FINAL YEAR PROJECT REPORT

FYP02-CS-T2510-0060

INQUERIES: AN EXPLAINABLE RAG-AI CHATBOT FOR SCAM LINK DETECTION IN MESSAGING PLATFORMS

1211103098
NUR INQSYIRA BINTI ZAMRI

BACHELOR OF COMPUTER SCIENCE (HONS.) (CYBERSECURITY)

JUNE 2025

FYP02-CS-T2510-0060
INQUERIES: AN EXPLAINABLE RAG-AI CHATBOT FOR SCAM LINK DETECTION IN MESSAGING PLATFORMS

BY

1211103098 NUR INQSYIRA BINTI ZAMRI

FINAL YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENT FOR THE DEGREE OF

BACHELOR OF COMPUTER SCIENCE (HONS.) (CYBERSECURITY)

in the

Faculty of Computing and Informatics

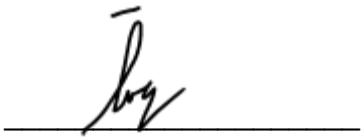
MULTIMEDIA UNIVERSITY
MALAYSIA

June 2025

Copyright

Declaration

I hereby declare that the work has been done by myself and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institution of learning.



Name of candidate:Nur Inqsyira Binti Zamri

Faculty of Computing & Informatics

Multimedia University

Date: 29.06.2025

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Mr. Khairil bin Anuar, my supervisor, for his invaluable insights, guidance, and unwavering support throughout this project. I am also sincerely thankful to Madam Qistina Najwa binti Mohamad Farid, my supervisor during FYP1, for believing in my potential and encouraging me to explore creative solutions along the way.

I would like to express my appreciation to Dr. Navaneethan A/L C. Arjuman, my moderator, whose thoughtful comments and constructive feedback truly helped me identify key areas for improvement and refine my approach moving forward in FYP2.

My thanks also go to Multimedia University (MMU) for providing the resources and platform essential for conducting this study. Special thanks to my lecturers and mentors, including those in the classroom, on YouTube, and in the broader online community, whose teachings shaped my understanding and supported me in tackling the continuous challenges faced throughout the completion of this project.

To my family and friends, your endless support, motivation, and belief in me have been my greatest source of strength, and I am forever grateful. Lastly, I extend my heartfelt gratitude to all the participants who contributed to this research by sharing their valuable insights. Your involvement has made this study truly meaningful for me.

This project stands as the result of many hands and hearts, and I am truly thankful to everyone who has been a part of this journey. Thank you.

Abstract

Phishing links now infiltrate messaging platforms like WhatsApp and Telegram with alarming ease, exploiting users' trust in peer-to-peer conversations. This project presents inQuiries, an explainable Retrieval-Augmented Generation (RAG) artificial intelligence (AI) chatbot that detects and explains suspicious Uniform Resource Locators (URLs) directly within a Telegram bot—eliminating the need to switch apps or use external tools. Built entirely on a zero-cost, publicly accessible stack, the system automates URL scanning, applies a rule-based risk classification, and generates clear, plain-language summaries via locally hosted large language models (LLMs). Key novelties include real-time integration in a chat interface, modular workflow orchestration via n8n, and a lightweight RAG pipeline using 1024-dimensional vector embeddings and Pinecone for retrieval. Testing showed that the system achieves average latencies of 17.74 seconds for cached URLs and 63.3 seconds for new ones, with cosine similarity scores above 0.70 consistently returned relevant threat matches. Despite no formal usability study, the bot delivered stable performance and consistent explanatory responses on limited hardware—highlighting its potential as a practical phishing detection tool within messaging platforms.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
AI	Artificial Intelligence
CPU	Central Processing Unit
FCI	Faculty of Computing and Informatics
FYP	Final Year Project
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HITL	Human-in-the-loop
IDE	Integrated Development Environments
IOC	Indicator of Compromise
LIME	Local Interpretable Model Agnostic Explanation
LLM	Large Language Model
MMU	Multimedia University
NLP	Natural Language Processing
PDF	Portable Document Format
RAG	Retrieval Augmented Generation
RAM	Random Access Memory
SHAP	SHapley Additive exPlanations
SMS	Short Message Service
URL	Uniform Resource Locator
XAI	Explainable Artificial Intelligence

List of Appendices

Chapter 1: Introduction

1.1 Overview

Messaging platforms such as WhatsApp and Telegram have changed the way people communicate. These platforms are now deeply embedded in everyday life, with billions of users exchanging messages across personal, professional, and social contexts. Their ease of use makes it simple to share links, media, and files. However, this same convenience has introduced new cybersecurity challenges.

Phishing attacks are increasingly targeting these messaging environments. Cybercriminals now distribute malicious Uniform Resource Locators (URLs) in both private and group chats, often disguising them to appear legitimate. These links may impersonate known contacts, trusted brands, or widely used services. When clicked, they can lead to credential theft, malware downloads, ransomware, or other harmful consequences. According to Symantec's Global Threat Report (2023), approximately one in three phishing attacks now originates in messaging applications. This shift away from traditional email-based attacks toward more informal, trusted platforms highlights an evolving threat landscape.

While platforms such as WhatsApp and Telegram include basic protective features—such as spam detection and link previews—these tools are largely reactive. They offer limited contextual information and are often bypassed when malicious links come from known contacts or appear in familiar settings. Users may still click on dangerous links without recognizing the warning signs. In many cases, they are left to evaluate links on their own or turn to external scanners and apps, which introduces friction and discourages regular use.

This growing disconnect between technical detection capabilities and practical user protection reflects a broader problem. As phishing tactics evolve, effective defenses must go beyond detection accuracy to support real-world

usability and comprehension. To address this challenge, the present study introduces a proof-of-concept system for scam link detection in messaging environments. The next section defines the specific problem this project seeks to address.

1.2 Problem Statement

Phishing detection has advanced significantly, with high-accuracy models and automated threat intelligence systems improving the ability to identify scam links. However, these tools remain largely inaccessible and impractical for everyday users. Many take the form of browser extensions, enterprise dashboards, or academic prototypes that prioritize detection performance over interpretability or usability. As a result, they are disconnected from the messaging platforms where phishing attacks increasingly occur.

Messaging platforms such as WhatsApp and Telegram have become primary targets for scam campaigns. Yet few phishing detection tools are designed to operate within these platforms or accommodate how users naturally interact with messages. Without integration, users are left to assess suspicious links manually and often in real time. Many tools offer only binary outputs such as “safe” or “unsafe,” without providing context or explanation. This places the burden of interpretation on the user, which is often confusing or ineffective, particularly for non-technical individuals.

As a result, awareness of scam link tactics remains low, and adoption of protective tools is minimal. Even when technically accurate solutions exist, they are often impractical in real-world use due to manual workflows, unclear feedback, or a lack of integration with everyday platforms.

This highlights a clear gap in the accessibility and interpretability of phishing detection for everyday users in messaging apps.

1.3 Objectives

The primary objectives of this research are:

1. To explore how recent phishing detection research can be operationalized into accessible, real-time tools that support safe link practices in messaging platforms.
2. To develop an automated workflow for scam link detection in messaging platforms with zero-cost requirements.
3. To design an artificial intelligence (AI)-powered mechanism that converts technical threat data into user-friendly summaries to support public understanding of scam risks.
4. To evaluate the system's practical feasibility and responsiveness under real-world usage conditions, focusing on latency performance and detection accuracy.

1.4 Scope

The scope of this project is defined across four areas: application, technical, user, and operational. Table 1.1 outlines what is included and excluded, based on the focus of the proof-of-concept.

Table 1.1: The scope of project

Scope Category	Description
Application Scope	Focused on detecting suspicious URLs shared in Telegram chats. It does not handle non-link scams (e.g., Portable Document Format (PDFs) or images).
Technical Scope	Built using a modular architecture that combines open-source tools (n8n, Ollama) with cloud-based services available under free-tier plans (VirusTotal, urlscan.io, Pinecone). Supports GPU (Graphics Processing Unit)-accelerated local LLMs (Large Language Models) such as LLaMA 3.2, and cloud-based LLMs like Gemini. Model fine-tuning, and explainability tools such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-gnostic Explanations) are explicitly excluded.
User Scope	Targeted at non-technical users. The chatbot handles link queries and summarizes risks. Advanced NLP (Natural Language Processing) tasks like sentiment or full chat parsing are out of scope.
Operational Scope	Designed for local execution via Docker and Ngrok, using free-tier APIs (Application Programming Interfaces) and a personal GPU. The system is not optimized for enterprise-scale deployment or formal large-sample usability testing.

1.5 Limitations

Design Tradeoffs

This project makes several deliberate design choices to prioritize simplicity, clarity, and accessibility:

- The system uses publicly available, pre-trained LLMs (e.g., LLaMA 3) without fine-tuning, executed locally via the Ollama framework. While this supports local deployment and zero-cost infrastructure, it may reduce fluency or explanation depth in edge cases.
- The risk classifier uses a custom rule-based heuristic, based solely on the number and type of verdicts returned from VirusTotal. Trusted vendors are referenced later to generate weighted confidence scores for explanation

purposes, but they do not influence the risk classification itself. While this rule-based approach enhances interpretability and transparency, it lacks the adaptability and scalability of data-driven or machine-learned classifiers.

- Cloud services, commercial Application Programming Interface (APIs), and model fine-tuning were excluded to maintain a fully self-hosted, zero-cost infrastructure. This constraint limited access to more powerful models and richer threat intelligence sources.

Evaluation Constraints

Certain aspects of the system were not evaluated or optimized due to resource or scope limitations:

- Real-time sandboxing was explored but not implemented due to complexity and API cost.
- No user testing was conducted to evaluate how well the explanations support comprehension or trust.
- Personalization and long-term feedback mechanisms are not implemented.
- The system is not tested for production-scale deployment or high-traffic robustness.

1.6 Methodology

This study adopts a design-based research methodology, which is appropriate for projects that aim to develop usable, real-world systems through iterative design and evaluation. The research focuses on building and deploying a lightweight, accessible system for scam link detection within messaging environments, with emphasis on human-centered explainability and technical feasibility.

The system architecture includes two primary components. First, a scam detection pipeline collects and analyzes metadata from third-party services (e.g., VirusTotal and urlscan.io) and classifies links into high, medium, or low risk using a custom rule-based heuristic. Second, a conversational explanation module powered by LLM generates natural language summaries to support user comprehension. This module relies on Retrieval-Augmented Generation (RAG), combining live semantic search with pre-written knowledge chunks.

The entire workflow is orchestrated using n8n, a low-code automation tool, and deployed through a Telegram chatbot interface to minimize user friction. To support accessibility and zero-cost deployment, the system avoids cloud APIs and model fine-tuning. Instead, it uses a publicly accessible, large language model (LLaMA 3.2 via Ollama) and integrates with Pinecone's free tier as the vector database to enable semantic retrieval and maintain adaptability within the RAG pipeline.

In addition to system design, the methodology includes a practical evaluation of performance under real-world usage scenarios. Two core aspects are assessed: (1) the system responsiveness, measured through end-to-end latency testing across different types of input, and (2) the accuracy of scam link detection via semantic retrieval. These evaluations aim to determine whether the system meets its design goals of usability, efficiency, and interpretability. Detailed testing procedures and results are presented in later chapters.

1.7 Target Audience

This report is intended for a diverse audience working at the intersection of cybersecurity, AI, and public safety. Given the project's emphasis on transforming cutting-edge research into accessible and explainable tools, several professional and academic groups are particularly aligned with its goals and contributions:

- **Cybersecurity Researchers and Academics**

  This report offers practical insights into real-time phishing detection, chatbot deployment, and the use of explainable AI (XAI) techniques such as LLMs and RAG. It contributes to ongoing discussions about how academic frameworks can be translated into tools that support public cybersecurity awareness and user trust.

- **AI/ML Engineers and Developers**

  For developers interested in deploying LLMs and RAG architectures in real-world, resource-constrained environments, this report provides a reproducible, cost-efficient blueprint. The system's use of n8n as a low-code orchestration tool, along with Pinecone, Ollama, and VirusTotal, demonstrates how modular AI workflows can be built with minimal infrastructure.

- **Cybersecurity Practitioners and Industry Professionals**

  Professionals focused on defending users from phishing, scams, and social engineering threats—especially in messaging platforms—may find the prototype useful as a model for early detection and in-chat intervention. While not a commercial product, the system illustrates how publicly accessible tools and real-time threat intelligence can be layered into everyday communication channels.

- **Human-Computer Interaction (HCI) Researchers and UX Designers**

  This project emphasizes human-centered design principles, particularly in the creation of clear, accessible explanations for non-technical users. The report will be valuable for those studying how to make cybersecurity tools more usable, transparent, and trustworthy.

- **Messaging-Platform Developers and Integrators**

  The system's Telegram chatbot, webhook orchestration, and automation workflows in n8n provide a replicable foundation for integrating security features into chat platforms. Developers working on secure communication tools may find the implementation details helpful for future extensions.

- **Public Awareness Advocates and Policymakers**

  This project aligns with broader goals of digital literacy, online safety education, and inclusive access to security technologies. It demonstrates how advanced AI models can be repurposed into low-cost tools that promote scam awareness and informed decision-making. The system's emphasis on explainability and transparency supports emerging ethical AI standards and public-facing cybersecurity policy.

1.8 Summary

This chapter established the motivations, goals, and scope of this research project, focusing on the need for accessible, interpretable phishing detection tools within messaging environments. It outlined the problem landscape, defined the project objectives, and described the system's intended audience and technical boundaries. The chapter also emphasized the importance of human-centered explainability in promoting public cybersecurity awareness. The next chapter reviews prior work on phishing detection, XAI, and RAG, which collectively inform the design of the proposed solution.

Chapter 2: Literature Review

2.1 Overview

This chapter reviews recent research published between 2021 and 2025 across four domains that underpin the system's design: phishing detection, human-centered security, XAI, and RAG. The goal is to identify technical patterns, usability gaps, and design principles that inform the architecture and functionality of the proposed chatbot. Special attention is given to detection models suited to conversational environments and the role of explainability in building user trust in AI-driven security tools.

2.2 Detection Models in Conversational Contexts

2.2.1 From Traditional Detection to LLM-Based Intelligence

Traditional phishing detection methods were built on blacklists, handcrafted lexical rules, and domain-based heuristics. While straightforward to implement, these systems often struggled to generalize. Many phishing campaigns today rely on obfuscation, shortened URLs, and dynamic domain behavior that evade such static filters. Basit et al. (2021) noted that in realistic settings, blacklist-based detection achieved less than 20% recall against novel phishing samples. This limitation reflects a broader trend: as phishing techniques evolve rapidly, rule-based systems fall behind in both coverage and adaptability.

To overcome this, researchers have turned to machine learning. Ozcan et al. (2023) proposed hybrid architecture combining deep neural networks (DNN) and long short-term memory (LSTM) units. Their model achieved over 97% accuracy in standard benchmarks, demonstrating strong classification performance. However, it lacked interpretability, and the decision-making process was not transparent. This raise concerns in real-world applications where users need to understand why something was flagged as malicious. Without explanation, users may ignore alerts or distrust the system entirely.

On the other hand, Tupsamudre et al. (2021) introduced a layered approach with PhishMatch, blending pattern recognition with supervised learning to improve phishing detection while reducing false positives. The system improved efficiency but still depended on structured inputs and labeled datasets, limiting flexibility in unstructured or multilingual environments such as messaging platforms. These systems often perform well under benchmark conditions but struggle in real-world conversational contexts, which are typically informal, multilingual, and structurally ambiguous.

2.2.2 LLMs in Scam Detection

LLMs offer new capabilities in phishing detection by enabling semantic reasoning, contextual interpretation, and response generation. Unlike traditional classifiers, LLMs can infer threat signals from language cues, tone, and intent. Chang and Aïmeur (2024) evaluated five LLMs against multilingual scam scenarios taken from real-world chat messages. Their results showed that models like LLaMA-3-8B and Gemma-2-9B performed exceptionally well, achieving over 0.97 accuracy and recall scores up to 0.976. These models consistently outperformed human participants in detecting nuanced deception techniques, including trust-building narratives and psychological manipulation.

This capability suggests that LLMs are well-suited for conversational settings, where social context matters as much as lexical content. Unlike traditional phishing filters, which treat messages as isolated strings, LLMs can reason over language patterns that signal intent, persuasion, or impersonation.

In a user survey conducted for this study, 74 participants were asked how confident they felt identifying scam links before clicking. Figure 2.1 presents the breakdown: 36% reported feeling very confident, while 59% were somewhat confident. Only 4% indicated they were not confident. These numbers reflect a general belief that users can recognize scams, especially in messaging platforms where trust is often inferred from the sender’s identity.

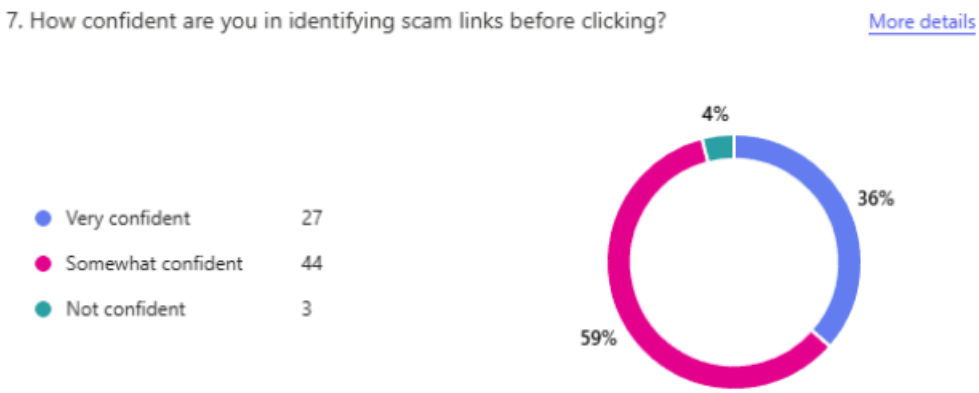


Figure 2.1: Confidence in Identifying Scam Links Among WhatsApp Users

However, this perceived competence does not always align with actual detection performance. While users believe they can identify scam links reliably, previous research shows that even security-conscious individuals often fall for well-crafted phishing messages. Figures 2.2 and 2.3, based on empirical studies from Chang and Aïmeur (2024), highlight the gap in performance between human users and LLMs. Human accuracy fluctuated depending on context and message design, whereas LLMs maintained consistently high scores across both malicious and benign samples.

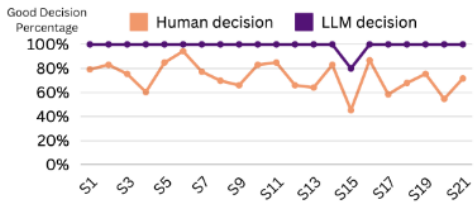


Figure 2.2: Detection Accuracy – Human vs. LLMs on Scam Messages

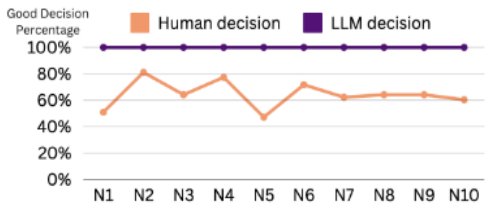


Figure 2.3: Detection Accuracy – Human vs. LLMs on Benign Messages

This disparity suggests an overconfidence bias, where users overestimate their detection ability. It also reinforces the need for systems that do not just label links as risky or safe but explain the rationale behind such decisions. Transparent, conversational feedback can help bridge this confidence gap and support informed decision-making, especially in environments where scams rely on social engineering rather than technical exploits.

2.3 Human-Centered and XAI Approaches

2.3.1 The Transparency Challenge

As detection systems grow more complex, the demand for transparency increases. Users are less likely to engage with tools they do not understand, even if those tools are highly accurate. Wei et al. (2023) interviewed 24 cybersecurity professionals and found that end users are frequently blamed for security breaches, yet the systems meant to protect them often fail to communicate effectively.

Additional insights come from Riasat et al. (2022), who reported that users frequently ignore security indicators in mobile apps because the signals are difficult to interpret. Even accurate systems may fail if users do not understand or trust their outputs. This highlights a growing consensus: detection alone is not enough—systems must communicate risk in a way users can interpret and act upon.

