detection system on 24<sup>th</sup> March 2025. Their system adopted a comparable architecture, validating many of the principles developed here, including agent collaboration and layered investigation. They extended the approach further by adding image analysis, QR code scanning, and detailed URL inspections using sandboxing, allowing for a more streamlined and cautious threat evaluation process. However, the originality of this project is clear, with the project poster submitted on 12<sup>th</sup> February 2025, demonstrating a forward-thinking application of LLMs in phishing defence ahead of Microsoft's release. The alignment between both systems highlights the relevance and robustness of the methods proposed.

## Chapter 10 – Future considerations

It is important to consider possible future work to improve the PoC and similar systems, such as the one offered by Microsoft, particularly in areas such as data privacy, reliance on cloud services, and the lack of sufficient training data.

One potential development is the use of edge devices for training models on user-specific data. This would allow more data to be analysed locally, potentially increasing model accuracy while preserving user privacy. To integrate these local models, federated learning techniques like FedAvg could be employed to aggregate the weights from each device. These aggregated weights could then be

pushed to a central server and redistributed back to all agents, maintaining a balance between training efficiency and privacy. However, a key challenge with this approach is data labelling, especially for incoming emails. Without access to labelled data, users may need to manually label messages to support supervised learning. Alternatively, semi-supervised or active learning methods could be explored to reduce the labelling burden.

Another critical aspect is the model's ability to run effectively on edge devices, which requires a lightweight model capable of operating efficiently on limited hardware while also maintaining high-quality training. Current research in TinyML focuses on developing such compact models, although an LLM capable of supporting this use case does not yet exist. If achievable, this would enable local agents on edge devices to autonomously analyse and act on emails, bringing intelligence and automation directly into the execution layer without the need for constant cloud-based processing.

While the system performs well as a proof of concept, scaling it for real-world deployment presents several challenges. The multi-agent architecture, which relies on external APIs and large language models, could face performance bottlenecks and increased costs when processing high volumes of emails. To handle real-time scanning across large organisational inboxes, optimised pipelines, parallel processing, and caching mechanisms would be necessary.

Ensuring long-term model reliability is critical as data evolves. To address concept drift (changes in email data distribution), a sliding window approach can be used. This focuses the model on recent data, helping it adapt to new phishing tactics while avoiding outdated information. By retaining only the latest data points, the system tracks emerging trends more efficiently and reduces memory overload. As noted in the literature review (Appendix A.4), this also mitigates class imbalance, improving adaptability and performance.

Allowing agents access to system-level tools like a Python REPL introduces additional risks. In production, these agents must be sandboxed with strict permission controls to prevent misuse. Integrating the system into corporate email platforms like Outlook or Gmail requires secure authentication and user-specific configurations, further complicating deployment. To enhance security—especially against prompt injection from untrusted email inputs—Debenedetti et al. (2025) proposed the CaMeL framework. It uses a dual-LLM setup: a quarantined LLM parses untrusted data without external access, while a privileged LLM generates code to act on that data. This separation ensures untrusted content cannot influence execution. Applying this to agents like the extractor, CAB, and Outlook analysis bot enables stricter data handling. With capability tags and security policies defining access, CaMeL strengthens resilience to prompt injection, ensuring secure, efficient processing.

Finally, to address the privacy concerns associated with using historical email data, robust safeguards must be implemented to comply with data protection regulations. While user authentication ensures controlled access, users must be fully informed about what data is being accessed, how it will be used, and must provide explicit consent. Additionally, if historical data is stored for analysis, it must be securely handled, with well-defined retention policies and the possibility of anonymisation or encryption to protect user privacy.