2.3.2 Design Approaches for User-Centered Security

Designing usable security tools requires empathy for the cognitive limitations of everyday users. Macke et al. (2023) proposed a human-in-the-loop (HITL) framework that allows users to receive gradual feedback and interact with the system over time. Instead of static alerts, the system offers context-aware explanations that adjust based on user behavior. This approach builds trust and fosters long-term engagement. It also reflects a shift from reactive design to proactive security education.

XAI plays a central role in enabling this feedback. Aljarah et al. (2025) introduced EXPLICATE, a framework that integrates model explanation tools like SHAP and LIME with LLM-based summarization. The goal is to turn complex, low-level model behavior into readable summaries that make sense to non-technical users.

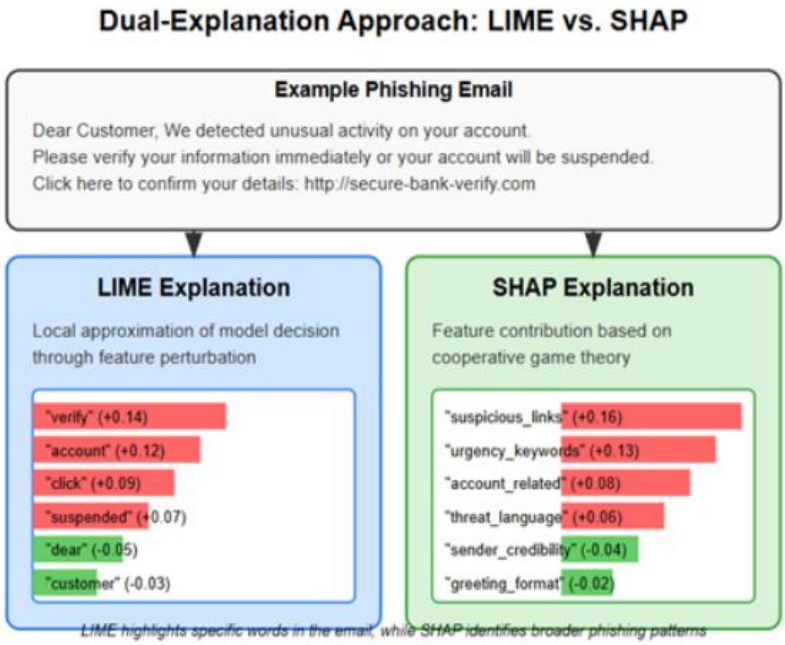


Figure 2.4: Comparison of LIME and SHAP Explanations of a Phishing Email

As illustrated in Figure 2.4, SHAP and LIME surface different feature contributions for the same input. When paired with an LLM summarizer, these features can be translated into natural language justifications, improving user comprehension and response accuracy.

A complementary example is ChatSpamDetector (2024), which uses transformer-based models to classify and explain scam messages in real time. Its architecture supports message parsing, simplification, and explanation generation, which are all delivered within a chat interface. The system improves not only detection accuracy but also user comprehension, offering a model for conversational cybersecurity tools.

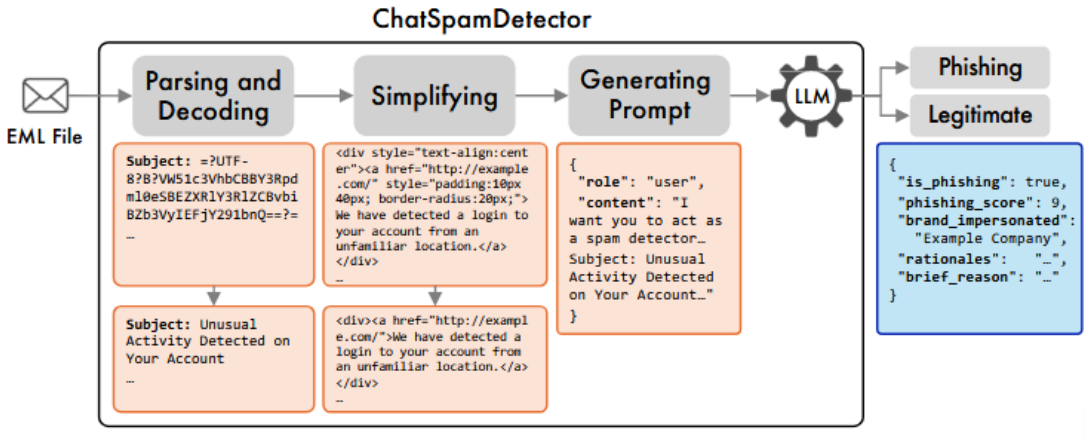


Figure 2.5: System Architecture of ChatSpamDetector

These examples point toward a common conclusion: detection systems must be designed with user understanding in mind. Accuracy without clarity is insufficient, especially in messaging environments where rapid judgment is required.

2.3.3 RAG for Cybersecurity

RAG is increasingly used to overcome one of the major limitations of LLMs: their lack of access to current and domain-specific information. In a RAG system, external knowledge is retrieved from a document store and appended to the input prompt. This allows the model to generate responses grounded in factual context, rather than relying solely on pre-trained data.

In cybersecurity settings, this approach enhances both reliability and explainability. RAG systems help ensure that output is not only accurate but also traceable to real-world evidence. This is especially important when users need to understand why a threat was flagged. A good explanation builds trust, and trust increases adoption.

Recent advances have further expanded the role of RAG. Kurniawan et al. (2024) proposed CyKG-RAG, a framework that integrates structured knowledge graphs into the retrieval process. Their model showed improved

performance on cybersecurity tasks, especially in scenarios where factual grounding and contextual relevance were critical. The inclusion of knowledge graphs helped organize threat data, making the system more interpretable and robust.

Evaluation methods for RAG systems have also evolved. Kenneweg et al. (2024) introduced RAGVAL, an automatic benchmarking framework for assessing the quality of RAG-generated outputs. Their study demonstrated that chunk size and retrieval settings have a significant effect on both accuracy and computational efficiency. Using GPT-4 as an evaluator, they found that small variations in configuration could impact how truthful and relevant the generated responses were.

These insights are directly relevant to phishing detection. Instead of relying on static rule-based classifiers, a RAG pipeline can dynamically retrieve scan metadata, such as antivirus verdicts and page screenshots, and use it to produce an explanation in natural language. This approach not only improves the detection process but also helps users understand the reasoning behind it.

Collectively, these developments highlight the strong potential of RAG in cybersecurity, especially in scenarios that demand real-time, explainable, and user-facing outputs. This project builds on these findings by incorporating RAG techniques into a lightweight, messaging-based prototype.

2.4 Synthesis and Implications

Building on the reviewed literature, several recurring gaps emerge that directly inform the goals and design priorities of this project. These include issues in explanation, real-time responsiveness, usability, and adaptability—each limiting the practical impact of current phishing detection systems. The proposed chatbot directly addresses these limitations by integrating a RAG-based explanation pipeline, real-time messaging support, and user-friendly outputs

tailored for non-technical users. Table 2.1 summarizes these challenges, the supporting research, and their design implications for the proposed solution.

Table 2.1: Key Gaps Identified in Literature and Their Implications

Gap	Description	Supporting Literature	Implication for This Project
Trust and Explanation	Users often overestimate their ability to detect phishing and ignore alerts that lack clear reasoning. Binary "safe/unsafe" labels are insufficient for informed decision-making.	Wei et al. (2023); Macke et al. (2023)	The system generates natural language explanations of scan results to improve user understanding and support trust in warnings.
Real-Time Phishing in Messaging Platforms	Phishing has shifted from email to messaging platforms, where traditional filters are less effective and users rely more on informal trust cues.	Chang & Aïmeur (2024); Koide et al. (2024)	The prototype is deployed within Telegram and responds in real time, allowing users to query suspicious links inside chat environments.
Usability Limitations in Existing Tools	Many phishing detection tools are technically effective but not usable by the general public. They often require copying links into external tools or interpreting technical outputs.	Riasat et al. (2022); Aljarah et al. (2025)	The system integrates detection into a familiar chat interface and provides clear, simplified outputs tailored to non-technical users.
Static Detection vs. Adaptive Explanation	Static classifiers struggle to adapt to evolving threats and lack grounding in real-world evidence. Users often receive unexplained or outdated responses.	Kurniawan et al. (2024); Kenneweg et al. (2024)	The system uses a RAG pipeline to generate explanations based on live scan metadata, improving grounding, timeliness, and response relevance.

2.5 Summary

This chapter reviewed recent advances in phishing detection, XAI, human-centered design, and RAG. The literature highlights growing concerns around the usability and interpretability of detection systems, particularly in messaging environments where scams rely more on social trust than technical exploits. While traditional phishing tools focus on accuracy, many fail to explain their reasoning or adapt to real-world usage patterns, limiting user engagement and trust.

Researchers have proposed various approaches to address these limitations, including human-in-the-loop design, natural language explanations, and RAG architectures that enhance grounding and transparency. However, many existing systems remain difficult to use, require technical expertise, or operate outside of messaging contexts where users most often encounter scams.

This project builds on these insights by developing a chatbot-based prototype that integrates phishing detection into a messaging environment (Telegram). While the system does not analyze conversational context, it supports dynamic, evidence-based explanation by retrieving and summarizing metadata from live scan sources. In doing so, it aims to bridge the gap between technical capability and public usability, offering a lightweight, accessible tool that enhances awareness and decision-making during scam encounters.

Chapter 3: Requirement Analysis

3.1 Overview

This chapter outlines the requirements for designing and implementing a chatbot-based scam link detection system intended for public use. The system is intended to assist users in identifying and understanding suspicious links directly within messaging platforms. To ensure technical feasibility, user accessibility, and alignment with cybersecurity best practices, this chapter defines the functional, non-functional, and user-oriented requirements based on exploratory research and early-stage prototyping.

3.2 Fact-Finding Techniques

Table 3.1 summarizes the fact-finding techniques used to inform system design decisions. A combination of the following methods was employed to identify both functional requirements and user-centered considerations. These approaches ensured that the system would be technically feasible, practically deployable, and aligned with real user needs.

Table 3.1: Fact-Finding Techniques

Technique	Description	Outcome
Literature Review	Synthesized academic research on phishing detection, human-centered design, RAG architecture, and XAI.	Identified technical design patterns and usability gaps in existing tools.
User Survey (n=74)	Surveyed users about their experiences with scam links on messaging platforms. Respondents were mostly aged 18–24, followed by 25–34. 66% were students, and 30% were working professionals, mainly from Computer Science/IT backgrounds. Most working participants were from the private sector.	Found that 99% had encountered suspicious links; few used security tools.
Exploratory Prototyping	Iterative development using n8n, Telegram API, and public threat intelligence services.	Validated system feasibility and refined architectural choices.
Tool Documentation Review	Studied API docs and community discussions for services like VirusTotal, urlscan.io, Hybrid Analysis, Pinecone, and n8n.	Identified integration limits and practical constraints for zero-cost deployment.
Pipeline Testing with Labeled Dataset	Used the data_bal - 20000.xlsx dataset from the ESDAUNG PhishDataset (GitHub) containing balanced benign and phishing URLs. Used solely for internal system validation.	Verified that the system could process labeled inputs and produce appropriate risk classifications and explanations end-to-end.

3.3 Requirement

3.3.1 Functional Requirements

Table 3.2 outlines the core functional requirements for the scam link detection chatbot. These describe the essential capabilities the system must perform, including link analysis, risk classification, user interaction, and explanation generation within a messaging environment.

Table 3.2: Functional Requirements

ID	Requirement
FR1	The system shall receive messages containing URLs via the Telegram chatbot.
FR2	The system shall extract and preprocess URLs from the received message.
FR3	The system shall submit the URL to VirusTotal and urlscan.io for scanning.
FR4	The system shall parse metadata returned by both APIs.
FR5	The system shall apply a custom rule-based algorithm to classify the risk level (High, Medium, Low) and calculate confidence score.
FR6	The system shall embed the metadata as a vector using a local embedding model and store it in Pinecone.
FR7	The system shall retrieve only contextually relevant documents using vector similarity.
FR8	The system shall generate a natural language explanation using a locally hosted LLM.
FR9	The system shall return the risk level and explanation to the user via Telegram.

3.3.2 Non-Functional Requirements

Table 3.3 presents the non-functional requirements that define the quality, performance, and operational constraints of the system. These include factors such as latency, reliability, scalability, and usability, which ensure the system functions effectively under real-world conditions.

Table 3.3: Non-Functional Requirements

ID	Requirement
NFR1	The system shall operate using zero-cost tools and services (free-tier APIs, open-source components).
NFR2	The system shall be hosted locally using containerized environments.
NFR3	The system shall respond to user messages within an acceptable delay, defined as: • ≤ 60 seconds for general message (non-URL), • ≤ 120 seconds for messages containing URLs (including API calls and vector store insertion via RAG)
NFR4	The system shall use lightweight models compatible with constrained hardware (≤ 16 GB (gigabytes) of RAM (Random Access Memory), using either CPU (Central Processing Unit) or GPU (Graphics Processing Unit) for inference.
NFR5	The system shall provide explanations in clear, jargon-free language.
NFR6	The system shall store only necessary metadata, avoiding full document storage to reduce vector size.
NFR7	The system shall be modular, allowing individual components (e.g., embedding or retrieval) to be updated independently.

3.3.3 User Requirements

Table 3.4 summarizes the key user requirements identified for the scam link detection chatbot. These requirements reflect what users expect in terms of interaction simplicity, platform accessibility, and clarity of output. The focus is on ensuring usability for casual, non-technical individuals without requiring prior cybersecurity knowledge.

Table 3.4: User Requirements

ID	Requirement
UR1	The user shall be able to interact with the system through Telegram by forwarding or pasting message containing suspicious links.
UR2	The user shall receive a clear risk label (e.g., “High Risk”) along with a simplified explanation.
UR3	The user shall not need to install or configure additional software to use the system.
UR4	The user shall not be required to understand cybersecurity terminology to interpret the system output.
UR5	The system shall support casual, non-technical users who have no prior experience with phishing detection tools.

Chapter 4: System Design

4.1 Overview

This chapter presents the system architecture and design principles behind the prototype scam link detection chatbot. The system is designed to be modular, transparent, and fully deployable on local infrastructure. Key components include a Telegram-based user interface, a heuristic risk classifier, an RAG for explanation module, and a backend workflow for threat intelligence. The architecture emphasizes semantic retrieval, transparency, and lightweight deployment—aligning with the project's goals of accessibility and real-world feasibility.

4.2 System Architecture Overview

The system is composed of two main workflows, implemented using n8n, a low-code workflow automation tool for orchestrating tasks, APIs, and logic flows:

1. **Frontend Workflow**

   This workflow handles user interaction through Telegram, processes incoming messages, extracts URLs (treated as Indicators of Compromise or IOCs), applies a heuristic-based risk classifier, and routes the query to either a cached response or a backend scan. If no prior scan exists, it forwards the request to the LLM explanation module, which uses RAG to generate a natural-language response.

2. **Backend Workflow**

   This workflow is triggered when a URL has not been previously scanned. It queries external threat intelligence APIs (VirusTotal and urlscan.io), transforms and embeds the returned metadata, stores the result in a cloud-hosted vector database (Pinecone free tier), and logs the entry for future cache checking by the frontend.

These workflows are modular and interconnected, with the frontend handling user interaction and the backend performing scan operations only when required. This design enables efficient orchestration, ease of debugging, and local deployment. Figures 4.1 and 4.2 illustrate the system's two primary workflows.

4.3 Frontend Interaction Workflow

The frontend workflow is initiated when a Telegram message is received by the system. The message is analyzed to detect the presence of any URLs. If a link is found, the system checks whether it has been previously processed. If so, a cached explanation is retrieved; otherwise, the backend analysis workflow is triggered.

If the message does not contain a URL, it is interpreted as a general user query and forwarded to the LLM chain for contextual natural language generation.

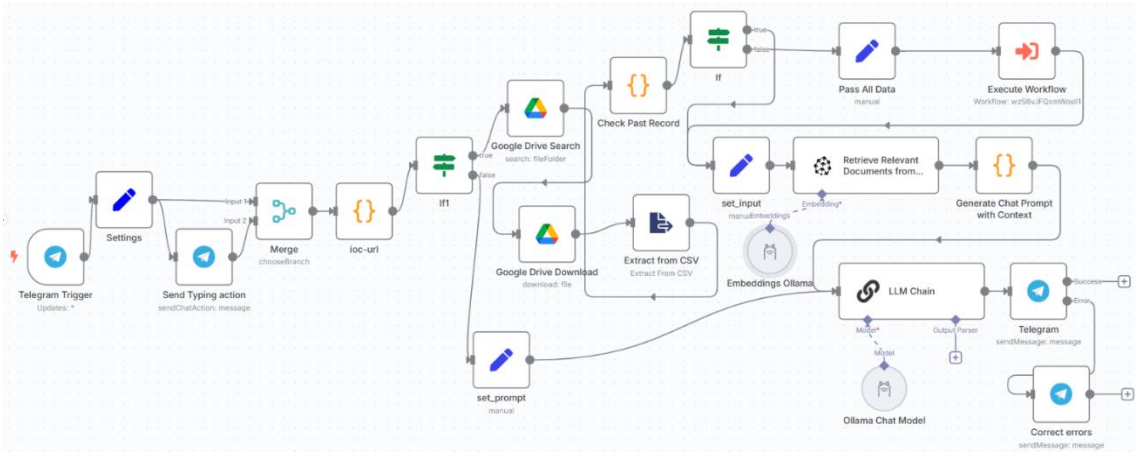


Figure 4.1: inQueriesFrontend-ver1

Key Components (Figure 4.1):

- Telegram Trigger: Listens to incoming Telegram messages and initiates the workflow.
- Settings Node: Dynamically sets system behavior based on user metadata and language.
- Send Typing Action: Sends a typing indicator to enhance UX.
- IOC Identification (ioc-url): Python script using the ioc-finder package extracts URLs from messages.
- Conditional Routing:
    - Past Record Check: Searches a Google Sheet to determine if a scanned result already exists.
    - If found, the pipeline retrieves context from the Pinecone vector store and forwards it to the LLM chain for semantic explanation.
    - If not, it calls the inQueriesBackEnd-ver1 via Execute Workflow node.
- LLM Chain:
    - Powered by a local Ollama LLM configured with the llma3.2:latest model.
    - Relies on a system prompt and the user's query to generate contextual reasoning and a natural language response.
- Telegram Output: Sends results back to the user using either plain response or error-corrected output.

This workflow balances responsiveness with explainability while avoiding redundant scanning and speculative LLM outputs.

4.4 Backend Threat Analysis Workflow

The backend workflow is triggered only when a URL has not been previously processed. It collects real-time threat intelligence from external sources (VirusTotal and urlscan.io), transforms the results into structured metadata, and performs semantic embedding. These outputs are cached for reuse and forwarded to the explanation module. This modular, just-in-time architecture reduces redundant API calls, improves efficiency, and ensures resource-aware local deployment.

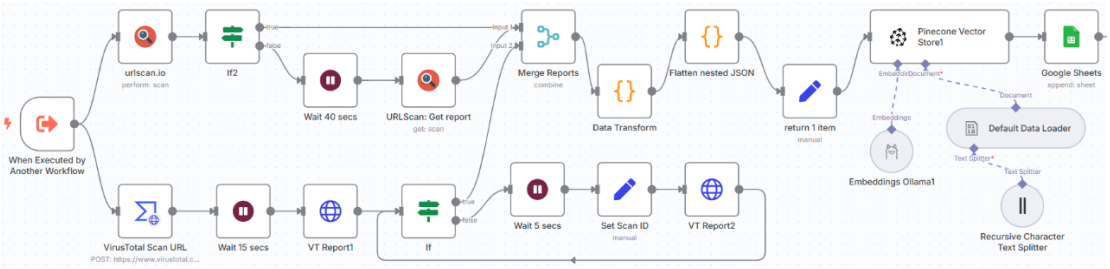


Figure 4.2: inQueriesBackend-ver1

Key Components (Figure 4.2):

- Trigger (When Executed by Another Workflow): The entry point, activated exclusively via the front-end workflow.
- Threat Intelligence Collection:
    - urlscan.io: Submits the URL and waits up to 40 seconds for results.
    - VirusTotal Scan: Submits the URL to VirusTotal's API and retrieves an analysis ID.
    - Wait & Fetch Logic: Includes dynamic delays and status polling to ensure report availability (15s, 40s, 5s waits),
    - Report Retrieval: Fetches full reports from both services once available.

- Data Consolidation:
    - Merge Reports: Combines data from urlscan.io and VirusTotal into a unified structure.
    - Data Transform: Applies heuristics to classify threat levels (High/Medium/Low) and calculates confidence scores. Generates a predefined threat summary to standardize input for the explanation module.
    - Flatten Nested JSON: Converts hierarchical JSON into flat key-value pairs for consistent downstream processing.
- Semantic Embedding & Storage:
    - Embedding using Ollama: Transforms threat summaries into vector representations.
    - Metadata Annotation: Adds relevant metadata (e.g. scan ID, risk level, exact URL).
    - Pinecone Vector Store Insertion: Embeds structured summaries for future semantic search by the front-end RAG system.
- Logging:
    - Google Sheets Append: Logs the scan results into a Google Sheet with fields including scanned link, scan ID, and risk level, ensuring traceability and offline review.

4.5 RAG Design

The RAG component serves as the bridge between raw threat intelligence and human-readable explanation. Instead of relying on model fine-tuning, it retrieves semantically relevant documents and prompts a local language model to generate grounded, contextual responses.

Key Components:

- Vector Store (Pinecone):
    - Indexed with processed scan metadata.
    - Segregated by risk level namespace for semantic filtering.
    - Queried using document embeddings derived from actual scan summaries.
- Embeddings (Ollama):
    - Uses mxbai-embed-large to convert flattened scan summaries into vector representations.
    - Embeddings are computed locally to minimize processing latency and reduce reliance on commercial LLM APIs.
- Local LLM (Ollama - llama3.2):
    - Retrieves relevant scan documents from Pinecone.
    - Generates answers based on embedded page content, risk classifications, and associated metadata.
    - Responds in the user's preferred language, with format and tone guided by a structured system prompt.

This structure maintains response consistency, relevance to real-world threat evidence, and ensures hardware efficiency for local operation.

4.6 Rule-Based Heuristic

The risk classification logic in InQueries relies on a handcrafted, rule-based heuristic, carefully structured to mimic expert reasoning using only scan verdicts retrieved from Virustotal's participating partners. The algorithm assesses the number of "malicious" and "suspicious" flags from trusted scanning engines, producing a corresponding risk level and confidence-based explanation.

Initially, the system iterates through each scan verdict to count how many reputable engines (from a predefined trustedEngines set) flag the URL as either malicious or suspicious. Based on these counts, the logic assigns one of the following risk tiers:

- High Risk
- Medium Risk
- Low Risk

After classification, the system uses a secondary heuristic to generate natural language justifications. It filters only the verdicts issued by trusted engines and groups them by verdict category (e.g., "phishing", "malware", etc.). These categories are then summarized into sentences such as “Reputable engine(s), like Microsoft, Kaspersky, and ESET, has categorized the link as phishing.” Figure 4.3 below illustrates the end-to-end heuristic logic: from verdict counting, risk classification, confidence computation, to trusted explanation generation. Each condition and decision node reflects real logic implemented in JavaScript-based modules powering the system.

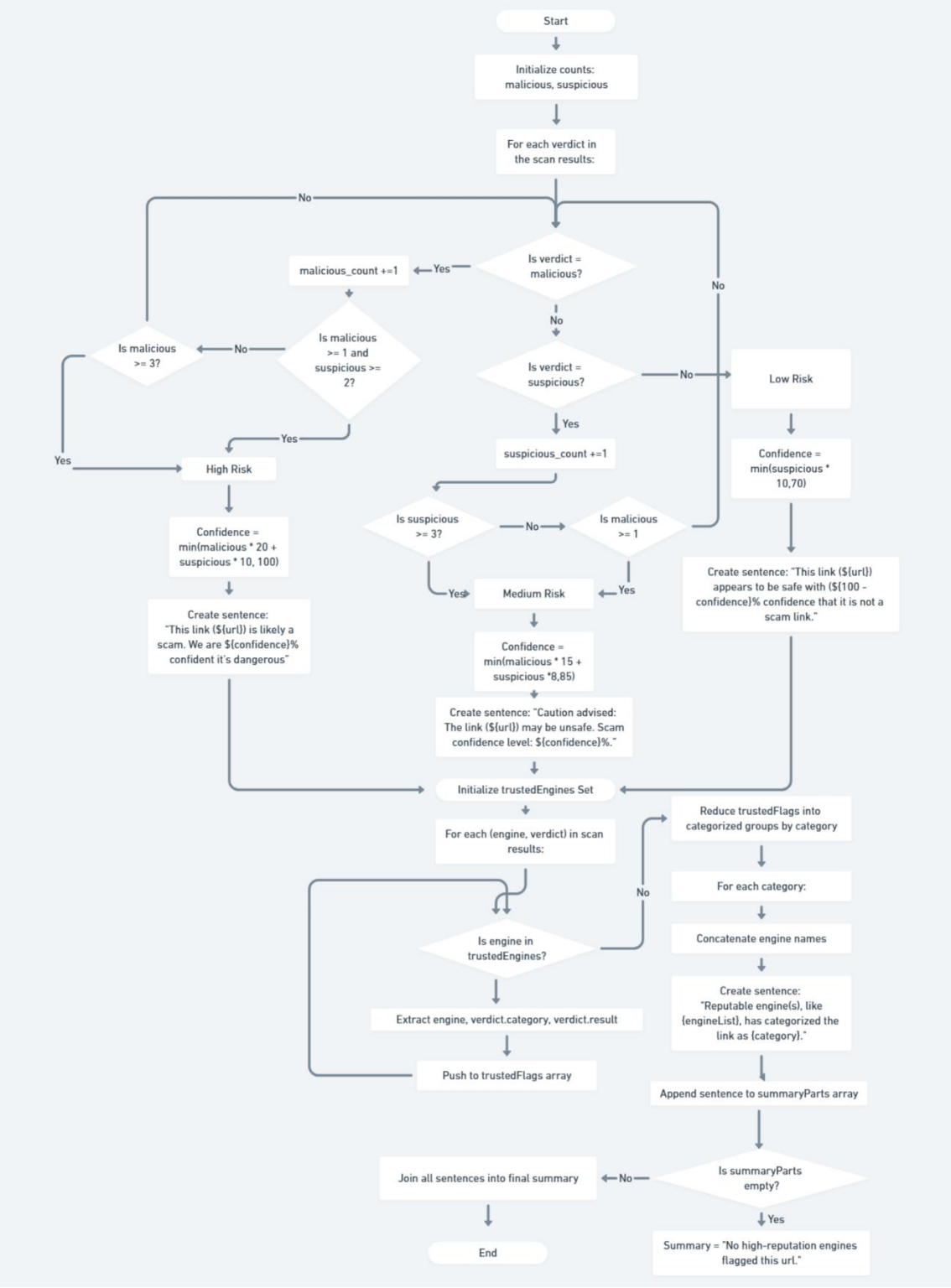


Figure 4.3: Flowchart of Rule-Based Heuristic Used in InQueries

The generated output—composed of risk level, confidence score, and trusted explanation—is then passed into the LLM chain as the context for simplified summary generation. The detailed logic and scoring rules behind this heuristic are described in Section 5.4.

4.7 Interface Design

The scam link detection system is embedded directly within Telegram and does not feature a separate graphical user interface (GUI). Instead, it leverages Telegram's built-in message formatting capabilities to create a lightweight, intuitive interaction model. The chatbot responds conversationally to user input and presents security evaluations in natural language, supported by simplified threat summaries and safety recommendations.

This section presents two representative interaction scenarios to illustrate how the chatbot interface behaves across different use cases.

4.7.1 Scenario 1: User Query About General Cybersecurity Knowledge

In this scenario, the user initiates a conversation with a general question not involving any URL—specifically, "What is phishing traps?" The system interprets this as an open-ended knowledge request and responds using the LLM's factual generation capabilities.

As shown in Figure 4.3, the bot replies with a simple yet concise explanation. The tone is clear and cautious, aligned with the bot's security-first design.

This interaction demonstrates the chatbot's ability to engage in general educational dialogue, independent of URL analysis. It helps build user trust by offering helpful, on-demand cybersecurity knowledge without requiring users to leave the messaging interface.

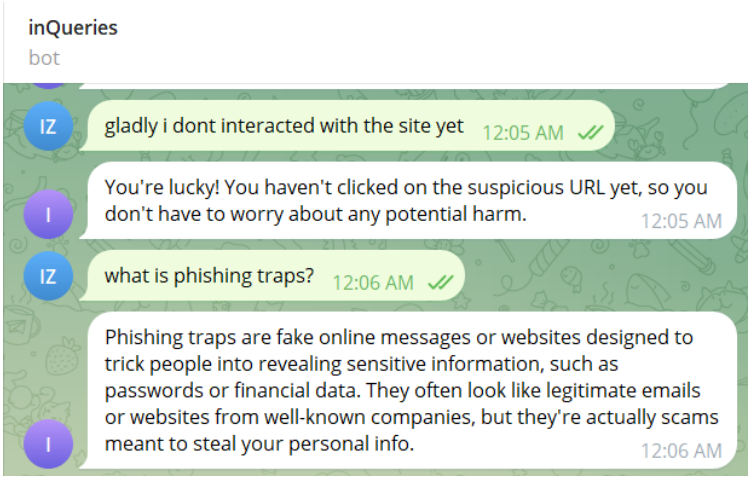


Figure 4.4: General query response; the bot explains in simple yet concise when asked, “What is phishing?”

4.7.2 Scenario 2: User Submits or Asks About a Suspicious URL

The second scenario involves the bot’s core functionality: analyzing and explaining the risk level of a user-submitted URL. The interaction typically begins when a user either shares a link directly or expresses intent to verify the safety of a link.

As shown in Figure 4.4, the bot begins the conversation by asking the user to share the suspicious URL. The tone is friendly but cautious, reminding users to be careful with unknown links.

In Figure 4.5, the user asks for clarification about how the system checks the link’s safety. The bot responds with a concise overview of its methodology, explaining that it uses VirusTotal and urlscan.io as the primary analysis tools. It reassures the user that it doesn’t click the link itself and will only review metadata from trusted sources.

Once the user provides the link, as illustrated in Figure 4.6, the bot performs a full analysis using the integrated APIs and returns a structured explanation. The message includes:

- The overall risk rating (e.g., “High Risk with 100% confidence”)
- Supporting metadata (e.g., antivirus detection counts, verdicts from engines like Fortinet and Sophos)
- A concise summary of why the link is dangerous
- Encouraging safety instructions or follow-up prompts

This sequence clearly demonstrates the chatbot’s ability to analyze threats, convey results in plain language, and create an interactive dialogue where users can ask for clarification, tools used, or expected response time (as seen in Figure 4.5). Even when queried about response latency, the bot provides a helpful and contextualized answer.

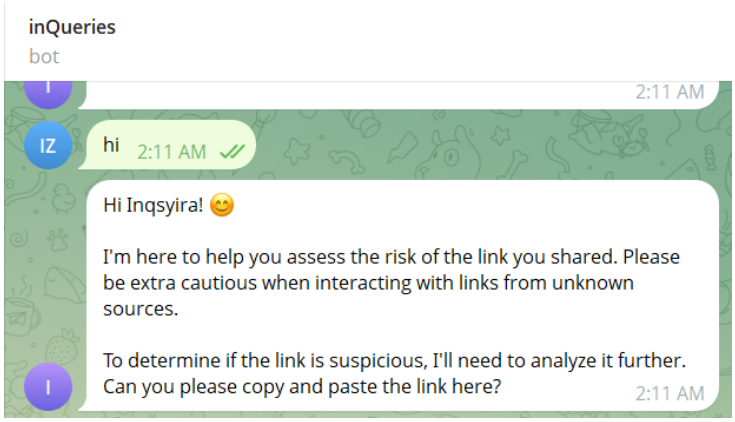


Figure 4.5: The bot welcomes the user and prompts them to submit a suspicious link for analysis.

Figure 4.6: The bot explains its approach, tools used (VirusTotal, urlscan.io) and responds to follow-up questions.

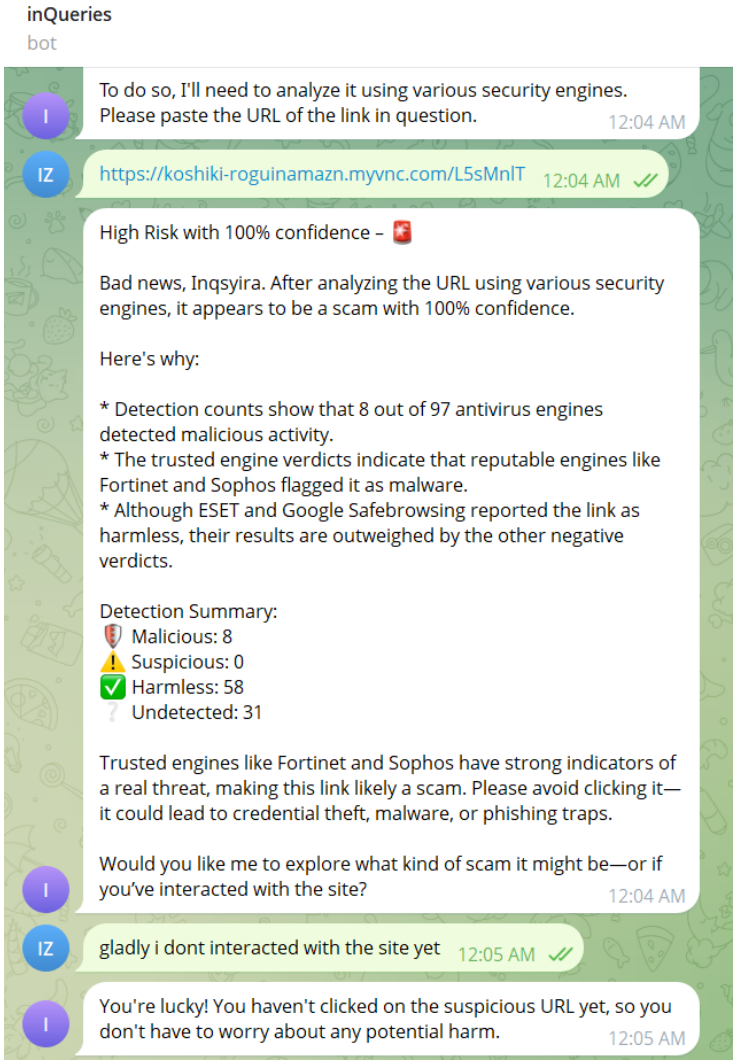


Figure 4.7: High-risk URL analysis output showing detection counts and trusted engine verdicts.

4.7.3 Summary of Interaction Design

These examples demonstrate that the interface is intentionally minimal yet expressive. By leveraging Telegram-native formatting, emojis, and natural language responses, the chatbot creates a familiar and reassuring experience—especially in high-stress or uncertain situations.

Chapter 5: Implementation

5.1 Development Environment Setup

This section presents the technical foundation for implementing the system. Every component in the stack is selected with a deliberate focus on zero-cost deployment and open-source technologies. The environment supports rapid prototyping and testing while enabling a clear path to future scaling.

5.1.1 Tools and Platforms Used

Table 5.1 outlines the tools and platforms used in constructing the system. Each tool serves a specific purpose across the pipeline, ranging from data retrieval and inference to message routing and data logging.

Table 5.1: Tools and Platforms Used

Tool / Platform	Purpose
Docker	Containerization of components for reproducibility and deployment.
n8n	Visual workflow automation and logic orchestration
PostgreSQL	Persistent storage for n8n workflow states and credentials
Ollama	Local hosting of LLMs for both embedding and generation
Pinecone	Vector database for semantic retrieval of embedded scan metadata
Ngrok	Secure tunnel for exposing local endpoints (Telegram webhook)
Telegram Bot API	End-user chat interface
VirusTotal API	External source of threat verdicts from security engines
urlscan.io API	Returns behavioral and structural link metadata
Google Sheets API	Lightweight data logging and CSV-like reporting

These tools collectively enable a fully local, end-to-end pipeline that integrates external intelligence into explainable AI outputs.

5.1.2 Programming Languages and Libraries

While most logic is implemented through visual nodes in n8n, several components require custom scripting for data extraction, decision-making, and transformation. Table 5.2 outlines the programming languages and libraries used within the low-code workflow environment.

Table 5.2: Programming Languages and Libraries

Language / Library	Purpose
Python	Extracts URLs and IOCs from messages using ioc-finder
JavaScript	Implements risk scoring, API response handling, and logic flows
JSONPath / JS Regex	Used for parsing nested fields in API outputs
Shell scripts	Bootstraps Docker containers and fetches LLM models
YAML (Docker Compose)	Configures orchestration across containers

5.1.3 Software Stack and Version Details

Table 5.3 lists version-specific details for each major component used. These selections ensure reproducibility and compatibility across both CPU and GPU environments.

Table 5.3: Software Stack and Version Details

Component	Version / Image
n8n	n8nio/n8n:latest
Ollama	ollama/ollama:latest
PostgreSQL	postgres:16-alpine
Qdrant	qdrant/qdrant:latest
Ngrok (external)	Tunnel endpoint for webhook URL
LLM pulled	llama3.2 (via ollama pull llama3.2)

The system also supports multiple deployment profiles including:

- CPU-only profile (ollama-cpu)
- NVIDIA GPU profile (ollama-gpu)
- AMD GPU ROCm profile (ollama-gpu-amd)

This flexibility makes the system portable across personal laptops and more capable compute nodes.

5.2 System Construction in n8n

The core system logic is fully implemented using n8n across two modular workflows: the front-end interaction layer and the backend scan processor.

The frontend workflow manages Telegram communication, language detection, URL identification, and conditional routing. It can detect if a link has already been scanned by checking a Google Sheets log. If it is new, the system triggers the backend workflow for analysis; otherwise, it directly passes results to the Pinecone for retrieval, which then forwards the relevant context to the LLM Chain for augmented response generation.

The backend workflow handles real-time API queries, risk classification, embedding, and storage. The two workflows are connected through an Execute Workflow node, allowing for clean modularity and shared state.

5.3 Threat Intelligence Integration

The threat intelligence layer is central to the system's factual accuracy. It provides the raw evidence used for downstream heuristic classification, embedding, and AI explanation. This section outlines what is retrieved from each external source—VirusTotal and urlscan.io—and how that data is structured for further processing.

5.3.1 VirusTotal Analysis Extraction

The system queries VirusTotal in two stages: first with a POST request to submit a new URL, and then a GET request using the returned scan ID to retrieve

the final analysis report. This report includes verdicts from over 60 engines, aggregated statistics, detection metadata, and supporting links for further investigation.

The core extracted data elements include:

- **Engine Verdicts**:
  The results object contains scan outcomes from each participating engine. Each entry includes:
  - engine_name: The name of the scanning engine (e.g., BitDefender, Kaspersky)
  - method: Typically “blacklist” for URL scans
  - category: One of malicious, suspicious, harmless, or undetected
  - result: Free-text descriptor (e.g., "clean", "phishing", "malware")

These entries are parsed and grouped by category to determine which engines contributed to the classification. Special attention is given to high-reputation vendors (see Section 5.4.2), which are filtered and summarized separately for token-efficient embedding.

- **Statistical Summary**:
  The stats object provides an aggregate view:
  - malicious: Number of engines that labeled the URL as malicious
  - suspicious: Number that marked it suspicious
  - harmless, undetected, timeout: Other outcome counts

This object forms the basis of the rule-based risk classification algorithm, where thresholds are defined for each risk level and a corresponding confidence score is computed (see Section 5.4.1).

- **Scan Metadata**:
  Other supporting attributes include:
  - date: UNIX timestamp of the scan, used for temporal analysis and context
  - id: Scan identifier used as the filterbyScanID metadata in Pinecone
  - links.item: Direct URL to the VirusTotal report page
  - meta.url_info.url: The raw URL that was analyzed

All of the above data is normalized and passed through a Data Transform Node, which restructures the result into a JSON format optimized for embedding. The output includes both factual attributes (counts, verdicts) and a hardcoded natural language summary used later in LLM-based explanation.

5.3.2 Summary Table: VirusTotal Data Extraction

The following Table 5.4 summarizes the key fields extracted from the VirusTotal response and their usage within the system.