## Bibliography

- Mitchell, S.M. and Seaman, C.B. (2009) A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review. 2009 3rd International Symposium on Empirical Software Engineering and Measurement [online].
- Wang, Z. Et Al. (2024) 'Smart Expert System: Large Language Models as Text Classifiers' *Arxiv.org*. 2022 [online]. Available from: <https://arxiv.org/html/2405.10523v1>.
- Papageorgiou, E., Chronis, C., Iraklis Varlamis and Yassine Himeur (2024) A Survey on the Use of Large Language Models (LLMs) in Fake News. *Future Internet* [online]. 16 (8), pp. 298–298.
- Wahba, Y., Madhavji, N. and Steinbacher, J. (2022) A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks. *arXiv:2211.02563 [cs]* [online]. Available from: <https://arxiv.org/abs/2211.02563>.
- Bountakas, P. and Xenakis, C. (2023) HELPED: Hybrid Ensemble Learning PHishing Email Detection. *Journal of Network and Computer Applications* [online]. 210, p. 103545.
- Jiang, D., Ren, X. and Lin, B.Y. (2023) LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. [online].
- Nahmias, D., Engelberg, G., Klein, D. and Shabtai, A. (2024) *Prompted Contextual Vectors for Spear-Phishing Detection* *arXiv.org*. 14 February 2024 [online]. Available from: <https://arxiv.org/abs/2402.08309>.
- Sivarajkumar, S., Kelley, M., Samolyk-Mazzanti, A., Visweswaran, S. and Wang, Y. (2023) An Empirical Evaluation of Prompting Strategies for Large Language Models in Zero-Shot Clinical Natural Language Processing. *arXiv (Cornell University)* [online].
- Sumner, A., Yuan, X., Anwar, M. and McBride, M. (2021) Examining Factors Impacting the Effectiveness of Anti-Phishing Trainings. *Journal of Computer Information Systems* [online]. 62 (5), pp. 1–23.
- Koike, R., Kaneko, M. and Okazaki, N. (2023) *OUTFOX: LLM-Generated Essay Detection Through In-Context Learning with Adversarially Generated Examples* *arXiv.org*. 2023 [online]. Available from: <https://arxiv.org/abs/2307.11729>.
- Afane, K., Wei, W., Mao, Y., Farooq, J. and Chen, J. (2024) *Next-Generation Phishing: How LLM Agents Empower Cyber Attackers* *arXiv.org*. 2024 [online]. Available from: <https://arxiv.org/abs/2411.13874>.
- Monarch, R. (2021) *Human-in-the-loop Machine Learning: Active Learning and Annotation For Human* [online]. [Accessed 08 December 2024].
- Yu, Y., Kong, L., Zhang, J., Zhang, R. and Zhang, C. (2021) *AcTune: Uncertainty-aware Active Self-Training for Semi-Supervised Active Learning with Pretrained Language Models* *arXiv.org*. 2021 [online]. Available from: <https://arxiv.org/abs/2112.08787> [Accessed 8 January 2025].
- Ilia Shumailov, Zakhar Shumaylov, Zhao, Y., Papernot, N., Anderson, R. and Gal, Y. (2024) AI models collapse when trained on recursively generated data. *Nature* [online]. 631 (8022), pp. 755–759.

Guo, Y., Pu, J., Jiao, B., Peng, Y., Wang, D. and Yang, S. (2024) Online semi-supervised active learning ensemble classification for evolving imbalanced data streams. 1 April 2024 [online]. Available from: <https://www.sciencedirect-com.uwe.idm.oclc.org/science/article/pii/S1568494624002266?via%3Dihub>.

Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. (2023) QLoRA: Efficient Finetuning of Quantized LLMs arXiv.org. 23 May 2023 [online]. Available from: <https://arxiv.org/abs/2305.14314>.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. (2021) LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs] [online]. Available from: <https://arxiv.org/abs/2106.09685>.

Heraldi, F. D. & Ruskanda, Z. (2024) 'Effective Intended Sarcasm Detection Using Fine-tuned Llama 2 Large Language Models', in *International Conference on Advanced Informatics: Concept, Theory and Application*. [Online]. 2024 IEEE. pp. 1–6.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. and Cao, Y. (2023) ReAct: Synergizing Reasoning and Acting in Language Models arXiv.org. 9 March 2023 [online]. Available from: <https://arxiv.org/abs/2210.03629>.

Desai, D. and Hedge, R. (2024) Phishing Attacks Rise 58% in the Year of AI: ThreatLabz 2024 Phishing Report www.zscaler.com. 23 April 2024 [online]. Available from: <https://www.zscaler.com/blogs/security-research/phishing-attacks-rise-58-year-ai-threatlabz-2024-phishing-report>.

Sumner, A., Yuan, X., Anwar, M. and McBride, M. (2021) Examining Factors Impacting the Effectiveness of Anti-Phishing Trainings. Journal of Computer Information Systems [online]. 62 (5), pp. 1–23.

Wiesinger, J., Marlow, P. and Vuskovic, V. (2024) Agents. September 2024 [online]. Available from: [https://media.licdn.com/dms/document/media/v2/D561FAQH8tt1cvunj0w/feedshare-document-pdf-analyzed/B56ZQq.TtsG8AY-/0/1735887787265?e=1736985600&v=beta&t=pLuArcKyUcxE9B1Her1QWfMHF\\_UxZL9Q-Y0JTDuSn38](https://media.licdn.com/dms/document/media/v2/D561FAQH8tt1cvunj0w/feedshare-document-pdf-analyzed/B56ZQq.TtsG8AY-/0/1735887787265?e=1736985600&v=beta&t=pLuArcKyUcxE9B1Her1QWfMHF_UxZL9Q-Y0JTDuSn38).

Heiding, F., Schneier, B., Vishwanath, A. and Bernstein, J. (2023) Devising and Detecting Phishing: Large Language Models vs. Smaller Human Models arXiv.org. 23 August 2023 [online]. Available from: <https://arxiv.org/abs/2308.12287>.

Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H. and Ji, H. (2024) Executable Code Actions Elicit Better LLM Agents arXiv.org. 2024 [online]. Available from: <https://arxiv.org/abs/2402.01030> [Accessed 14 January 2025].

Talebirad, Y. and Nadiri, A. (2023) MULTI-AGENT COLLABORATION: HARNESSING THE POWER OF INTELLIGENT LLM AGENTS [online]. Available from: <https://arxiv.org/pdf/2306.03314>.

Xu, Z., Fang, X. and Yang, G. (2021) Malbert: A novel pre-training method for malware detection. *Computers & Security* [online]. 111, p. 102458. Available from: [https://www.sciencedirect.com/science/article/pii/S0167404821002820?casa\\_token=LVRcl1HZCOUAAAAA:KfcnlvxVhUGN7RApbYOL3H7XbXByll-sauVYeDCZw2fvAhMbK\\_rezUXrlSCHeI9QWuviPmatySk](https://www.sciencedirect.com/science/article/pii/S0167404821002820?casa_token=LVRcl1HZCOUAAAAA:KfcnlvxVhUGN7RApbYOL3H7XbXByll-sauVYeDCZw2fvAhMbK_rezUXrlSCHeI9QWuviPmatySk) [Accessed 16 May 2022].

- Caselli, T., Basile, V., Mitrović, J. and Granitzer, M. (2020) HateBERT: Retraining BERT for Abusive Language Detection in English. *arxiv.org* [online]. Available from: <https://arxiv.org/abs/2010.12472>.
- Joshi, M., Chen, D., Liu, Y., Weld, D.S., Zettlemoyer, L. and Levy, O. (2020) SpanBERT: Improving Pre-training by Representing and Predicting Spans. *arXiv:1907.10529 [cs]* [online]. Available from: <https://arxiv.org/abs/1907.10529>.
- Taha, A. (2021) Intelligent Ensemble Learning Approach for Phishing Website Detection Based on Weighted Soft Voting. *Mathematics* [online]. 9 (21), p. 2799.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2017) *Attention Is All You Need* *arXiv*. 12 June 2017 [online]. Available from: <https://arxiv.org/abs/1706.03762>.
- Gupta, M., Akiri, C., Aryal, K., Parker, E. and Praharaj, L. (2023) From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy. *IEEE Access* [online]. 11, pp. 80218–80245. Available from: <https://ieeexplore.ieee.org/abstract/document/10198233>.
- Sun, C., Qiu, X., Xu, Y. and Huang, X. (2020) *How to Fine-Tune BERT for Text Classification?* [online]. Available from: <https://arxiv.org/pdf/1905.05583>.
- Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.-W. and Lim, E.-P. (2023) *Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models* *arXiv.org*. 2023 [online]. Available from: <https://arxiv.org/abs/2305.04091?ref=blog.langchain.dev> [Accessed 26 March 2025].
- Miltchev, R., Rangelov, D. and Evgeni, G. (2024) Phishing validation emails dataset. 29 August 2024 [online]. Available from: <https://research.utwente.nl/en/datasets/phishing-validation-emails-dataset>.
- Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S. and Chadha, A. (2024) A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications Arxiv.org. 2024 [online]. Available from: <https://export.arxiv.org/abs/2402.07927> [Accessed 23 April 2025].
- ‘Cyber security breaches survey 2024’ (2024) GOV.UK. 2024 [online]. Available from: <https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2024/cyber-security-breaches-survey-2024>.
- Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., Kern, C., Shi, C., Terzis, A. and Tramèr, F., 2025. *Defeating prompt injections by design*. [online] arXiv. Available at: <https://arxiv.org/pdf/2503.18813>
- Microsoft, 2025. Security Copilot agents | Supercharging SOC efficiency: Meet the Phishing Triage Agent. [video] March. Microsoft Medius. Available at: <https://medius.microsoft.com/Embed/video-nc/ce51c0df-54e0-4e0a-8012-d42c03c3e30e?r=597084081565>