Table 5.4: VirusTotal API Response Data Usage

Extracted Field	JSON Path	Purpose in System
Engine verdicts	data.attributes.results	Supports per-engine filtering and trusted summarization
Malicious / Suspicious count	data.attributes.stats	Used for risk level scoring logic
Analyzed URL	meta.url_info.url	Used for display, embedding, and Pinecone metadata
Scan ID	data.id	Used as filterbyScanID key for metadata-based retrieval

This detailed and structured data capture from VirusTotal ensures that every downstream module—including risk classification, vector embedding, and AI summarization—is grounded in verifiable third-party intelligence. The

emphasis on trusted engine filtering also mitigates noise from unknown or unverified scanners, which is critical in open-source multi-engine environments.

The following section outlines how urlscan.io complements the VirusTotal metadata by providing behavioral and structural insights. Together, these two sources establish a comprehensive foundation for evidence-based risk assessment.

5.3.3 urlscan.io Metadata Integration

While VirusTotal provides categorical verdicts from antivirus engines, urlscan.io complements the analysis by offering structural and behavioral metadata for the submitted URL. This includes data about the webpage layout, embedded scripts, trackers, redirects, domain infrastructure, and more. However, due to the high dimensionality and verbose nature of the urlscan.io response, only a minimal subset of fields is retained for vector embedding. This is a strategic decision made to comply with token limits during upsert operations into the Pinecone vector database.

The full response from urlscan.io is hierarchically structured across several top-level keys, including data, lists, page, task, verdicts, and stats. These objects encapsulate detailed observations about the scanned page, such as:

- IP address mappings
- Redirect chains
- Domain-level scripts
- TLS certificate fingerprints
- Embedded resource domains
- Visual representation of the rendered page

Although these fields offer rich context, only two core fields are extracted for downstream use:

1. **reportURL**

   Path: task.reportURL

   This field provides a direct link to the full scan result hosted by urlscan.io. It serves as a human-readable reference, allowing users or analysts to explore the full behavioral footprint of the scanned URL.

2. **screenshotURL**

   Path: task.screenshotURL

   This field provides a static image of the rendered webpage at the time of the scan. This visual cue can help users confirm the legitimacy of the page based on appearance and design, especially in cases involving phishing or impersonation.

The use of these two fields ensures that the system preserves visual and forensic transparency without exceeding token limits during vector upsert. The reportURL and screenshotURL is made available for user inspection via clickable links in the Telegram interface.

The full urlscan.io response tree is shown below in a flattened format for reference.

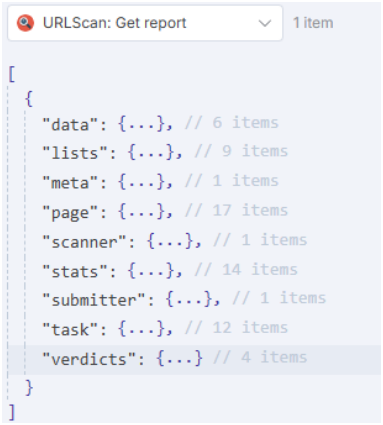


Figure 5.1: Flattened structure of urlscan.io scan report (top-level keys)

Although only the task object is used directly, the entire response is downloaded and logged internally in case future extensions are needed, for example, extracting tracker lists or redirect chains.

In the proposed design, urlscan.io would complement VirusTotal by contributing render-time visual and structural intelligence, enhancing the system's evidence base for phishing detection and suspicious content profiling. This dual-source approach is intended to combine machine-derived threat verdicts with contextual web behavior, enriching the LLM's semantic memory. However, in the current implementation, only selected metadata fields are extracted from urlscan.io due to token limitations during vector upserts.

5.3.4 Asynchronous Processing Strategy

Both VirusTotal and urlscan.io require asynchronous handling due to the inherent latency of live threat scans. Without proper delay management, the system would risk fetching incomplete or null data, which could undermine risk scoring accuracy and break downstream logic. To address this, the backend workflow incorporates a multi-phase waiting and validation strategy using a combination of Wait and conditional If nodes.

VirusTotal Scan Timing Handling

VirusTotal's scan API provides an analysis ID immediately after submission, but the full verdict data—especially from all scanning engines—may not be available instantly. To accommodate this latency, the system applies a two-stage delay mechanism along with a result readiness check:

1. Initial Delay: A 15-second pause is introduced after the URL is submitted (via POST) to allow the scan to initialize.
2. Early Check: A GET request is then made to retrieve the scan results (VT Report1). The workflow proceeds to an If node that evaluates whether the status field in the response equals "completed".

3. Fallback Delay: If the status is not yet complete (i.e., still "queued" or "running"), an additional 5-second delay is applied, followed by a second GET request (VT Report2) to retrieve the updated results.

This logic is visualized in Figure 5.2, which shows the full VirusTotal scan handling pipeline. The fallback logic ensures that even when scans are slow to finish, the workflow dynamically waits rather than proceeding prematurely.

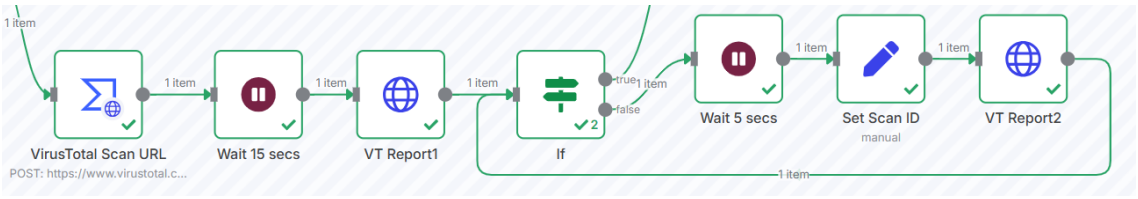


Figure 5.2: n8n workflow segment for VirusTotal asynchronous handling

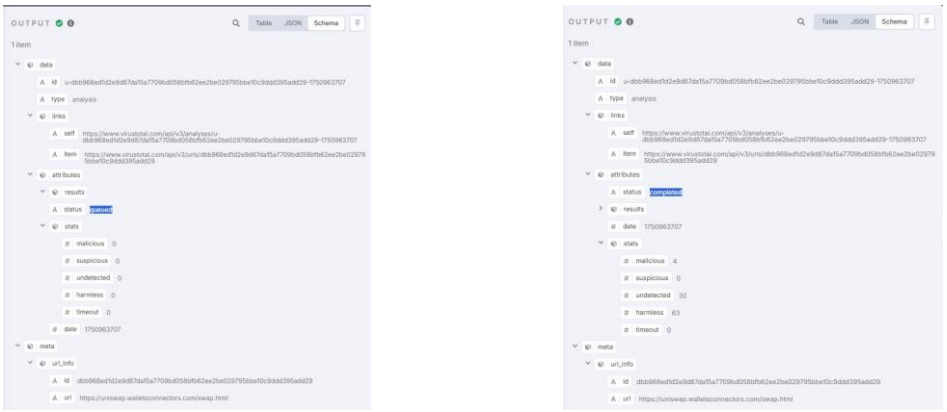


Figure 5.3: Comparison between incomplete (status: queued) and complete (status: completed) VirusTotal responses.

urlscan.io Scan Submission and Handling

Unlike VirusTotal, urlscan.io does not offer a direct mechanism to query the scan status after submission. This limitation requires a different handling approach. The system addresses this by implementing input validation

immediately after submission and relying on a fixed waiting strategy only when the input is confirmed to be valid.

Upon submitting a URL to urlscan.io, the API responds with one of two possible outcomes:

- A valid response, containing fields such as scanId, result, and api. This confirms that the URL was accepted for scanning.
- An error object, typically returning the message "Bad request - please check your parameters". This indicates the URL was rejected and will not be scanned.

To manage this behavior, the system evaluates the submission response before proceeding further. If the response contains a scan ID and result URL, it is considered valid. The workflow then continues by applying a fixed 40-second wait, allowing the scan to be completed in the background.

However, if the submission fails and an error message is returned, the system bypasses the wait and exits the scan flow entirely. No further API calls are made, and the urlscan.io branch is skipped. This decision avoids unnecessary delays and prevents the pipeline from attempting to retrieve a report that will never exist.

This preemptive filtering is visualized in the figures below:

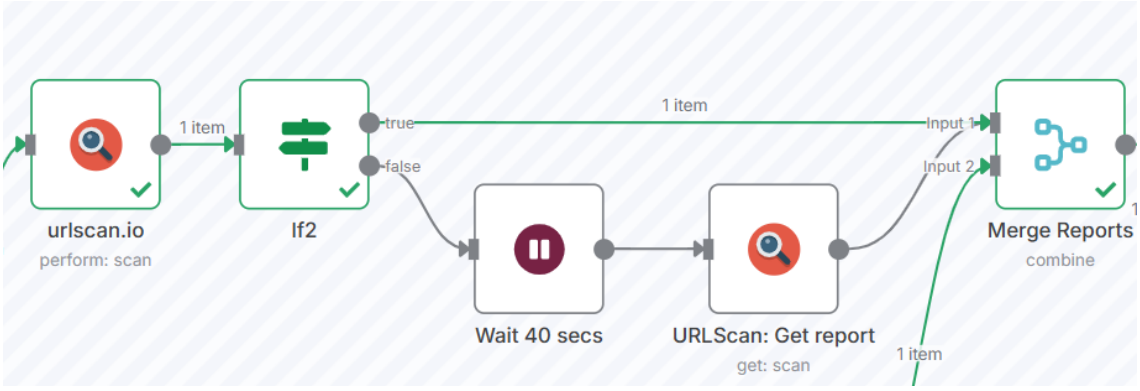


Figure 5.4: If2 Node checks and separates valid and invalid scan submissions.

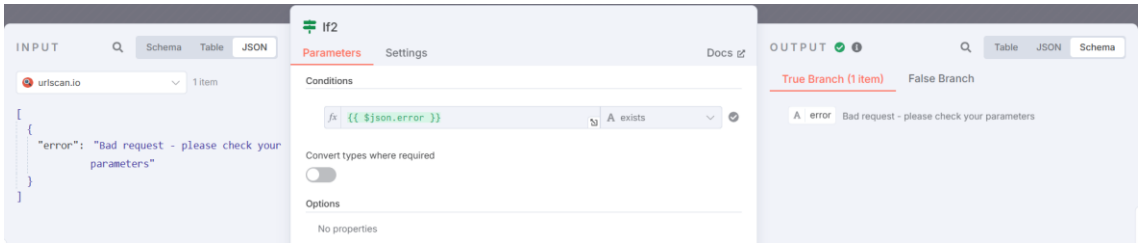


Figure 5.5: Detailed breakdown of the If2 Node: Invalid Scan

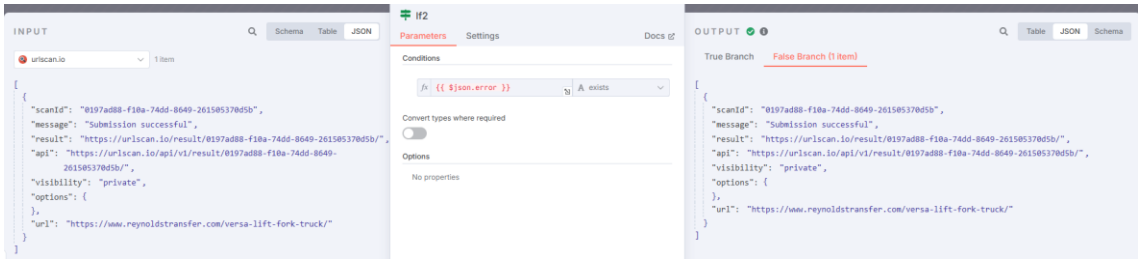


Figure 5.6: Detailed breakdown of the If2 Node: Valid Scan

It is important to note that no logic is applied after the 40-second wait. The system does not verify the presence or quality of the report following the scan, nor does it validate the screenshot or metadata fields at that stage. This reflects a conscious design choice: urlscan.io's limitations are acknowledged and accounted for at the entry point, not after the scan.

5.4 Heuristic Risk Classification Logic

The classification logic in this project relies on a rule-based scoring algorithm designed to interpret results from external threat intelligence APIs. This mechanism serves as a transparent, deterministic layer that categorizes the risk level of a scanned URL, while also providing a confidence score. The classification is implemented using JavaScript within a Code node inside the n8n workflow.

5.4.1 Logic Overview

At the core of this logic is a function that evaluates two key indicators obtained from VirusTotal: the number of malicious and suspicious verdicts. Based on these counts, the system assigns one of three risk categories—High, Medium, or Low—along with a computed confidence score. The complete function is shown below:

```
function classifyRisk({ malicious = 0, suspicious = 0 }) {
  if (malicious >= 3 || (malicious >= 1 && suspicious >= 2))
    return { risk: "High Risk", confidence: Math.min(malicious * 20 + suspicious *
10, 100) };
  if (malicious >= 1 || suspicious >= 3)
    return { risk: "Medium Risk", confidence: Math.min(malicious * 15 +
suspicious * 8, 85) };
  return { risk: "Low Risk", confidence: Math.min((suspicious || 0) * 10, 70) };
}
```

This function returns both a risk label and a numerical confidence score, the latter of which is capped at predefined thresholds to avoid inflation. The logic does not apply weight based on the reputation of specific antivirus engines, although it does incorporate engine-specific filtering for downstream summary generation (see Section 5.4.2).

To support the semantic embedding process, the classification result is also converted into a hardcoded natural-language message. This improves dense vector similarity in Pinecone during retrieval, allowing the LLM to later produce more fluent and context-aware explanations.

The text message generation function is defined as follows:

```
// Hardcoded text message to support dense similarity
function renderMessage(url, risk, confidence) {
  switch (risk) {
    case "High Risk":
      return `⚠️ This link (${url}) is likely a scam. We are ${confidence}% confident it's dangerous.`;
    case "Medium Risk":
      return `⚠️ Caution advised: The link (${url}) may be unsafe. Scam confidence level: ${confidence}%.`;
    case "Low Risk":
      return `✅ This link (${url}) appears to be safe with (${100 - confidence}% confidence that it is not a scam link.`;
    default:
      return `All clean🔍! The link (${url}) shows no threat signals.`;
  }
}
```

This summary serves both human readability and AI model comprehension. It ensures the embedded text has a clear structure and risk orientation, which supports effective semantic retrieval when user queries are processed later by the LLM Chain.

5.4.2 Trusted Engine List

While the classification logic itself treats all antivirus verdicts equally, the system includes a predefined list of high-reputation engines for the purpose of generating more meaningful summaries. These engines are known for their reliability and accuracy and are used to filter the full set of scan results into a more trusted subset.

The filtering logic is implemented as follows:

```
const trustedEngines = new Set([
  "Microsoft", "Google Safebrowsing", "Kaspersky", "Bitdefender", "ESET",
  "Sophos", "Fortinet", "McAfee", "TrendMicro", "Symantec", "CrowdStrike",
  "Cisco Talos", "Palo Alto Networks", "Check Point", "F-Secure"
]);

// Extract categorized verdicts by trusted engines
const trustedFlags = Object.entries(results)
  .filter(([engine, verdict]) => trustedEngines.has(engine))
  .map(([engine, verdict]) => ({
    engine,
    category: verdict.category,
    result: verdict.result
  }));
```

The filtered verdicts are then grouped by risk category (e.g., malicious, harmless), and composed into plain-text explanations. This provides a more interpretable and compact summary for users while also minimizing token usage during embedding.

```
// Group by category
const categorized = trustedFlags.reduce((acc, { engine, category }) => {
  if (!acc[category]) acc[category] = [];
  acc[category].push(engine);
  return acc;
}, {});

// Compose readable sentences
const summaryParts = Object.entries(categorized).map(([category, engines]) =>
{
  const engineList = engines.map(e => e.replace(/ Safebrowsing/, "")).join(", ");
  return `Reputable engine(s), like ${engineList}, has categorized the link as
${category}.`;
});

const summary = summaryParts.length > 0
  ? summaryParts.join(" ")
  : "No high-reputation engines flagged this url.";
```

These trusted summaries are embedded along with the risk classification in the final data object. The full structured output is returned by the Data Transform Node, and includes metadata such as detection counts, participating engine list, and scan date. The structure of this output is shown below:

```
return [{
  json: {
    title: message,
    filterbyScanID: scanId,
    detection_counts: stats,
    risk_level: risk,
    confidence_score: confidence,
    participating_partners: totalScanners,
    trusted_engine_verdicts: trustedFlags,
    trusted_summary: `${summary} The link was scanned by ${totalEngines}
antivirus engines on ${scanDate}. Full report of the scan can be access here at
` + `${task?.reportURL}\n` + `The screenshot of the page can be access here at
` + `${task?.screenshotURL}`
  }
}];
```

This output is then embedded using the local LLM and stored in Pinecone for future retrieval. It forms the core of the explanation layer used by the AI agent during user interaction. Figure 5.7 below demonstrates the text formatting designed to support LLM comprehension during response generation.

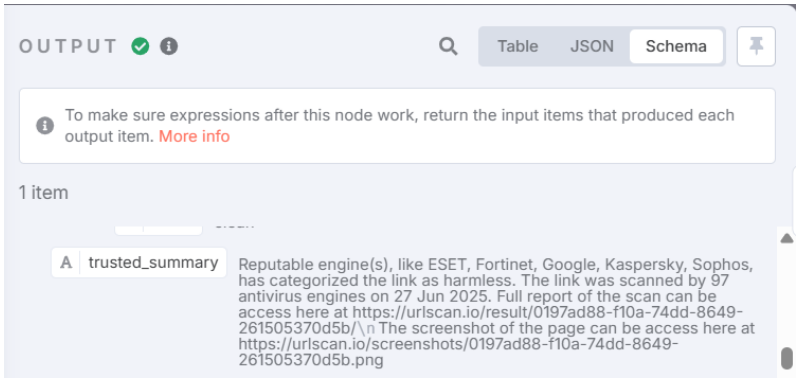


Figure 5.7: Part of output from the Data Transform Node.

5.5 Semantic Embedding and Vector Storage Configuration

After the risk classification and summary generation stages, the structured output is prepared for semantic embedding and long-term vector storage. This step is essential for enabling RAG, allowing the LLM to retrieve relevant past entries based on semantic similarity, rather than basic keyword matching.

To perform this embedding, the system uses a local embedding model—mxbai-embed-large—hosted via Ollama, which converts textual data into 1024-dimensional vector representations. These vectors are then inserted into Pinecone, a cloud-based vector database optimized for fast and scalable similarity search using cosine distance.

The embedded content includes not only the risk labels and numeric statistics but also a human-readable summary string. These summaries are intentionally crafted to be semantically rich and interpretable by LLMs, enabling accurate matching against natural language queries during retrieval.

Figure 5.6 shows the actual Pinecone index configuration used in this project. The index—inqueries-final—runs on a serverless deployment in AWS us-east-1, using Dense capacity mode, with a metric set to cosine and dimensionality fixed at 1024.

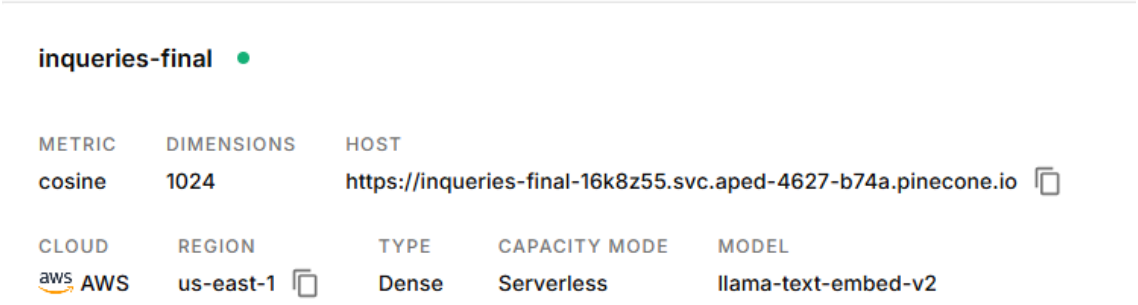


Figure 5.8: Index Configuration on Pinecone Storage

Note: Although the Pinecone dashboard displays llama-text-embed-v2 as the model label, all embeddings in this implementation were generated using mxbai-embed-large via a local Ollama instance.

5.5.1 JSON Flattening and Sentence Enrichment

Before data is embedded, a Flatten nested JSON node is used to convert structured fields into full sentences. This is a critical step because embedding models such as mxbai-embed-large perform better on natural text rather than key-value pairs or symbolic representations. The transformation converts fields like risk_level: High Risk into expressive phrases such as outline in Figure 5.9 below.

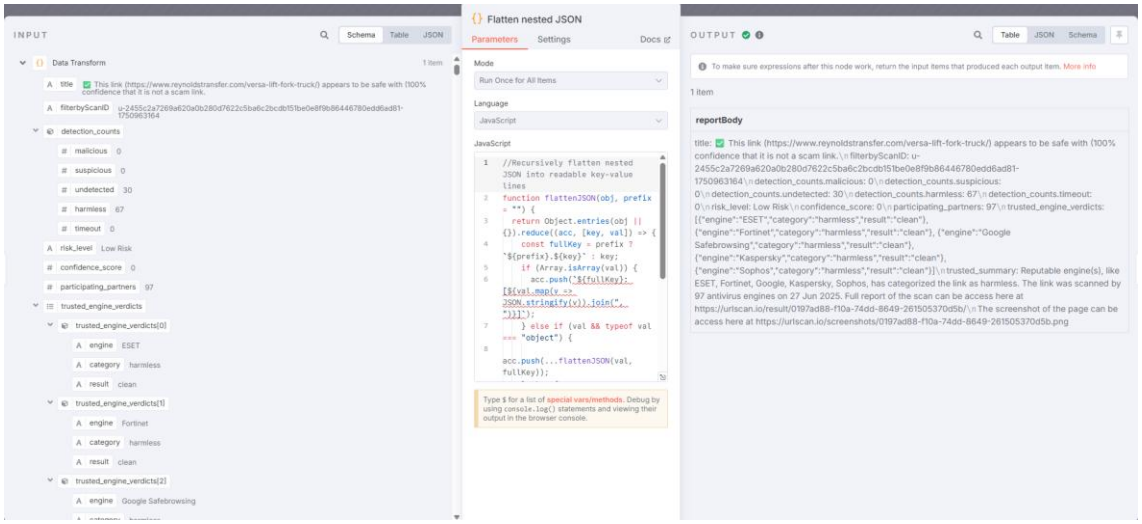


Figure 5.9: Output from the Flatten nested JSON node showing enriched, sentence-level content designed for optimal embedding performance.

5.5.2 Vector Storage Metadata and Indexing

Once embedded, the resulting vectors are stored in Pinecone, a scalable vector database optimized for semantic retrieval. Each vector is inserted along with associated metadata that allows for fine-grained filtering and traceability. These metadata fields play a critical role in supporting both semantic similarity and exact-match lookups by the AI agent during inference.

Specifically, each vector chunk is stored with the following metadata keys:

- exact_url: The raw URL that was scanned.
- filterbyScanID: The unique scan identifier obtained from VirusTotal.
- risk_level: A label categorizing the threat level (e.g., High, Medium, Low).
- trusted_summary: The full sentence-level summary generated from trusted antivirus engines.
- title: The system-generated risk message used in chatbot outputs.

The field filterbyScanID serves as a critical retrieval key, allowing the system to match user queries directly with previously scanned URLs, bypassing similarity search when exact matches are available. Risk level labels, on the other hand, are used to namespace the vectors and enable scoped retrieval within known threat categories.

This storage strategy is implemented within a dedicated Pinecone node in the n8n workflow. As shown in Figure 5.10, the configuration includes both the upsert parameters and a preview of the metadata fields that are inserted alongside each vector. This visual output confirms that the stored data is both semantically rich and structurally aligned for AI reasoning.

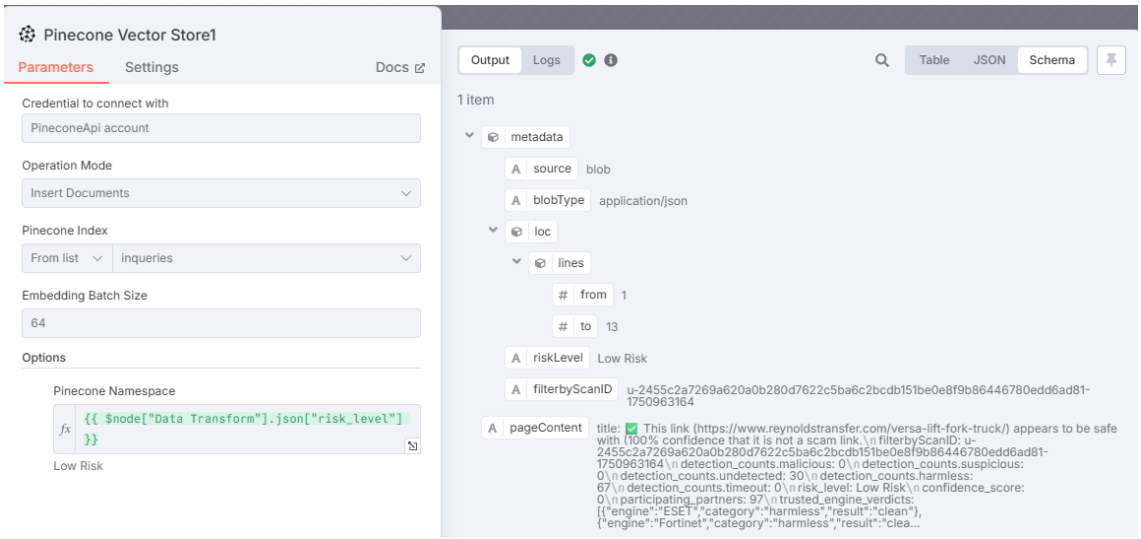


Figure 5.10: Pinecone Vector Store configuration in n8n, showing the upsert operation with metadata attached.

The captured output as demonstrated in Figure 5.7 confirms successful insertion of fields such as filterbyScanID, risk_level, and trusted_summary, which are essential for structured retrieval and downstream response generation.

5.5.3 Embedding Constraints with Nested Threat Intelligence Data

A key challenge emerged during the embedding stage: the system's input data — primarily responses from VirusTotal and urlscan.io — consisted of deeply nested JSON structures that encode symbolic, highly structured threat intelligence. However, the embedding approach used in this project relies on dense vector similarity (via cosine distance) using mxbai-embed-large, which is optimized for semantically rich, narrative-style text.

When raw or minimally flattened JSON was passed directly into the embedding model, the resulting vectors often lacked meaningful semantic representation. This mismatch degraded the quality of similarity retrieval in Pinecone and, by extension, reduced the precision and relevance of AI-generated explanations in some cases.

To overcome this, the system applied a manual reformatting strategy:

- Flattening key metadata into natural-language summaries
- Enriching the content with trusted engine verdicts and plain-English scoring
- Hardcoding explanations that retained semantic density for vector search

This solution improved RAG behavior but introduced a dependency on prompt phrasing and sentence structuring, which may not scale well across diverse input formats or edge cases. It also may place cognitive load on the system designer to anticipate and compress security-specific metadata into textual formats suitable for embedding.

Future improvements may involve experimenting with:

- Using a hybrid search method that combines keyword matching (sparse retrieval) with vector similarity, so the system can better handle both structured fields and semantic text.
- Exploring retrieval systems that handle structured data more naturally, such as tools that index fields like “malicious count” or “engine verdicts” directly.
- Trying embedding models trained specifically on cybersecurity or phishing content, which may better understand threat indicators and structured reports without needing manual rewriting.

5.6 LLM Chain Setup and Prompt Engineering

5.6.1 Overview of the LLM Chain

The implementation of the LLM Chain in this project uses a LLM hosted through the Ollama runtime. Specifically, the model used is llama3.2:latest, which operates in a zero-cloud environment to preserve user data privacy and reduce latency. The LLM Chain was developed within the n8n automation platform using LangChain nodes, which allow seamless integration of embedding, retrieval, and language model capabilities in a visual pipeline.

The primary function of this chain is to accept user queries, enrich them with contextual information retrieved from a vector database (Pinecone), and generate a well-structured, explainable response. This setup follows the RAG paradigm, where a user-submitted suspicious URL is first matched with relevant vectorized scan records before being passed to the LLM for final judgment. The overall aim is to balance detection accuracy with user interpretability.

5.6.2 Prompt Structure

The LLM in this system is prompted using a structured and dynamic template. The final prompt sent to the model includes four components:

1. A system prompt defining the chatbot role and response behavior.
2. A fixed instruction block guiding step-by-step reasoning and response formatting.
3. The retrieved context, typically consisting of scan summaries (e.g., detection counts, confidence scores).
4. The user's exact query, which includes the suspicious URL and user identity (first name).

The instructions specifically guide the model to:

- Analyze the scan report starting from total detections, presence of trusted sources, and contextual significance.
- Output a risk summary (e.g., "High Risk – 4 malicious (including trusted engines) ") as a header.
- Use emoji markers and a markdown-style layout to make the summary more readable and visually informative.
- Provide direct links to the original scan report and its screenshot.

This formatting not only standardizes the response but also improves its explainability for non-technical users. This formatting not only standardizes the response but also improves its explainability for non-technical users. The final prompt template used in this project, including the exact structure and injected variables, is documented in Appendix D.1: Prompt Template Samples for reference.

5.6.3 Embedding and Retrieval

To retrieve relevant context, each document (scan record) is first embedded using mxbai-embed-large, a multilingual embedding model compatible with Ollama. The embedded vectors are stored in Pinecone, where similarity search is performed based on the suspicious URL provided by the user.

During retrieval, the top-1 most similar document is fetched from the vector index, optionally filtered by riskLevel and scanID. These filters are introduced to ensure contextual relevance and prevent false matches from unrelated past records. This retrieved content is then passed into the prompt composition stage, where it is merged into the full query before being sent to the LLM.

5.6.4 Explainability via Prompt Engineering

The design of the LLM prompt aims to make the model's reasoning process transparent. Rather than simply returning a label (e.g., "this is a scam"), the response includes:

- The detailed count of malicious, suspicious, harmless, and undetected results.
- Whether any trusted detection engines flagged the link.
- The associated confidence score and scan ID.
- A short explanation of the reasoning behind the verdict.

Additionally, the response always includes direct links to the scan report and screenshot on urlscan.io. This supports traceability, allowing users to independently verify the judgment. To further improve explainability and reproducibility, the similarity score between the user's query and the retrieved document is logged during each detection and recorded within the Pinecone dashboard for monitoring and analysis.These values are analyzed further in Chapter 6 during testing.

5.6.5 Limitations

Due to the nature of the local LLM setup, this project did not implement extensive prompt tuning or parameter optimization. While multiple versions of the prompt were tested during development, these iterations were not systematically evaluated or benchmarked. Furthermore, unlike GPT-based models that benefit

from public prompt engineering studies, this system uses an offline model with limited public reference, which restricts tuning transferability.

Error handling features such as fallback prompts, alternative retrieval logic, or response validation were also not implemented. Cases such as no retrieval match from Pinecone, incomplete context injection, or irrelevant input queries were not explored due to time constraints and prioritization of other objectives.

5.7 Deployment and Hosting Configuration

5.7.1 Docker-Based Deployment

All services are containerized using Docker Compose under a shared bridge network (demo). The containers communicate over a user-defined bridge network named demo, which Docker Compose automatically registers as self-hosted-ai-starter-kit_demo based on the project directory name as shown in Figure 5.11. This isolated network ensures secure internal communication between services such as n8n, PostgreSQL, Qdrant, and Ollama.

```
PS D:\Degree\Year3\FYP2\self-hosted-ai-starter-kit> docker network inspect self-hosted-ai-starter-kit_demo
[
    {
        "Name": "self-hosted-ai-starter-kit_demo",
        "Id": "fb23c6fdcbfcd6bfc4aec7308c71b4225431b69e925f5898d9cda89641f10570",
        "Created": "2025-05-25T18:28:21.984412369Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {
            "com.docker.compose.config-hash": "49d79f971e7f023ab8b72837a50872bee954cc8e8cd4f7825b5e91ee194286da",
            "com.docker.compose.network": "demo",
            "com.docker.compose.project": "self-hosted-ai-starter-kit",
            "com.docker.compose.version": "2.34.0"
        }
    }
]
```

Figure 5.11: Docker Network Inspection

The following table Table 5.5 summarizes the exposed ports configured in the Docker Compose file:

Table 5.5: Port Configuration

Service	**Port (Host:Container)**	**Purpose**
n8n	5678:5678	Access to the n8n editor interface and webhooks
Ollama	11434:11434	API access for LLM embedding and generation
Qdrant	6333:6333	(Early testing only)
PostgreSQL	Internal only	Used by n8n for storage

Figure 5.12 below captures detailed configuration on docker. The blue marked one is all the containers necessary to run this project.

	Name	Container ID	Image	Port(s)	CPU (%)	Last started
☐	elegant_babbage	05bbd11a13a1	inq-docker		N/A	2 months ago
☑	self-hosted-ai-starter-kit	-	-	-	N/A	3 days ago
☑	n8n	7b5a281c664e	n8nio/n8n:latest	5678:5678	N/A	3 days ago
☑	n8n-import	b138a797e102	n8nio/n8n:latest		N/A	3 days ago
☑	ollama-pull-llama	d7e3dbe30071	ollama/ollama:latest		N/A	3 days ago
☑	postgres-1	6452eb1a7b2a	postgres:16-alpine		N/A	3 days ago
☑	qdrant	05b8c1b1db70	qdrant/qdrant	6333:6333	N/A	3 days ago
☑	ollama	89737c00c934	ollama/ollama:latest	11434:11434	N/A	3 days ago

Figure 5.12: Docker GUI Containers

All services communicate internally through the demo Docker network, with only necessary ports exposed to the host system. The webhook URL for n8n is tunneled via Ngrok and assigned statistically using free tier basis.

5.7.2 Ngrok Tunneling

The system is exposed to the public internet using Ngrok (Figure 5.13), which creates a secure HTTPS tunnel to the locally hosted n8n webhook endpoints. This allows the Telegram Bot API to communicate with the system for real-time interaction.

```
ngrok - tunnel local ports to public URLs and inspect traffic

USAGE:
ngrok

❤️  ngrok? We're hiring https://ngrok.com/careers

Session Status                online
Account                       Nur Inqsyira Zamri (Plan: Free)
Update                        update available (version 3.23.1, Ctrl-U to update)
Version                       3.22.1
Region                        Asia Pacific (ap)
Latency                       14ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://cunning-totally-wren.ngrok-free.app -> http://localhost:5678
```

Figure 5.13: Ngrok Tunnel

5.7.3 Deployment to Server or Live Environment

At present, the system is deployed and tested using a local development machine, with external access provided by Ngrok tunneling. This setup is suitable for prototyping and user testing, especially under zero-budget constraints.

For permanent deployment, the following steps are recommended:

- Use a cloud VM or container orchestration platform (e.g., Docker Swarm, Kubernetes).
- Replace Ngrok with a reverse proxy such as Nginx behind a domain.
- Secure endpoints with SSL/TLS and add authentication tokens to webhook routes.
- Move environment secrets (e.g., JWT keys, API tokens) into a secure vault or secret manager.

5.9 Challenges Encountered and Solutions

Throughout the system development process, several technical, conceptual, and platform-related challenges emerged. Table 5.6 summarizes key issues and how each was addressed or resolved.

Table 5.6: Challenges Encountered and Solutions

Challenge	**Action Taken**
Sandbox Environment Selection	Explored multiple tools (Defender for Endpoint, VirusTotal, ANY.RUN); selected Hybrid Analysis for free access, but later excluded in favor of real-time URL intelligence only.
Toolchain Migration (Replit to VS Code)	Migrated to VS Code and used Ngrok for HTTP inspection. Later replaced by n8n for modular low-code logic and native debugging.
WhatsApp API and Webhook Misunderstanding	Designed early logic assuming link interception in WhatsApp. Later abandoned due to time and arising platform constraint.
Meta Business Policy Enforcement	WhatsApp API blocked due to account verification. Switched to Telegram, which does not require business registration.
Tool Discovery and Workflow Redesign	Adopted n8n to replace prior scripting approach. Enabled faster, modular logic orchestration.
Telegram Webhook with Docker	Webhook failed inside Docker. Ngrok tunnel used to expose endpoint successfully.
Hybrid Analysis API Integration in n8n	Integrated using documentation, ChatGPT support, and testing. Later excluded due to arising bad request error and decide to shift to URL-based scanning and abandoned sandbox.
Merging and Formatting JSON Data	Attempted DOCX generation via Google Docs API; file too large. Switched to JavaScript-based metadata extraction in n8n.
Embedding Model Mismatch	Initial models incompatible with Pinecone. Resolved by switching to mxbai-embed-large, which supports 1024 dimensional vectors.
Less suitable embedding model	Embedding model used which is mxbai-embed-large with dense vector and cosine similarity is not the ideal approach for the pipeline that pull fresh report in the form of nested JSON as the base knowledge. The impact of this consequence is being reduced by adapting custom code that hardcoded the report into full text explanation to assist the AI in the reasoning and response generation.
Metadata Filtering Limitation	Dynamic filters returned null. Confirmed as n8n platform limitation. Read the Pinecone documentation and try with different metadata that obey the rules set by the n8n and Pinecone.
Ngrok Rate Limiting	Hit free-tier cap on May 29. Waited for June 1 reset.

Figures for selected issues are included in Section 5.9.1.

5.9.1 Visual Snapshots and Debug Artifacts

Figures include visual logs from:

- Ngrok quota exhaustion
- Incorrect vector dimensionality error
- WhatsApp business account rejection
- Early sandbox integration attempts

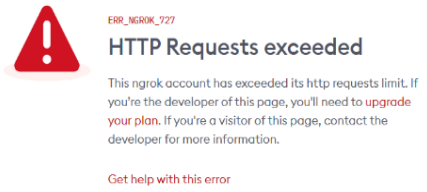


Figure 5.114: Ngrok Rate Limit

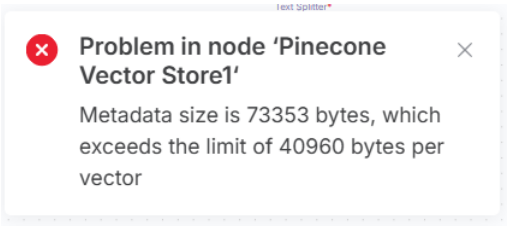


Figure 5.115: Incorrect Vector Dimension

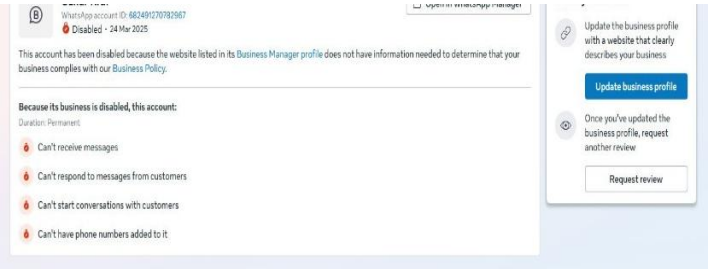


Figure 5.116: Meta Team Flag Unverified Business

Figure 5.117: Sandbox Implementation

Table 5.7: Figure mapping table that breaks down the challenge, action taken, and final status of the debugging.

Challenge	Action Taken	Final Status	Figure Ref
Hardcoded logic and no sandbox integration	Switched to n8n workflow and used VirusTotal & urlscan.io instead of manual logic	Resolved	Fig 5.17
WhatsApp API restriction	Pivoted to Telegram due to Meta's business verification policy	Resolved	Fig 5.16
Pinecone Vector Dimensionality Mismatch	Replaced model with correct 1024-dimension embedding (mxbai-embed-large)	Resolved	Fig 5.15
Ngrok Rate Limiting	Waited for quota reset on free tier	Resolved	Fig 5.14

5.10 Summary

This chapter presented the technical implementation of the scam link detection system, developed using a modular, low-code architecture centered on RAG. The implementation process covered environment configuration, containerized toolchain setup, construction of system workflows in n8n, integration of external threat intelligence sources, and deployment of explainable AI techniques using locally hosted LLM and a cloud-based vector store (Pinecone).

The implementation closely aligns with the system's original design goals, demonstrating feasibility across the following areas:

- A fully automated scam detection pipeline was implemented using only open-source tools and free-tier APIs, enabling zero-cost deployment.
- A transparent, rule-based classification mechanism was developed to assign threat scores without reliance on black-box machine learning models.