# Appendices

## Appendix A: Further literature review

### A.1

Another ensemble approach for detection of phishing and spear phishing is through Prompted Contextual Vectors, proposed by Nahmias et al. (2024). This method involves providing LLMs with human-crafted prompts to evaluate the contextual aspects of an email. These prompts are transformed into numerical vectors representing the email and concatenated with responses from other LLMs. The KNN algorithm then performs binary classification on the concatenated vectors to categorise the email as phishing or benign. This method achieves an F1 score of 0.97 for general phishing and 0.91 for spear phishing, indicating that spear phishing is more challenging to detect due to its similarity to normal writing. The dataset size, with only 333 generated phishing emails, may also be a limiting factor along with the reliance on crafted questions could bottleneck the LLMs' view of the email.

An alternative approach for classification tasks using LLMs is Zero-Shot Learning. This method leverages pre-trained knowledge to recognise and categorise text without task-specific training. Research by Sivarajkumar et al. (2023) explores various sub-strategies in the context of clinical sense disambiguation classification, including prefix prompt, Cloze, anticipatory, chain of thought, heuristic, ensemble, and few-shot prompt. The results show that heuristic and simple prefix prompts achieve the best results, with GPT-3.5 notably achieving a score of 0.96 using heuristic prompts. However, GPT-3.5's 175 billion parameters make it impractical for local device use. For smaller models like LLaMA-2, which has 70 billion parameters, prefix prompts yielded better results. Heuristic prompts involve constructing input prompts that guide the model's predictions, defined in the paper as a function that loops through tokens and rules to output a transformed sequence. Prefix prompts start the task with a guide to help the model interpret the input effectively. These methods heavily rely on prompt efficiency and may falter in complex situations with unseen cases.

### A.2

Another method of handling this is through active learning, where human annotators provide feedback on the data to assist in binary classification. Two useful techniques for this are uncertainty and diversity sampling. Uncertainty sampling involves presenting human annotators with data points where the model's certainty is below the threshold, such as 60%. By focusing on potentially mislabelled points, the model can learn from human input and improve future classifications. Diversity sampling should be used in combination with this approach; it involves selecting underrepresented data that might offer rare features beneficial to the model. Monarch, R. (2021) notes that this is particularly important for data that evolves over time, such as phishing data. After gathering these data points, they receive human annotation, and the model is retrained on this data.

### A.3

Yu et al. (2021) propose a framework for handling uncertainty sampling by alternating between batch sampling of uncertain data points for human annotation and self-training the model on high-certainty data points. This approach reportedly increases label efficiency by 56% compared to baseline active learning methods. However, this framework does not include diversity sampling,

which is crucial for enhancing the model's training. Combining uncertainty and diversity sampling would provide a more comprehensive and diverse dataset for the model.

#### A.4

When implementing a classification solution for phishing detection on a corporate scale, this will include thousands if not millions of data points, which can be classed as a data stream. This introduces various problems with the data which could damage the model when using active learning techniques; concept drift is a problem where the data which is gathered changes over time which leads to a mismatch between the model's learned representation and the current data distribution – For example, through various current world events like Covid 19. Class imbalance is another issue which will arise, this refers to a significant difference between two classes such as phishing emails and non-phishing emails, with the latter being more prominent. To combat these issues, the paper by Guo et al. (2024) discusses method which utilises a sliding window, which maintains a limited set of recent data to ensure adaptability and reduce memory requirements. This component is applicable to the phishing detection classifier, as the trends in the emails can be captured within it to keep the classifier up to date. The method also utilises a pseudo label which is based on the data point's proximity to the nearest cluster by using algorithm to infer the most likely label. This may not be as applicable to phishing emails content as they are complicated and so difficult to be clustered, however certain aspects of the email such as the email header could utilise this.

## Appendix B: Data processing design

### *Data Collection*

The collected phishing email data will serve two purposes: fine-tuning a language model for classification and training an ensemble model's weight distribution. To avoid bias, these datasets must be distinct, ensuring the fine-tuned model doesn't gain an unfair advantage by classifying familiar data.

To achieve broad coverage and prevent imbalance, data will come from multiple sources, maintaining an equal distribution of phishing and legitimate emails. The main sources include:

1. **Phishing Honeypot** – A publicly available GitHub repository containing phishing emails submitted by individuals and organisations for academic research.
2. **Phish Bowls** – Publicly available phishing emails published by universities, primarily sourced from Cornell University.
3. **CEAS 2008 Dataset** – A large dataset from the Conference on Email and Anti-Spam, sponsored by Microsoft, Google, and IBM. Despite its age, it remains valuable when combined with other sources.
4. **Nahmias et al. (2024) Dataset** – A dataset from the study on “Prompted Contextual Vectors for Spear Phishing Detection,” including both real and LLM-generated phishing emails, which have shown strong fine-tuning results.

### *Data cleaning*

To ensure effective model training, the data will undergo cleaning and normalisation with a focus on:

- **Handling missing values** – If minimal, missing entries will be removed.
- **Removing duplicates** – Eliminates biases from repeated data points.
- **Standardising format** – Since datasets vary in structure, all will be converted to a unified format with only two columns: text and label.
- **Removing identifiable data** – Source-specific details (e.g., university names in Phish Bowls) will be standardized to prevent location- or name-based correlations.

The final dataset will prioritise high-quality sources for fine-tuning, while lower-quality datasets will have reduced influence. To prevent bias from large but less desirable datasets like CEAS, their sample size will be reduced, ensuring smaller but more relevant datasets (e.g., Phishing Honeypot) contribute more prominently.

### *Validation dataset*

The validation dataset which will be used for the ensemble weight distribution will consist of the CEAS dataset and the generated emails to get initial weight distribution. As the CEAS dataset has a large amount of datapoints and most will be unused for the training, it will be useful for the validation.

## Appendix C: Data cleaning tools

| Component       | Data cleaning process                       | Tool                  |
|-----------------|---|-----------------------|
| Null values     | Removal                                     | Python Pandas         |
| Duplicates      | Removal                                     | Python Pandas         |
| URLs/links      | Replacement with [Link]                     | Python Pandas + Regex |
| Names           | Replacement with [Name]                     | SpaCy                 |
| Email addresses | Replacement with [Email]                    | Python Pandas         |
| Subject         | Format to start of email with [Subject:...] | Python Pandas         |

## Appendix D: Integration tests

| Test                | Requirements   | Test steps  | Expected results  | Result | Retest result |
|---------------------|----------------|---|---|--------|---------------|
| Memory saving       | FR1, FR5, NFR1 | Through the prompt, state a personal fact<br>Monitor console log and check Pinecone database.                   | The chatSupervisor should use the save_memory tool to store the statement under the user's ID into the Pinecone database.   | Pass   |               |
| Memory recall       |                | Prompt the system to recall a fact about the user.<br>Check Pinecone database for existence of relevant record. | The chatSupervisor should use the search_recall_memory tool to search the statement with the user's ID in the Pinecone database. The response should recall the statement | Pass   |               |
| Conversation test   |                | Through the prompt ask about the signs of phishing emails   | The chatSupervisor should respond with a valid list of signs of phishing emails.  | Pass   |               |
| Malicious request   |                | Prompt the system to write a marketing email which is malicious   | The system should deny the request and state only analysis is possible  | Fail   | Pass          |
| Malicious request 2 |                | Prompt the system to write a marketing email and analyse that email.  | The system should recognise that a step which is required to perform analysis, is not permitted.  | Fail   | Fail          |
| Analysis summary    | FR1, FR5, NFR5 | Provide an email to analyse and   | The chatSupervisor's summary should include both CAB's and the Outlook  | Pass   |               |