- The RAG-based explanation module was successfully integrated, allowing the chatbot to generate contextual, natural language summaries grounded in actual scan metadata, rather than relying on generic templates or static warnings.

These outcomes confirm the viability of combining explainable AI, live threat intelligence, and low-code orchestration to support practical, user-facing cybersecurity tools within messaging environments.

Chapter 6: Testing

6.1 Overview

This chapter presents the evaluation of the inQueries prototype across two primary dimensions: latency and semantic detection accuracy. Testing was conducted in a self-hosted, GPU-accelerated environment using containerized components to simulate a practical, low-cost deployment environment.

The evaluation supports Objective 4, which focuses on assessing the system's practical feasibility and responsiveness under real-world conditions. Rather than relying on rule-based detection or string matching, inQueries employs semantic retrieval through vector similarity. Specifically, user-submitted URLs are embedded locally using the mxbai-embed-large model, and the resulting vectors are queried against a cloud-hosted Pinecone index using cosine similarity.

Each test scenario exercises the complete pipeline—from user input to final system output—allowing for evaluation of both performance and detection quality.

6.2 Test Environment Setup

Table 6.1 outlines the system configuration used during development and testing. The system ran locally on a GPU-enabled machine and relied entirely on open-source tools and free-tier services.

Table 6.1: Testing Environment

Component	Specification
OS	Windows 11 64-bit
Processor	Intel Core i5-12450H (10th Gen, 8-core)
RAM	32 GB DDR4
GPU	NVIDIA GeForce RTX 3050 Laptop GPU (4 GB VRAM), used via gpu-nvidia Docker profile
LLM Models Tested	llama3.2:latest (via local Ollama), models/gemini-1.5-flash (cloud-based)
Embedding Model	mxbai-embed-large:latest (1024-dimensional vectors)
Vector Store	Pinecone (free-tier, cloud-hosted vector database)
Automation Tool	n8n (running inside a Docker container)
API Services	VirusTotal, urlscan.io (free-tier access)
Internet Speed	~100 Mbps download / 8 Mbps upload, with ~30 ms ping, measured using Speedtest.net.

6.3 Latency and Responsiveness Testing

Purpose

To measure the end-to-end response time of the system under different usage scenarios, including malicious URL detection and general natural-language queries.

Methodology

Three categories of user input were tested:

- **Cached URLs**: URLs that had already been scanned and embedded, enabling direct retrieval from Pinecone.
- **New URLs**: Previously unseen URLs requiring a full processing pipeline—scan, embedding, retrieval, and LLM response generation.
- **General Queries**: Natural-language input that does not contain a URL (e.g., “What is phishing?”), routed directly to an LLM response without embedding or vector retrieval.

Five test cases were executed for each category. End-to-end latency was measured using timestamps from n8n execution logs, which capture the exact moment of workflow initiation and completion. Table 6.2 below summarizes the response times for all three query types.

Table 6.2: Response Latency by Query Type (in seconds)

Query Type	Min (s)	Max (s)	Avg (s)	Notes
Cached URL	11.581	23.929	17.7424	Retrieving Pinecone and LLM inference
New URL	64.2	62.4	63.3	Includes scanning, embedding, and LLM inference
General Query	6.527	10.146	7.7282	LLM inference only; fastest, skips scanning and embedding

Findings

As shown in Table 6.2, new URLs produced the longest response times, largely due to external API calls, embedding, and LLM processing. Cached URLs rank second due to their precomputed embeddings and direct vector retrieval. New General queries—processed without embedding or Pinecone retrieval—achieved faster performance than cached and new URLs.

These findings demonstrate that the system performs within acceptable latency thresholds as stated in NFR3 (Table 3.3), supporting interactive use for both technical and non-technical queries.

6.4 Semantic Similarity Scoring and Retrieval Accuracy

Purpose

To evaluate the accuracy and reliability of the system's semantic retrieval mechanism—specifically, how well it identifies threat-relevant context vectors for user-submitted URLs. This mechanism plays a central role in enabling the chatbot to deliver coherent and risk-informed responses, as it governs what threat intelligence is retrieved from the vector database prior to LLM reasoning. The cosine similarity score serves as a proxy for retrieval confidence, which underpins

the system's ability to retrieve and reason over semantically relevant threat intelligence for user-submitted URLs.

Methodology

When a user submits a URL via Telegram, its threat intelligence is embedded using the mxbai-embed-large model inside a Dockerized Ollama container. The resulting 1024-dimensional vector is sent to Pinecone (configured with cosine similarity, free-tier, AWS us-east-1).

Retrieval is scoped using semantic similarity thresholds as well as metadata filtering (riskLevel, filterbyScanID). Cosine similarity is used to measure the directional alignment between the embedded user input and stored vectors (range: 0 to 1). Empirical testing showed that scores above 0.70 tended to indicate strong contextual matches.

To validate this retrieval mechanism, a set of URLs with known threat levels was submitted. The quality of the retrieval was assessed through:

- Cosine similarity score,
- Metadata alignment,
- Telegram output consistency with the intended risk level and supporting explanation.

These results are summarized in Table 6.3 below.

Findings

Table 6.3: Semantic Retrieval and Reasoning Outcome

Input URL	Cosine Score	Retrieved Risk Level	Metadata Match	Telegram Output
https://buildingcorp.com.au/wp-admin/ty/	0.7607	High	Yes	Accurate
https://drive.google.com/file/d/1ik3UxH3RDigwAr7D269WVkoWp17Cci1n/edit	0.6584	Medium	Yes	Accurate
https://www.villacustomhomes.com/gb-98-commercial-alterations/	0.8731	Low	Yes	Accurate

As shown above, the system was able to:

- Retrieve the appropriate risk-aligned vector based on semantic similarity,
- Maintain correct metadata filtering,
- Generate Telegram messages with LLM reasoning that matched the retrieved data.

Figures 6.1 to 6.3 further illustrate this process using the first test case:

- Figure 6.1 shows the matched Pinecone vector with metadata including riskLevel, filterbyScanID, and antivirus verdicts.
- Figure 6.2 presents the n8n workflow with semantic and metadata filters.
- Figure 6.3 displays the structured Telegram alert generated from the retrieved context.

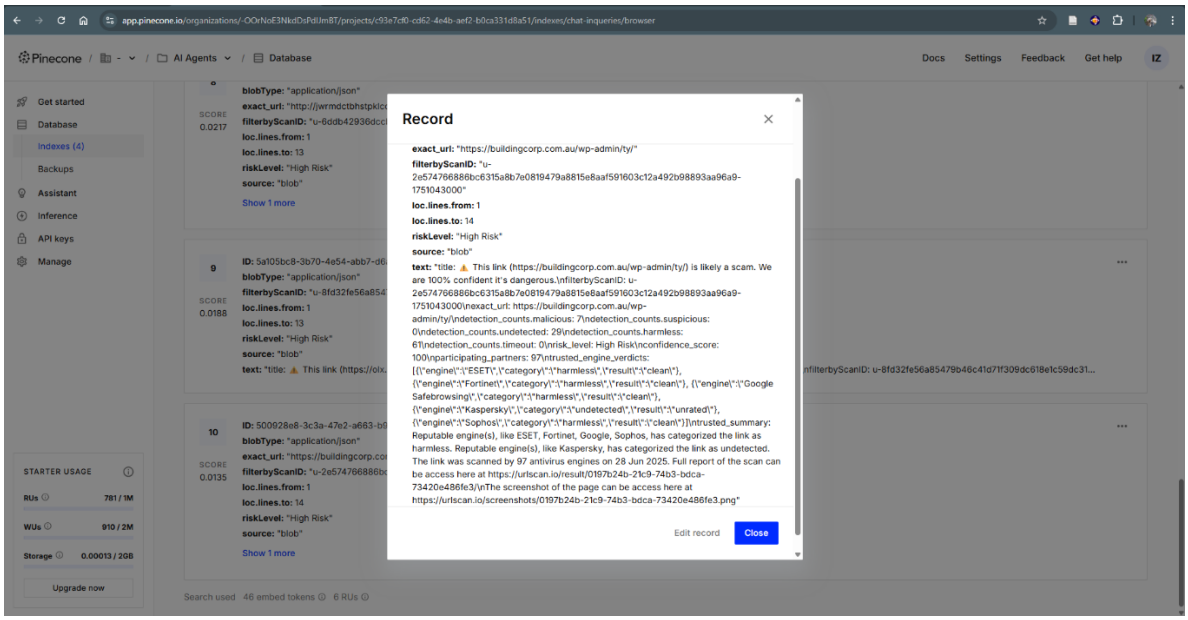


Figure 6.1: Retrieved Pinecone vector record for the URL https://buildingcorp.com.au/wp-admin/ty/, showing cosine score, metadata, and full text match.

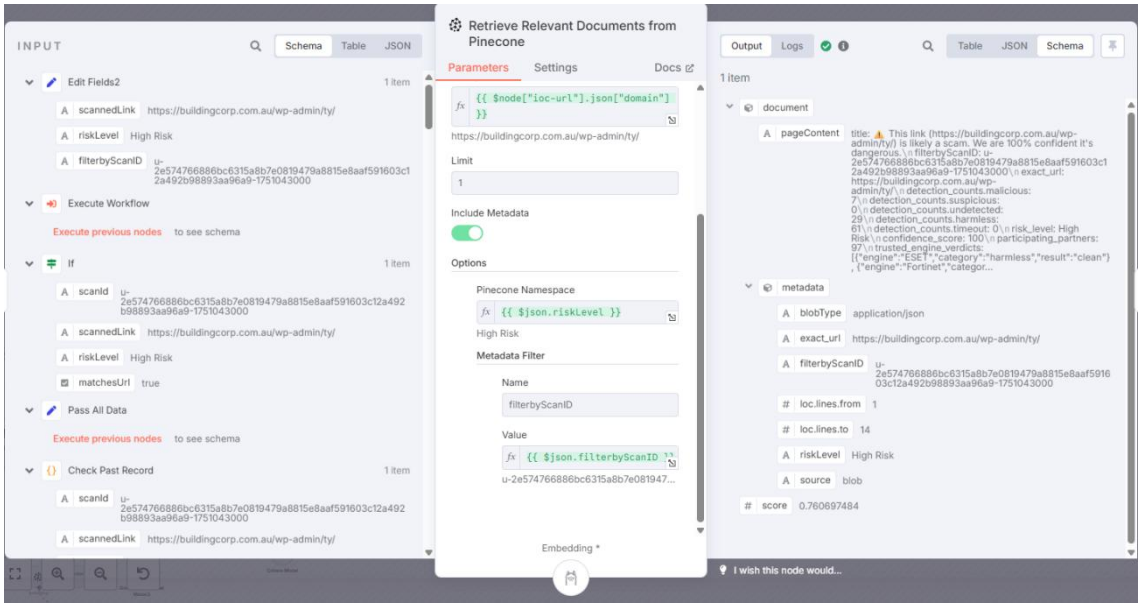


Figure 6.2: n8n retrieval configuration using semantic similarity and metadata filters (riskLevel, filterbyScanID).

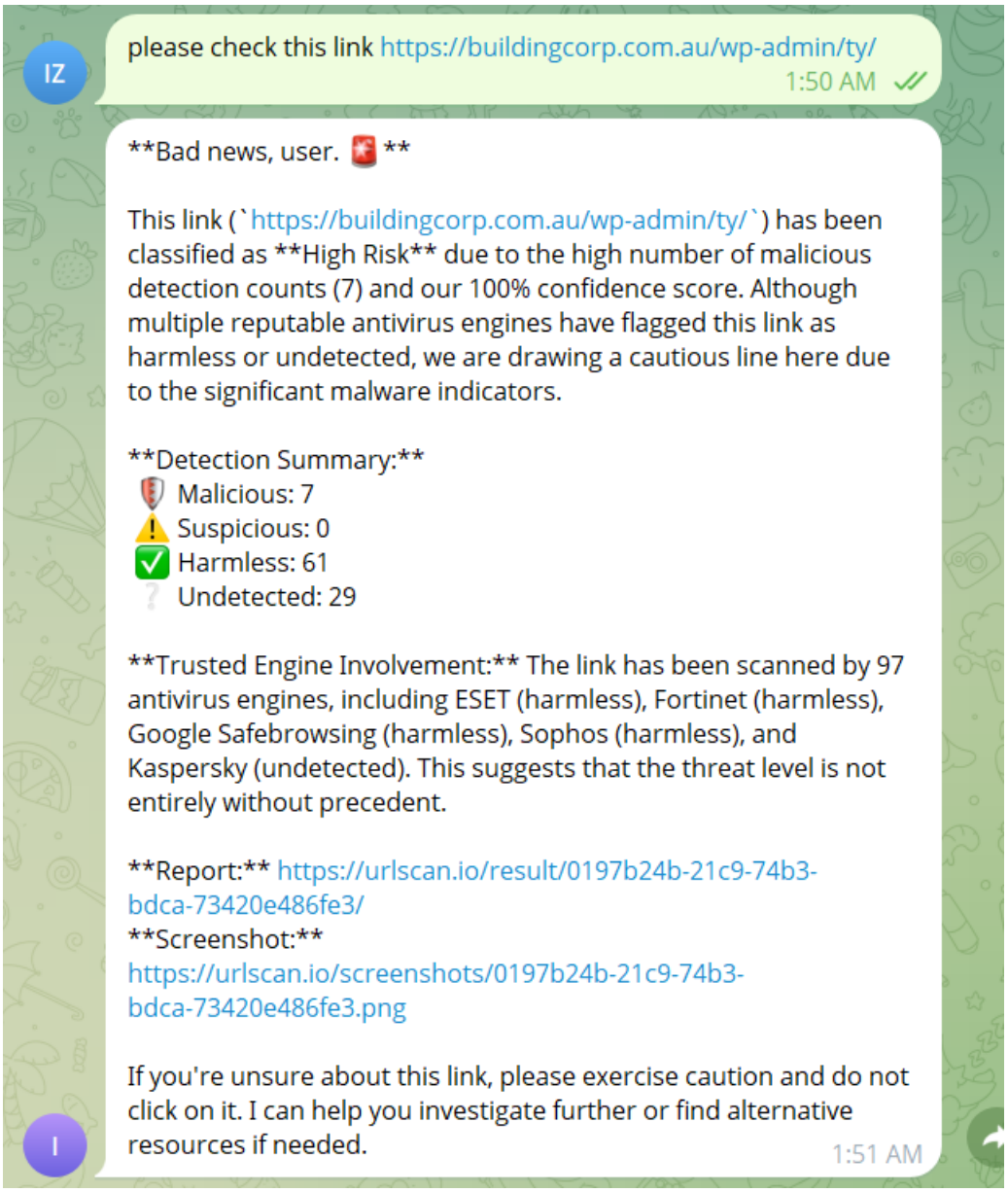


Figure 6.3: Telegram response showing correct risk level, antivirus engine verdicts, and reasoning generated from matched vector.

These figures confirm alignment between semantic retrieval, metadata filtering, and LLM output.

Conclusion

The semantic retrieval component of InQueries proved effective in identifying the correct threat context from stored vector embeddings. Cosine similarity scoring, combined with metadata filtering (filterbyScanID, riskLevel), consistently retrieved the intended phishing vectors. A practical threshold of 0.70 was adopted based on internal testing using known URLs and mxbai-embed-large embeddings. While this threshold was not derived from literature, it reliably reflected strong contextual matches during evaluation, though results just below (e.g., 0.6584) still yielded accurate metadata and Telegram outputs—demonstrating that effective retrieval can occur near the margin.

Table 6.3 shows that retrieved records aligned with expected risk levels, and the LLM consistently generated appropriate reasoning. This final column validated that the LLM interpreted the retrieved context correctly—without hallucinating metadata or fabricating antivirus verdicts.

It's important to note that this evaluation relied solely on internal testing using a small set of developer-verified URLs. The project did not compare results against third-party labeled datasets like ESDAUNG, as InQueries is not intended to match external labeling assumptions. This choice avoids introducing unnecessary bias or misleading conclusions about generalizability.

6.6 Known Testing Limitation

Several technical and design limitations were observed during testing:

- **API Variability:** VirusTotal and urlscan.io responsiveness varied depending on the structure and nature of the submitted URLs. Shortened links, redirects, and highly suspicious domains often took longer to scan. These delays introduced inconsistencies in end-to-end latency measurements and reduced the reliability of timing benchmarks across different test scenarios.

- **Model Inference Dependency:** Although this evaluation used GPU acceleration, the system architecture still faces latency bottlenecks in local LLaMA inference. CPU-only performance was not evaluated due to time constraints.
- **Free-tier Constraints**: Pinecone and API quotas restricted continuous testing and throughput.
- **Single-User Load**: Testing simulated only a single user. Multi-user performance and concurrency were not evaluated.
- **Explainability Evaluation**: While the system generates natural-language responses, no formal assessment of explanation quality, factual grounding, or reasoning traceability was performed due to time constraints.
- **Exact Match Retrieval Only**: The system is designed to retrieve previously scanned URLs via vector similarity, using embedding and metadata filters. It does not attempt to identify structurally or semantically related but unseen phishing links. Retrieval is limited to exact or near-identical matches, based on stored vector representations and metadata.
- **Detection Accuracy Scope**: Accuracy was evaluated only on a small, handpicked set of known phishing URLs verified by the developer. The evaluation measured whether the correct metadata was retrieved and whether the LLM's reasoning matched the retrieved data. It did not assess generalization across external phishing datasets or novel attack patterns.

These limitations do not undermine the proof-of-concept goals, but they highlight areas for refinement in future versions of the system.

6.6 Summary

This chapter evaluated the inQueries prototype across two key dimensions: system responsiveness and semantic retrieval accuracy. Under self-hosted, GPU-assisted conditions, the system demonstrated reliable real-time performance. Cached URLs returned results within 17.74 seconds on average, while general natural-language queries averaged under 7.72 seconds. New URLs—requiring full threat analysis, embedding, and language generation—remained within a reasonable latency window, averaging 63.3 seconds.

Semantic retrieval, based on locally generated embeddings and cosine similarity within Pinecone, successfully identified previously scanned URLs with high precision. A similarity threshold of 0.70 proved effective for distinguishing relevant matches, enabling accurate metadata retrieval and contextual explanation via the chatbot interface.

While the system was tested under single-user conditions and limited API throughput, core detection and explanation capabilities performed as intended. Importantly, the retrieval mechanism was explicitly scoped to detect only exact or previously embedded URLs; generalization to unseen or structurally related threats was not within the system's design.

Overall, the testing confirms that the inQueries pipeline functions as a responsive, accurate, and usable proof-of-concept for AI-assisted scam link detection in messaging environments. The results validate its foundational design assumptions and provide a strong basis for future enhancements in scalability, explainability, and generalized detection.

Chapter 7: Conclusion

7.1 Summary of the Project

. This project set out to design and implement a lightweight, explainable chatbot system for scam link detection in messaging environments. The proposed solution integrates a rule-based classification mechanism, real-time threat intelligence APIs, semantic vector retrieval, and a locally hosted LLM for natural language explanation. The architecture is built entirely with open-source components and free-tier services, demonstrating the feasibility of applying RAG to phishing awareness without relying on commercial infrastructure.

The resulting system, inQueries, operates within Telegram and enables users to submit URLs for analysis and receive structured, human-readable feedback. By embedding both detection and explanation into a familiar messaging interface, the system bridges the gap between high-performance cybersecurity techniques and public usability.

7.2 Achievements and Contributions

The project successfully met all four of its research objectives. These achievements are summarized in Table 7.1 below.

Table 7.1: Achievements and Contributions

Objective	Outcome
1. Translate phishing detection research into public-facing tools	Achieved through integration of RAG and explainable AI into a Telegram-based chatbot.
2. Develop a zero-cost, automated detection workflow	Implemented using n8n, VirusTotal, urlscan.io, and Ollama-hosted LLMs without incurring operational fees.
3. Convert technical metadata into accessible summaries	Achieved using structured prompts and hardcoded summaries to guide LLM-based explanation generation.
4. Evaluate system responsiveness under constrained conditions	Response times measured under GPU-accelerated local conditions, validating system feasibility for interactive use.

In addition to these core goals, the system introduced a manually tuned rule-based risk classifier that relies on trusted antivirus engine verdicts to assess threat severity. Summary strings were generated for each URL and embedded into a vector database using the mxbai-embed-large model, enabling semantic retrieval within a Pinecone index. This setup allowed real-time alignment between user-submitted queries and previously scanned threat records, improving explanation specificity and system interpretability.