|  |  |   |                            |  |  |
|--|--|---|----------------------------|--|--|
|  |  | check the final output.<br>Monitor console log. | analysis agent's response. |  |  |
|--|--|---|----------------------------|--|--|

| Test         | Requirements | Test steps   | Expected results  | Result | Retest result |
|--------------|--------------|--|---|--------|---------------|
| PDF file     | FR8, NFR4    | Upload file in the specified format.<br>Monitor console terminal | Extractor bot extracts: <ul style="list-style-type: none"><li>• Sender name</li><li>• Sender email</li><li>• Recipient</li><li>• Subject</li><li>• Body</li><li>• Timestamp</li></ul> | Pass   |               |
| EML file     |              |  |   | Pass   |               |
| Json file    |              |  |   | Pass   |               |
| MSG file     |              |  |   | Pass   |               |
| exe file     |              |  |   | Pass   |               |
| TXT file     |              |  |   | Pass   |               |
| Prompt input | NFR1         | Input email in the prompt.<br>Monitor console terminal           |   | Pass   |               |

| ID   | Test                          | Requirements              | Test steps  | Expected results   | Result | Retest result |
|------|-------------------------------|---------------------------|---|--|--------|---------------|
| CAB1 | Obvious phishing email        | FR2, FR4, FR6, NFR8, NFR9 | Provide email through prompt or file upload.<br>Monitor console terminal. | CAB classifies the email as phishing with a confidence above 75% | Pass   |               |
| CAB2 | Standard phishing email       | FR2, FR4, NFR8, NFR9      |   | CAB classifies the email as phishing with a confidence above 65% | Pass   |               |
| CAB3 | Organisational phishing email | FR2, FR4, FR6, NFR8, NFR9 |   | CAB classifies the email as phishing with a confidence above 70% | Pass   |               |

|      |                            |                           |   |  |      |  |
|------|----------------------------|---------------------------|---|--|------|--|
| CAB4 | Business compromise email  | FR2, FR4, NFR8, NFR9      | Provide email through prompt or file upload. Input previous correspondence in MongoDB aligning new email with the previous. | CAB classifies the email as phishing with a confidence above 70%   | Pass |  |
| CAB5 | Fraud email                | FR2, FR4, NFR8, NFR9      | Provide email through prompt or file upload. Input previous correspondence in MongoDB aligning new email with the previous. | CAB classifies the email as phishing with a confidence above 60%   | Fail |  |
| CAB6 | Legitimate marketing email | FR2, FR4, FR6, NFR8, NFR9 | Provide legitimate email from Amazon. Analyse console logs for expected result.   | CAB classifies the email as Legitimate with a confidence above 60% | Pass |  |
| CAB7 | Legitimate sender          | FR2, FR4, NFR8, NFR9      | Provide email from a new sender. Analyse console logs for expected result.  | CAB classifies the email as Legitimate with a confidence above 80% | Pass |  |

| ID  | Test                          | Requirements   | Test steps  | Expected results  | Result | Retest result |
|-----|-------------------------------|----------------|---|---|--------|---------------|
| OA1 | Obvious phishing email        | FR3, FR9, NFR3 | Same as CAB1  | Agent finds no previous correspondence, determining this as suspicious            | Pass   |               |
| OA2 | Standard phishing email       |                |   |   | Pass   |               |
| OA3 | Organisational phishing email |                |   |   | Pass   |               |
| OA4 | Business compromise email     |                | Same as CAB4  | Agent finds previous correspondence but identifies the final email as suspicious. | Fail   |               |
| OA5 | Fraud email                   |                | Agent finds previous correspondence and identifies the chain of emails as suspicious, | Pass  |        |               |

|     |                            |  |  |  |      |  |
|-----|----------------------------|--|--|--|------|--|
|     |                            |  |  | especially with the urgency.   |      |  |
| OA6 | Legitimate marketing email |  | Same as CAB5   | Agent finds previous correspondence and identifies the chain of emails as legitimate | Pass |  |
| OA7 | Legitimate new sender      |  | Same as CAB6   | Agent finds no previous correspondence but identifies that the email isn't malicious | Pass |  |
| OA8 | Legitimate previous sender |  | Provide a crafted email relating to a payment. Input into MongoDB previous correspondence leading to this email. | Agent finds previous correspondence and identifies that the email isn't malicious    | Pass |  |