The resulting architecture represents a self-contained, low-cost alternative to commercial security platforms by combining real-time scanning, heuristic interpretation, and natural language reasoning in a single workflow.

7.3 Limitations

While the system achieved its primary objectives, several technical and design limitations were identified:

- **API Variability:** The responsiveness of VirusTotal and urlscan.io varied based on URL structure and perceived threat. Shortened or suspicious links often triggered longer scan durations, introducing inconsistency in latency benchmarking.
- **Model Inference Latency:** Although GPU acceleration was used, inference time for the local LLaMA model remained a contributing factor to overall latency. Based on console observations, the model typically required between 6 and 12 seconds to generate responses, depending on prompt length and system load. Combined with API and embedding delays, this sometimes-caused total response times to exceed 60 seconds for newly scanned URLs.
- **Retrieval Precision:** While strict metadata filtering (filterbyScanID) was consistently applied to ensure exact-match retrieval from the vector index, occasional variation in embedding quality or LLM interpretation resulted in explanations that were less specific than expected. These issues were not

caused by incorrect vector matches, but by how the retrieved context was summarized or interpreted during generation.

- **Webhook Reliability:** The use of Ngrok for exposing local endpoints introduced fragility. Session expiry and rate limits interrupted extended testing sessions.
- **Scalability Constraints:** The system was tested under single-user conditions only; multi-user behavior, concurrent message handling, and state tracking were not evaluated.
- **Explainability Assessment:** While the system generates natural-language summaries, no formal study was conducted to evaluate user comprehension, trust, or explanation quality. Future work should include usability testing — particularly with non-technical participants — to assess whether explanations are understandable, trustworthy, and actionable.
- **Exact Match Limitation:** The current system retrieves only previously scanned URLs using strict metadata filtering and vector similarity. It does not attempt to detect semantically similar or structurally related phishing links that have not been embedded in the index.
- **Prompting Technique Limitation:** The LLM explanations relied on a manually designed prompt intended to guide the model's reasoning and structure. While the prompt included relevant scan details and emphasized clarity, no formal experimentation was conducted to evaluate alternative prompting strategies, template variations, or explanation styles. As a result, the consistency, accuracy, and helpfulness of the responses may vary across different threat scenarios or user queries.

These constraints do not undermine the system’s proof-of-concept goals but suggest key areas for future development—especially if transitioning toward production or multi-user deployment.

7.4 Future Work

Several directions could be explored to improve the system's robustness, scalability, and usability:

- **Cloud Deployment:** Replace Ngrok with scalable hosting solutions (e.g., AWS Lambda, Google Cloud Run, or Dockerized containers on Railway/Render) to ensure stable webhook uptime, persistent workflows, and broader availability.
- **Phishing-Specific Fine-Tuning:** Future iterations could fine-tune small open-source models (e.g., Zephyr, Mistral) on domain-specific phishing explanation datasets to improve alignment and clarity in high-risk contexts—without relying solely on general-purpose LLMs for reasoning.
- **Improved Retrieval Logic:** Combine exact string matching with semantic filtering and structured metadata constraints to ensure tighter alignment between query and retrieved context.
- **Multi-user Handling:** Add message state tracking and concurrency controls to support simultaneous interactions from multiple users.
- **User Feedback Loop:** Incorporate mechanisms for users to rate, confirm, or flag the quality of explanations, enabling the system to evolve through user-in-the-loop learning. This could be implemented through interactive buttons in the Telegram interface, which are supported by n8n's Telegram node.

These improvements would support more resilient performance, enable broader adoption, and make the system suitable for high-stakes or enterprise-level use cases.

7.5 Final Reflection

The inQueries project demonstrates that phishing detection systems can be made both explainable and accessible—without requiring commercial infrastructure or deep machine learning expertise. By combining semantic retrieval, real-time threat intelligence, and local LLM reasoning in a lightweight messaging interface, the system transforms technical scan data into plain-language insights that everyday users can understand and act on.

This work bridges the gap between advanced cybersecurity research and practical, public-facing tools. It shows that high-quality threat detection and explanation are not solely the domain of enterprise dashboards or cloud-native services. With careful system design, open-source tools, and a focus on usability, it is possible to deliver trustworthy, real-time security insights directly where users need them most—within their everyday communication platforms.

Although inQueries remains a prototype, it offers more than just a technical demonstration. It contributes a modular, transparent, and reproducible blueprint for future development—whether in academic research, public cybersecurity education, or lightweight commercial defense tools.

Ultimately, this project advances the conversation around human-centered security. It reinforces the idea that interpretability, responsiveness, and user empowerment are not secondary features—but essential pillars of effective and ethical cybersecurity systems.

References

Aljarah, I., Habib, M., Ghosh, S., & Khan, S. (2025, March 28). *EXPLICATE: Enhancing phishing detection through explainable AI and LLM-powered interpretability*. arXiv. https://arxiv.org/abs/2503.20796

Basit, A., Zafar, M. U., Liu, D., Javed, A., & Umer, T. (2021, January 15). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems, 76*(1), 139–162. https://doi.org/10.1007/s11235-020-00733-2

Bayer, R., Saeed, A., & Barna, L. (2023, August 2). *Building a resilient domain whitelist to enhance phishing blocklist accuracy*. APWG eCrime Research. https://docs.apwg.org/ecrimeresearch/2023/20240080211286378.pdf

Blake, E. (2025, March 18). *PhishSense-1B: A technical perspective on an AI-powered phishing detection model*. arXiv. https://arxiv.org/pdf/2503.10944v1

Chang, C., & Aïmeur, E. (2024, April 24). Chat or trap? Detecting scams in messaging applications with large language models. *IEEE*. https://ieeexplore.ieee.org/document/10851753

Du, Y., Li, X., Zhang, J., & Zhou, J. (2024, April 18). A novel RAG framework with knowledge-enhancement for biomedical question answering. *IEEE*. https://ieeexplore.ieee.org/document/10822837

ESDAUNG. (2021). *Segmentation-based phishing URL detection*. In *Proceedings of the 2021 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)* (pp. 307–312). ACM. https://doi.org/10.1145/3486622.3493983

ESDAUNG. (2023). *PhishDataset: Balanced malicious and benign URL dataset* [Data set]. GitHub. https://github.com/ESDAUNG/PhishDataset

Kenneweg, T., Kenneweg, P., & Hammer, B. (2024). *RAGVAL: Automatic dataset creation and evaluation for RAG systems*. In *Proceedings of the 2nd International Conference on Foundation and Large Language Models (FLLM 2024)* (pp. 470–475). IEEE. https://doi.org/10.1109/FLLM63129.2024.10852482

Korkmaz, O., Cinar, M., & Ozdemir, S. (2022, October 15). A hybrid phishing detection system using deep learning-based URL and content analysis. *Elektronika ir Elektrotechnika, 28*(5), 72–79. https://eejournal.ktu.lt/index.php/elt/article/view/31197

Kurniawan, K., Kiesling, E., & Ekelhart, A. (2024, May 20). CyKG-RAG: Towards knowledge-graph enhanced retrieval augmented generation for cybersecurity. *Proceedings of the RAGE-KG 2024 Workshop at ISWC 2024*, 1–12. https://www.researchgate.net/publication/386198861_CyKG-RAG

Macke, J., Bansal, A., & Shcherbakov, K. (2023, September 9). Human-in-the-loop machine learning: Active learning and annotation for human-centered AI. *IEEE*. https://ieeexplore.ieee.org/document/10280384

Ozcan, M. H., Polat, K., & Yildirim, M. (2023, June 22). A hybrid DNN–LSTM model for detecting phishing URLs. *Neural Computing and Applications, 35*, 10665–10681. https://doi.org/10.1007/s00521-021-06401-z

Riasat, A., Shabbir, S., & Ali, M. (2022, December 1). Adoption barriers of cybersecurity features in mobile banking: A usability-focused study. *Unpublished manuscript*.

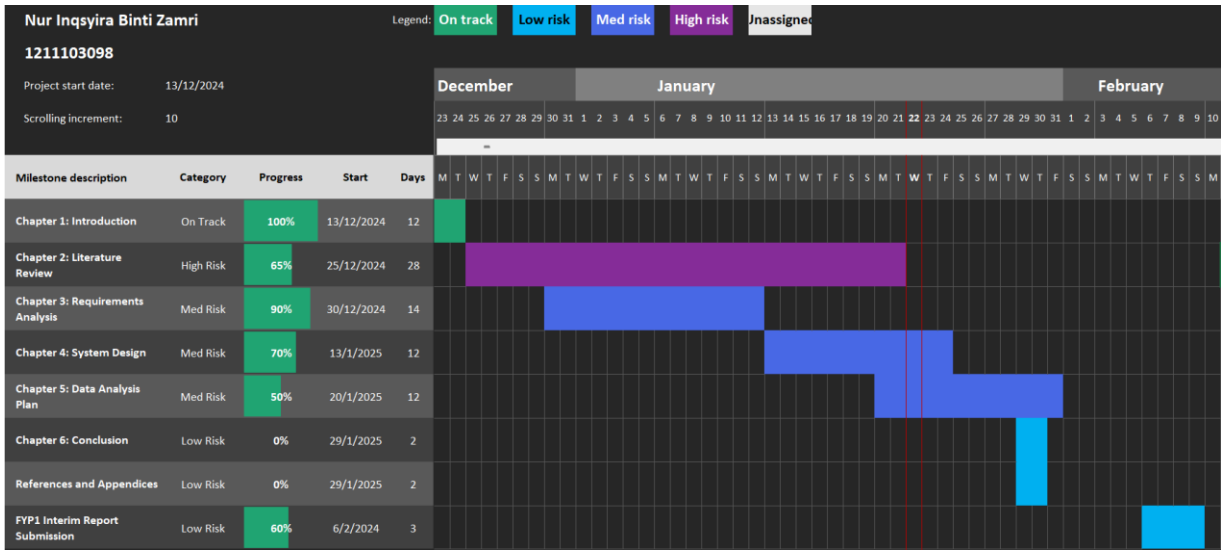
Sun, J., Li, Y., Zhang, X., & Jiang, T. (2024, February 3). A product-aware query auto-completion framework for e-commerce search via retrieval-augmented generation. *Amazon Science*. https://assets.amazon.science/5a/e2/46450cad4f3cae1dbefd7b81deb8/a-product-aware-query-auto-completion-framework-for-e-commerce-search-via-retrieval-augmented-generation-method.pdf

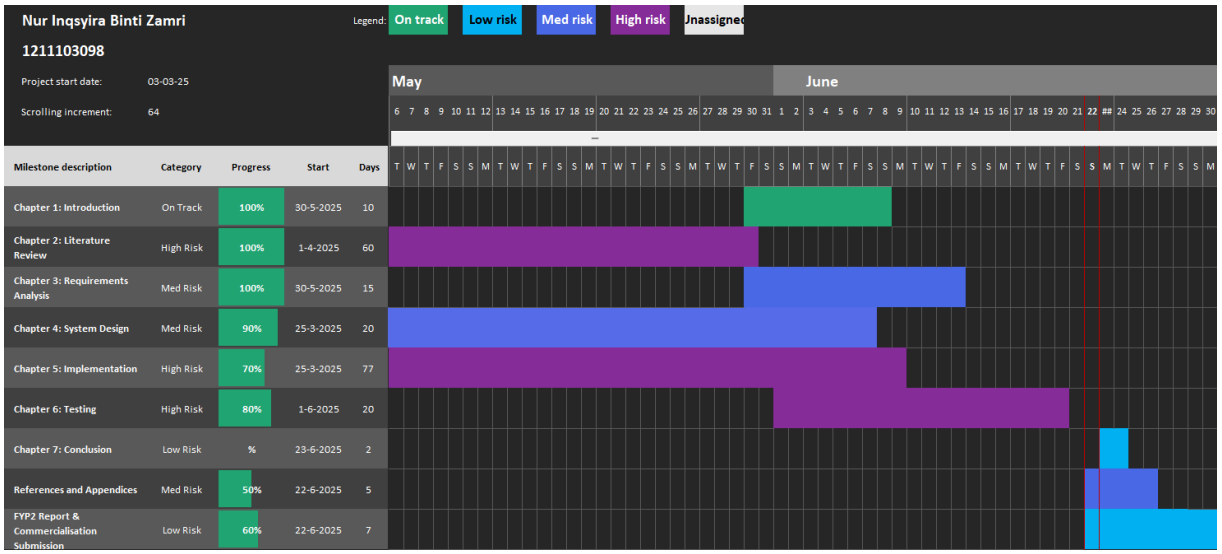
Symantec. (2022, December 15). *2022 threat landscape year in review*. https://www.symantec.broadcom.com/hubfs/2022_Threat_Year_in_Review.pdf

Tupsamudre, H., Meshram, A., & Dubey, H. (2021, December 3). *PhishMatch: A layered approach for effective detection of phishing URLs*. arXiv. https://arxiv.org/pdf/2112.02226

Wei, W., Das, S., McDonald, A., & Roesner, F. (2023, April 24). “There’s so much responsibility on users right now”: Expert advice for staying safer from hate and harassment online. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–16. https://doi.org/10.1145/3544548.3581229

Appendix A: Gantt Chart





Appendix B: FYP1 and FYP2 Meeting Logs

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 28 November 2024	**Meeting No.: 1**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-CS-T2430-0062	**Project Type:** ~~Research based/Application based/~~ Application & Research based

Project Title : Enhancing Signature-Based Network Intrusion Detection System (NIDS) with Dynamic Threat Intelligence Integration	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** Khairil bin Anuar **(if applicable)**

Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology /~~
~~Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Discussion of the relevancy of project direction with supervisor, considering the project's time consumption, hardware capabilities, and available resources.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize project title
- Create gantt chart for project planning
- Conduct background study

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Problems:

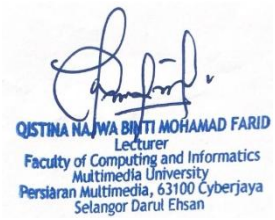
- The project title needs to be changed after discussing the current findings with the supervisor

Plan for the solutions:

- The supervisor provided alternative ways to find new title by tweaking the current title and eliminating the word 'integration' to make it more achievable.
- The supervisor also advised focusing on a research-based approach with suggested keywords based on the current title (e.g., improvement, analysis, threat intelligence)

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Students need to explore more on the current trend of technology and how the way her project could be different from others. Topic was quite relevant but need to be more achievable since their timeline to do FYP is only a year including other subject that should not be enough for them to do a complex project. LR and Introduction should be start by now since she would have explore more on the NIDS.



……..
Supervisor's Signature

……
Student's Signature

……..
Co-Supervisor's Signature
Signature
(if applicable)

……
Company Supervisor's

(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can
   have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 13 December 2024	**Meeting No.: 2**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-CS-T2430-0062	**Project Type:**

	~~Research-based/Application-based/~~ Application & Research based
Project Title : AI-Driven Scam Link Detection in Messaging Platform	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** Khairil bin Anuar **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize project title
- Create gantt chart for project planning
- Chapter 1: Introduction (version 1)
- Chapter 2: Literature Review (version 1)

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Modification based on supervisor's feedback:
  - Chapter 1: Introduction (version 2)
  - Chapter 2: Literature Review (version 2)
- Read more research papers for the literature review.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Problems:

- The background overview in Chapter 1: Introduction need to be written longer.
- The problem statement should follow the format of one problem statement, one paragraph.
- Add elaboration on each project scope and convert it into table to enhance readability.
- Need to further specify the target audience.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The student should review more research papers to strengthen the literature review, as the topic has changed quite a bit from the original one. Since the new topic is innovative, it's important for the student to have a clear understanding of how to integrate the messaging app into the proposed system. It might help to work on the project implementation alongside the research phase to better manage expectations.

…….. ……

Supervisor's Signature Student's Signature

…….. ……

Co-Supervisor's Signature Company Supervisor's Signature

(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 17 December 2024	**Meeting No.: 3**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-CS-T2430-0062	**Project Type:** ~~Research-based/Application-based/~~ Application & Research based
Project Title : AI-Driven Scam Link Detection in Messaging Platform	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** Khairil bin Anuar **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Chapter 1: Introduction (version 2)
- Chapter 2: Literature Review (version 2)
- Created several diagrams to be included on Chapter 4: System Design

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / ~~Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Read more research papers to strengthen literature review.
- Create 10 survey questions.
- Chapter 3: Requirements Analysis

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

No problems encountered upon completion of task

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Keep up the good work. Theres seems to be no problem with the updated chapters and system design diagrams. Continue to focus on strengthening the literature review and start working on the requirements analysis, plus the survey questions by next week. Great effort so far :)

…….. ……

Supervisor's Signature Student's Signature

…….. ……

Co-Supervisor's Signature Company Supervisor's Signature

(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 6 January 2025	**Meeting No.: 4**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-CS-T2430-0062	**Project Type:** ~~Research-based/Application-based/~~ Application & Research based
Project Title : AI-Driven Scam Link Detection in Messaging Platform	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri

Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework~~ / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Chapter 2: Literature Review (version 3)
- Finalize and collect survey.
- Chapter 3: Requirement Analysis

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework~~ / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Chapter 4: System Design
- Chapter 5: Data Analysis Plan
- Prototype Development

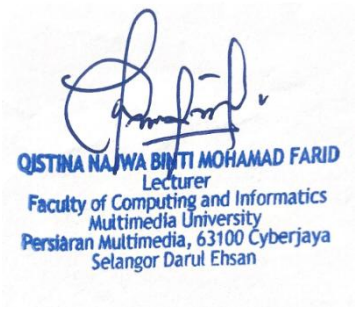
3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

No problems encountered upon completion of task

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Review the literature for another version as student got a bit confused. Provide insight on Chapter 4 and 5 for student to start her interface implementation



……...
Supervisor's Signature

……
Student's Signature

……..
Co-Supervisor's Signature
Signature
(if applicable)

……
Company Supervisor's
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 22 January 2025	**Meeting No.: 5**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-CS-T2430-0062	**Project Type:** ~~Research-based/Application-based/~~

	Application & Research based
Project Title : AI-Driven Scam Link Detection in Messaging Platform	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / ~~Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Chapter 4: System Design (except for 4.7 Interface Design)
- Chapter 5: Data Analysis Plan (version 1)
- Prototype Development phase 1: explore and planning.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework /~~ Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Success integration of WhatsApp API with Nodejs
- 4.7 Interface Design
- Chapter 5: Data Analysis Plan (version 2)
- Chapter 6: Conclusion
- Draft Report Completion

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

I need a Facebook Business account to use the WhatsApp Business Cloud API on Meta. After creating one, I realized that the business must first be verified by Meta, requiring me to upload business documents, which I didn't have. Fortunately, I remembered that I had created a business account in the past for my small business, so I used that instead.

Another challenge with Meta's developer platform is that every new message template requires admin approval. This process is quite time-consuming, as I must wait for the status to change from "In Review" to "Active", which can take up to 24 hours.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Review the ideas for student to start implementation. The ideas had been threw are magnificient. However student need to signify few improvision from her part.