## Appendix E: Email chain for Business email compromise and fraud

```
{  
  "_id": {  
    "$oid": "67c08fa05bc49dd9bd8ddb57"  
  },  
  "subject": "Introduction – Partnership Inquiry",  
  "body": "Hi Lucas,\n\nI hope you're doing well! My name is John Reynolds, and I'm with FutureConsulting, a firm specializing in IT solutions for companies like [Target's Company Name].\n\nWe've been working with several companies in your industry, and I came across [Target's Company] through a recent industry report. I wanted to introduce myself and explore if there's any way we can support your team with technology solutions.\n\nNo rush at all—I'd love to learn more about your company and see if there's potential for collaboration in the future. Let me know if you'd be open to a quick 15-minute call sometime next week.\n\nLooking forward to hearing from you!\n\nBest,\nJohn Reynolds\nSenior Consultant | FutureConsulting\njohn.reynolds@futureconsulting.com\n+1 (555) 123-4567\n[Fake Website Link]",  
  "sender_name": "John Reynolds",  
  "sender_email": "john.reynolds@futureconsulting.com",  
  "email_domain": "futureconsulting.com"  
}  
  
{  
  "_id": {  
    "$oid": "67c0c4a7b4b68adb152335d8"  
  },  
  "subject": "Quick Follow-Up – Exploring Collaboration",  
  "body": "Hi Lucas,\n\nJust following up on my last email—totally understand if you're busy! I just wanted to check if there's a chance for a quick call to explore if FutureConsulting could add value to your team.\n\nNo pressure at all, but even if now isn't the right time, I'd love to keep in touch and send over some insights from our latest report on industry trends. Let me know if that'd be helpful!\n\nLooking forward to your thoughts.\n\nBest,\nJohn Reynolds",  
  "sender_name": "John Reynolds",  
  "sender_email": "john.reynolds@futureconsulting.com",  
  "email_domain": "futureconsulting.com"  
}  
  
{  
  "_id": {  
    "$oid": "67c0c4beb4b68adb152335d9"  
  },  
  "subject": "Great Connecting – Sharing Our Report",  
  "body": "Hi Lucas,\n\nThanks for your time on our call last week! It was great learning more about your role at [Target's Company Name] and how your team is working on [insert relevant business topic].\n\nAs promised, I'm attaching our latest market insights report. I think you'll find page 7 particularly interesting regarding security best practices.\n\nAlso, I noticed [mention something personal from the call, like an event or project they mentioned]. Hope that went well!\n\nLet's stay in touch—I'll follow up in a couple of weeks. Let me know if you need anything in the meantime.\n\nBest,\nJohn Reynolds",  
  "sender_name": "John Reynolds",  
  "sender_email": "john.reynolds@futureconsulting.com",  
  "email_domain": "futureconsulting.com"  
}  
  
{  
  "_id": {  
    "$oid": "67c0c4eab4b68adb152335da"  
  },  
  "subject": "Urgent: Updated Payment Details for Upcoming Invoice",  
  "body": "Hi Lucas & Jill,\n\nI hope you're both doing well! I'm following up regarding our vendor update. Our finance team recently completed the transition, and we now have updated banking details for processing payments.\n\nI wanted to send this over ahead of the next invoice cycle to prevent any delays. Please find our new payment details below:\n\nUpdated Payment Info:\n- Bank Name: Nationwide\n- Account Number: 123-456-789\n- SWIFT Code: Z78dwe133\n- Beneficiary Name: FutureConsulting Ltd.\n\nKindly update this in your system, and let me know if you need anything else. We really appreciate your support!\n\nBest,\nJohn Reynolds",  
  "sender_name": "John Reynolds",  
  "sender_email": "john.reynolds@futureconsulting.com",  
  "email_domain": "futureconsulting.com"  
}
```

## Appendix F: Email chain for Legitimate communication

```
{  
  "_id": {  
    "$oid": "67dd3a98d167a658d99a799e"  
  },  
  "subject": "Weekend Plans?",  
  "body": "Hey Alex,\n\nHope you're doing well! Are we still on for the hike this weekend? I was thinking we could try that new trail up in Blue Ridge-heard it has some amazing views!\n\nLet me know what you think.\n\nCheers,\nRyan",  
  "sender_name": "Ryan Carter",  
  "sender_email": "ryan.carter@gmail.com",  
  "email_domain": "gmail.com"  
}
```

```
{  
  "_id": {  
    "$oid": "67dd3adad167a658d99a799f"  
  },  
  "subject": "Re: Weekend Plans?",  
  "body": "Hey Ryan,\n\nSounds great! I'm definitely in. What time were you thinking of heading out? Should we bring some snacks or just grab something on the way?\n\nLooking forward to it!\n\n- Alex",  
  "sender_name": "Alex Thompson",  
  "sender_email": "alex.thompson@yahoo.com",  
  "email_domain": "yahoo.com"  
}
```

```
{  
  "_id": {  
    "$oid": "67dd3b05d167a658d99a79a0"  
  },  
  "subject": "Re: Weekend Plans?",  
  "body": "Hey Alex,\n\nAwesome! Let's meet at my place around 7:30 AM so we can get an early start. I'll bring some energy bars and water, but we can grab something bigger on the way if we need to.\n\nSee you Saturday!\n\n- Ryan",  
  "sender_name": "Ryan Carter",  
  "sender_email": "ryan.carter@gmail.com",  
  "email_domain": "gmail.com"  
}
```

```
{  
  "_id": {  
    "$oid": "67dd3b27d167a658d99a79a1"  
  },  
  "subject": "Re: Weekend Plans?",  
  "body": "Hey Ryan,\n\nPerfect! I'll see you at 7:30. Looking forward to it!\n\n- Alex",  
  "sender_name": "Alex Thompson",  
  "sender_email": "alex.thompson@yahoo.com",  
  "email_domain": "yahoo.com"  
}
```

# Appendix G: Stakeholder Questionnaire

## Phishing Detection Tool – Stakeholder Questionnaire

### Section 1: General Information

1. How frequently do you use email for work/study purposes?  
 a. Daily  
 b. A few times a week  
 c. Occasionally
2. Have you ever encountered a phishing email  
 a. Yes, frequently  
 b. Yes, but rarely  
 c. No, never
3. What email provider(s) do you use? (Gmail, **Outlook**, Yahoo, Other)
4. How confident are you in identifying phishing emails  
 a. 5 – Very confident  
 b. 4 – Confident  
 c. 3 – Somewhat confident  
 d. 2 – Unsure  
 e. 1 – Unable to distinguish

### Section 2 – Detection tool features

1. What features would you expect in a phishing detection tool? (Select all that apply)
  - Email content scanning
  - URL/Link analysis
  - Past email comparison
  - Integration with email service
  - Other (Please specify) **Option to save malicious email as sample for future analysis**
2. Would you prefer the tool to automatically block suspicious emails or provide warning for user action?  
 a. Block automatically  
 b. Provide warnings  
 c. Both, depending on the confidence level
3. To what level of detail should the explanations be, on why an email flagged as phishing?  
 a. Detailed explanations  
 b. Simple terms  
 c. Flagging is enough

**Providing a detailed analysis would enable people who are really in need of a tool like this to slowly become less dependent on it. From a user experience perspective the best thing this tool can do is become useless by educating its users so efficiently. As an application/product venture it would be better to retain customers by not emphasizing the education aspect, but I hope for the former.**

4. Should users be able to report false positives/negatives to improve detection accuracy?  
 a. Yes  
 b. No
5. Would you be comfortable allowing the tool to analyse your past emails (with confirmation) to improve phishing detection?  
 a. Yes  
 b. No

### Section 3: Additional feedback

1. Do you have other expectations that haven't been covered in this questionnaire?

No

2. Are there any other concerns which you have surrounding this tool?

No