…….... ……
Supervisor's Signature Student's Signature

…….... …….
Co-Supervisor's Signature Company Supervisor's
Signature
(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 28 January 2025	**Meeting No.: 6**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-CS-T2430-0062	**Project Type:** ~~Research-based/Application-based/~~ Application & Research based
Project Title : AI-Driven Scam Link Detection in Messaging Platform	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Cybersecurity	
Supervisor Name: Qistina Najwa Binti Mohamad Farid	**Co-Supervisor Name:** Khairil bin Anuar **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / ~~Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Success integration of WhatsApp API with Nodejs
- 4.7 Interface Design
- Chapter 5: Data Analysis Plan (version 2)
- Chapter 6: Conclusion
- Draft Report Completion (version 1)

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Draft Report Completion (version 2)
- Slides for Oral Presentation

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

No problems encountered upon completion of task

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Fine review for all the chapters before oral presentation

…….. ……
Supervisor's Signature Student's Signature

…….. ……
Co-Supervisor's Signature Company Supervisor's
Signature
(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can

have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 14 April 2025	**Meeting No.:** 1
Meeting Mode: **(In-person / ~~Online~~)**	
Project ID: FYP02-CS-T2510-0060	**Project Type:** **~~Research-based / Application-based~~ / Hybrid**
Project Title : AI-Driven Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri

Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) ~~/ Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Exploration of Link Analysis Tools that Incorporate Sandboxing:
  - Explored multiple sandbox environments: Microsoft Defender for Endpoint, VirusTotal, ANY.RUN, Hybrid Analysis, and Cuckoo Sandbox
  - Joined Microsoft Developer Program (unsuccessful access to E5)
  - Responded to ANY.RUN for trial (no reply)
  - Chose Hybrid Analysis as final sandbox due to balance of usability and API access
- Project Direction / Title Development:
  - Brainstormed on how to tweak the project titles and ideas, focusing on n8n capability and academic research requirement.
  - Analyzed feasibility of each title against various factors.
- AI Element Direction / Exploration:
  - Learned differences between predictive and generative AI
  - Considered tweaking to explainable AI for the scam link detection system
  - Reviewed 3rd-party AI-based APIs like Google Safe Browsing and PhishTank.
  - Discovered constraints around using others' AI models (no training = rule-based only)
- System Design / Workflow Changes
- Development & Implementation:
  - Migrated implementation environment from Replit to VS Code
  - Solved VS Code logging issue using Ngrok
  - Reconfirm webhook flow concept (understood its limitations)
  - Switched from WhatsApp to Telegram after account flag by Meta

- Self-hosted n8n automation tool using Docker
- Debugged API requests and webhook routing on n8n with Ngrok
- Integrated n8n with Telegram and Hybrid Analysis API

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize decision on how to incorporate AI element:
  - Option 1: Explainable AI using LLMs (e.g., via Gemini/Ollama)
  - Option 2: Improve risk classification logic (symbolic/rule-based AI)
  - Option 3: Use existing AI-powered APIs (clearly state in title)
- Finalize project title, objectives, and project scope

Additional Considerations:

- Determine if Hybrid Analysis qualifies as AI-based for academic purposes
- If using rule-based logic, clearly define it as symbolic AI
- Clarify if explainability features (e.g., chatbot explanation of classification) will be part of the system
- Consider converting project to application-only, if research component becomes too complex
- Apply agile methodology to finish within timeline

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Throughout the development during trimester break, I've encountered several technical and conceptual challenges. Initially, selecting an appropriate sandbox environment was difficult. Microsoft Defender for Endpoints required MMU admin approval for trial access, while VirusTotal did not offer sandboxing in the free version. ANY.RUN satisfied the project requirements but required a paid API, and attempts to activate a free trial via their support went unanswered. Meanwhile, Hybrid Analysis was ultimately selected due to its accessible free API and sufficient behavioral analysis, despite using a predefined sandbox. Attempts to self-host Cuckoo Sandbox were abandoned due to its complex setup requirements.

Conceptual challenges arose in defining AI's role within the project. It became evident that while Hybrid Analysis is a powerful tool, it does not inherently qualify as AI-based, complicating the justification of an "AI-powered" system. Efforts to explore generative and predictive Ai, including integrations with Google Safe Browsing and PhishTank, were hindered by unclear documentation and the realization that employing prebuilt models without training data aligns more with rule-based AI rather than adaptive learning models. To ensure transparency, a

potential solution is to explicitly state in the title and documentation that the system employs symbolic AI, specifically leveraging rule-based logic informed by Hybrid Analysis scoring.

Further complications appeared during implementation. When migrating from Replit to VS Code, logging limitations made debugging difficult. This was solved using Ngrok to expose local servers and capture full request and response data. WhatsApp webhook integration posed another issue, as it was initially misunderstood that messages could be intercepted and modified before reaching recipients. Upon realizing WhatsApp's limitations, the system logic was redesigned to quiz users on links rather than block them preemptively. This shift in approach, from proactive link blocking to quiz-based user awareness, was part of Version 1 of the solution. However, due to time constraints and the complexity of collecting user insights, this version was deemed no longer applicable. The project pivoted back to the original idea, automated scam link detection system.

Additionally, in the middle implementation, I encountered an unexpected setback when Meta flagged my WhatsApp API account due to their strict policies, which required business registration. As a result, I was forced to switch to using Telegram. Faced with this ongoing issue, I decided to explore new technologies that could help streamline my complex development, and gladly discovered n8n!

While setting up the Telegram trigger on n8n, I ran into a new challenge: a miscommunication with the webhook caused by the tool being hosted on Docker. To resolve this, I replaced the internal webhook URL within n8n with a public Ngrok URL, which allowed me to establish the necessary external API connections and move forward with the project.

Integration of the Hybrid Analysis API within n8n was another challenge, as there was no clear community guidance or documentation available. After significant trial and error and AI-assisted learning, the API was successfully implemented by relying on Hybrid Analysis documentation.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Excellent, please maintain your progress.

……
Supervisor's Signature

……..
Student's Signature

……..
Co-Supervisor's Signature
Signature
(if applicable)

……
Company Supervisor's

(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 30 April 2025	**Meeting No.:** 2
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-CS-T2510-0060	**Project Type:**

	~~**Research-based / Application-based /**~~ **Hybrid**
Project Title : AI-Driven Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Decided to go with option 1: Explainable AI using deepseek-r1 model. This project would utilize Telegram API as the chatbot interface for all the interaction between system and user.
- To enhance the accuracy, RAG AI is actively being explored
- Implemented workflow with Hybrid Analysis (HA) but after several testing, it is confirmed that HA offer several inconvenient API barriers
- Performed a quick analysis of Hybrid Analysis (HA), VirusTotal, and URLScan.io using ChatGPT, Gemini, and Grok, and the responses showed that VirusTotal and URLScan.io ranked highest as the most relevant tools in the industry.
- Implemented new workflow incorporating VirusTotal and URLScan.io (ver. 1)
- Research and finalize tools for implementing RAG AI (use Pinecone for the data insertion and retrieval)

- Configured ollama node with llama3.2 & deepseek-r1 as the potential models to deliver explainable AI later.
- Configured and authenticated THE Pinecone node on n8n for RAG.
- Decided on the embedding model to be used for vector store and conversion on the RAG
- Solved Pineocne insertion problem: implement RAG AI to have data from both VirusTotal and URLScan's scans result as the stored knowledge base.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Workflow (ver. 2) – complete flow/solve errors + implement RAG AI to retrieve the correct knowledge and generate an accurate response when user ask about url risks.
- Explore research paper on RAG AI for phishing detection and explainable AI (XAI) which uses AI to explain threats.
- Study and implement prompt technique to be used for the explainable AI.
- Craft expected responses for low/medium/high risk scanned results to better train the AI chatbot.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

During the early stages of developing my Retrieval-Augmented Generation (RAG) AI Agent, I faced several technical hurdles, particularly in processing and embedding real-time reports from VirusTotal and URLScan.io.

Initially, I attempted to merge both reports into a single document and format the data as JSON using stringify(). The plan was to upload this structured data into Pinecone and store it as part of my knowledge base. To align with common practices in the AI community, where data is often stored in formats like PDF, Word, or Excel, I configured and authenticated Google Drive and the Google Docs node within n8n.

However, I quickly ran into a limitation. My dataset was already structured in JSON, and I couldn't find any existing workflows that handled JSON directly as the knowledge base for a RAG system. To work around this, I converted the JSON content into a DOCX file before uploading it. After successfully triggering Google Drive to create and populate the file with the merged scan reports, the resulting document used up to 119 pages, reflecting the sheer volume of information.

When testing this approach, the system returned an error: the document exceeded the allowed size limit. To address this, I introduced a data transformation step using JavaScript in n8n, which extracted only the most critical

information from the scan results. This trimmed down the document significantly, resolved the size-related error, and allowed for successful Pinecone insertion.

Another major challenge came from embedding model compatibility. At first, I used publicly available embedding models popular in community tutorials, but this led to recurring errors in the embedding node. Upon further investigation, I realized the problem stemmed from incompatible dimensionality. My Pinecone setup apparently, required vectors of a specific length. So, after some trial and error, I identified the issue: Pinecone was configured to support embeddings with a 1024-dimensional vector space. I resolved this by switching to the mxbai-embed-large:latest model, which supports cosine similarity-based retrieval and can be deployed using the Ollama node. This choice aligned perfectly with my system's requirements and allowed the embedding process to function smoothly.

Now, my focus is on retrieving the successfully stored knowledge effectively into my workflow. I need to integrate a new node that functions as an AI Agent, designed to operate based on a predefined system prompt using Llama3.2 as the base model. This agent will determine whether to invoke specific tools based on the questions received and will process the final response from the tool before passing it back.
To achieve this, I plan to attach an answer questions node to the AI Agent that interacts with the Pinecone vector store and select an appropriate reasoning model (DeepSeek-R1, for now, due to convenience).

To sum up, my next step involve refining the retrieval mechanism to ensure the AI fetches only the most relevant knowledge and structuring effective prompts to optimize how the AI processes the retrieved data, generates reasoning, and delivers accurate final responses.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Excellent, please maintain your progress.

……
Supervisor's Signature

……..
Student's Signature

……..
Co-Supervisor's Signature
Signature
(if applicable)

……
Company Supervisor's
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 20 May 2025	**Meeting No.:** 3
Meeting Mode: **(In-person ~~/ Online~~)**	

Project ID: FYP02-CS-T2510-0060	**Project Type:** ~~**Research-based / Application-based /**~~ **Hybrid**
Project Title : AI-Driven Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report / ~~Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Completed workflow (ver. 2) – implemented message handling and RAG but still faced issue with data retrieval.
- Tested metadata filtering, applied contextual retrieval (Anthropic approach) before Pinecone insertion, and experimented with different chunking techniques to address inconsistent results.
- Studied research papers to gain insights into RAG AI, AI Agents, and explainable AI projects.
- Created easy-to-understand response messages to improve public accessibility to cybersecurity practices.
- Tested various prompting techniques to achieve desired response quality.
- Started report writing (initially for Mecon 2025 submission), but due to limited test data, shifted focus to FYP2 report instead.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Fix inaccurate Pinecone retrieval issue
- Finalize effective prompting strategy.
- Implement possible fix solution to address Pinecone reranking inaccuracies.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Problem: Inaccurate knowledge retrieval

Implemented solutions:

1. Using metadata filtering on Pinecone – but after robust testing, it still retrieves other URLs from previous sessions (meaning it retrieves the incorrect scanned result).
   With respect to this issue, I tried limiting the maximum response number from 4 to 1. It did retrieve only one URL scanned result, but still faced inconsistent retrieval.
   I figured out this is because my current RAG works with cosine similarity, meaning it searches the URL *semantically* instead of matching the exact URL.
   For example: if the intended URL is https://www.villacustomhomes.com/gb-98-commercial-alterations/, it will also retrieve any URL from the stored vector that contains matched words like “homes” and “alterations.”
   From my findings, even the exact URL match is not ranked in the highest place (1st).

2. Implemented enhanced contextual retrieval approach from Anthropic – this helps my AI understand the JSON-structured knowledge base better during the retrieval reasoning. However, it does not solve the incorrect retrieval issue I am facing.
3. Tried various chunking strategies – including character text split, token-based split, and even custom chunking using my own function node. Sadly, none of them solved the problem.
4. Tried separating AI Agent calling in the workflow – by first calling the retrieval node, then only calling the AI Agent.
   This means I need to modify my workflow to handle different message types: those containing a URL, those without, and the first welcome message.
   I couldn't figure out how to separate them properly in n8n, but I did implement message handling for a more modular workflow approach.

Rejected solutions (discover barriers beforehand):

1. Create comprehensive instructions for the AI Agent and store them in Google Drive, which would then be embedded together with the Pinecone storage containing other scanned URLs.
   Due to free-tier limitations of Pinecone and other AI models, token usage became a problem.
   This solution was not implemented as it is both time-consuming to craft effective instructions and consumes a large number of tokens.

Key takeaway:
From the robust solutions I've tried, the core issue seems to be with Pinecone's reranking. The current plan is to implement a fix that addresses this reranking issue, so the exact URL appears at the top.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Excellent, please maintain your progress.

…… ……..
Supervisor's Signature Student's Signature

…….. ……
Co-Supervisor's Signature Company Supervisor's Signature
(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MMU® MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 2 June 2025	**Meeting No.:** 4
Meeting Mode: **(In-person ~~/ Online~~)**	
Project ID: FYP02-CS-T2510-0060	**Project Type:**

	~~**Research-based / Application-based /**~~ **Hybrid**
Project Title : AI-Driven Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name: (if applicable)**
Collaborating Company: (if applicable)	**Company Supervisor Name: (if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report / ~~Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Studied RAG to solve inaccurate retrieval (in-depth: chunking, embeddings, reranking, hallucinations, vector databases, etc) from eBook Mastering RAG by Galileo.
- Simplified workflow and reduced unnecessary complexity.
- Finalized prompting.

- Replaced stringify approach, which converts JSON into a string, with Recursive JSON Flattener approach, to preserve nested relationships between attributes.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Complete Chapter 1 (Introduction)
- Complete Chapter 2 (Literature Review)
- Complete Chapter 3 (Requirement Analysis)
- Complete Chapter 4 (System design)
- Complete Chapter 5 (Implementation)

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Problem #1: Pinecone handling on n8n

During my investigation, I identified a similar issue reported by a member of the n8n community source. Their findings confirm that the problem I encountered stems from a loophole in the system, specifically in how Pinecone handles namespace and metadata filtering. Identifying this root cause has helped me to discover other alternatives.

On top of that, I discovered a new loophole in the system. When configuring namespace and/or metadata filtering, the JavaScript function for dynamic calling consistently returns an empty retrieval response, but not with static calling. While I manage to resolve the issue, it remains impractical for my pipeline, as I need to call the metadata dynamically based on the url for each session.

Problem #2: Less suitable type of vector embedding

Spotted an issue with the currently used embedding model (mxbai-embed-large). It might not be the best fit for my use case since it's a dense embedding model optimized for capturing the overall meaning of natural language. It

works by converting sentences into numerical vectors that represent their meaning, which is effective for comparing longer, human-readable text.

My dataset, however, is in nested JSON format and contains a mix of structured fields, semi-structured metadata, and unstructured descriptive text. Because of this blend, a more suitable approach would be hybrid embeddings (combining dense and sparse vectors) to support both understanding of overall meaning and precise matching on keywords or field values.

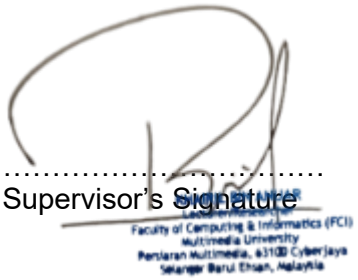
That said, since my fyp takes a proof-of-concept approach, I won't be utilizing all the reports retrieved from VirusTotal and URLScan just yet. Instead, I'll move forward with the current embedding setup and focus on other ways to reduce the impact of this limitation, such as transforming the input to be better suited for embedding. One early version of this solution involves flattening the JSON structure using custom code.

P/S: Previously, most of my analysis was based on RAG project examples on n8n community, but those didn't involve using code or structured data as external input. That difference definitely impacts on how I should approach chunking, embedding, and storage.

Problem #3: Ngrok reaches limit

On 29 May, I encounter HTTP Request limit from ngrok and had to wait for my limit resetting which scheduled on 1st June.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)



…… ……..

Supervisor's Signature Student's Signature

…….. …….

Co-Supervisor's Signature Company Supervisor's Signature

(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 16 June 2025	**Meeting No.:** 5
Meeting Mode: **(In-person ~~/ Online~~)**	

Project ID: FYP02-CS-T2510-0060	**Project Type:** ~~**Research-based / Application-based /**~~ **Hybrid**
Project Title: AI-Driven Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report / ~~Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- ver1 Chapter 1 (Introduction)

- ver1 Chapter 2 (Literature Review)
- ver1 Chapter 3 (Requirement Analysis)
- ver1 Chapter 4 (System design)
- ver1 Chapter 5 (Implementation)

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report~~t / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Update Chapter 1 – 5
- Complete Chapter 6 (Testing)
- Complete Chapter 7 (Conclusion)
- Complete Commercialisation document
- Finalize workflow / project implementation

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

Problem: Minor corrections needed for report writing.
Expected solutions based on discussion with the supervisor:

- Revise the abstract to clearly present the project.
- Add descriptive captions for each table and figure.
- Elaborate on the requirement analysis, especially the survey conducted with 74 respondents during FYP1.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

……
Supervisor's Signature

……..
Student's Signature

……..
Co-Supervisor's Signature
Signature
(if applicable)

…….
Company Supervisor's

(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 22 June 2025	**Meeting No.:** 6
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-CS-T2510-0060	**Project Type:** **~~Research-based / Application-based /~~ Hybrid**
Project Title: inQueries: An Explainable RAG-AI Chatbot for Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri
Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report / ~~Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- ver2 Chapter 1 – 5
- ver1 Chapter 6 & 7
- Complete Commercialisation document
- Finalize workflow / project implementation
- Finalize title

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) /~~ ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ ~~Draft Final Report~~ / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Complete the final report
- Create and print the poster
- Create video

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

No problems encountered upon completion of task

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

…… ……..
Supervisor's Signature Student's Signature

…….. ……
Co-Supervisor's Signature Company Supervisor's Signature
(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 26 June 2025	**Meeting No.:** 7
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-CS-T2510-0060	**Project Type:** **~~Research-based / Application-based /~~ Hybrid**
Project Title: inQueries: An Explainable RAG-AI Chatbot for Scam Link Detection in Messaging Platforms	
Student ID : 1211103098	**Student Name:** Nur Inqsyira Binti Zamri

Student Programme and Specialisation: Bachelor of Computer Science (Hons.) (Cybersecurity)	
Supervisor Name: Mr. Khairil Bin Anuar	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report / ~~Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- ver2 Chapter 6 (Testing): Shifted from individual node latency evaluation to RAG latency evaluation.
- Finalized the final report
- Finalized the poster
- Solved the RAG incorrect retrieval issue
- Further simplified the workflow by replacing the longer method of handling different message types with a more comprehensive prompting approach.
- Accepted URLScan.io's limitation of rejecting certain url.
- Recorded and stored an Excel sheet containing the list of URLs approved by URLScan.io through manual testing in the developed pipeline.
- Conducted robust testing with different prompting styles and different AI chat models.
- Observed differences in AI response when using the Basic LLM Chain node without memory vs. the AI Agent node with memory to recall past conversations.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / ~~Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Print the poster
- Create the video

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

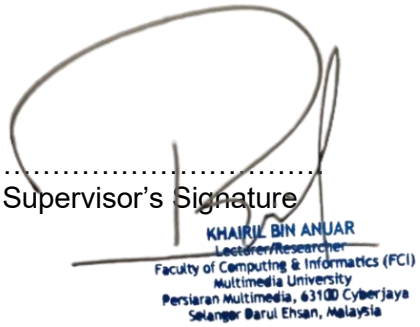
Problem #1: The Pinecone insertion on live scanning report that acts as the chatbot knowledge base encountered duplication during insertion.
Solution: Placed a set node that returns only 1 item right before the Pinecone node. Then acknowledged that this single item would be labeled as `pageContent` during the vector insertion.

Problem #2: Incorrect Pinecone retrieval issue
Solution: Went through the Pinecone documentation and discovered the limitation of their metadata and namespace usage. This explains why only static calling was working (as noted in Meeting Log 4). So, I proceeded to use another metadata. Instead of using the full URL for metadata filtering, I used the scan ID from VirusTotal as the metadata filter because of the robust coverage demonstrated by VirusTotal across different types of URLs. As a result, the Pinecone retrieval issue is finally solved using the `riskLevel` namespace and `scanID` metadata filter.

Problem #3: Prompting in the system message (AI Agent node) does not work as intended.
Solution: Modified the Data Transform node to include text that explains what the report is about to the AI assistant, so that when the report is retrieved, the AI knows how to interpret it and can query it correctly. This approach also helps reduce the impact of using dense instead of sparse similarity vectors, as previously discussed in Meeting Log 4, Problem #2: Less suitable type of vector embedding.

Problem#4: Google Gemini 1.5-Flash, which had been used as the AI chat model, reached its limit on 24th June after robust testing had been conducted.
Solution: Continued the testing using locally hosted Ollama with two models: deepseek-r1:1.5b and llama3.2:latest.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)



…… ……...
Supervisor's Signature Student's Signature

…….. ……
Co-Supervisor's Signature Company Supervisor's Signature
(if applicable) **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

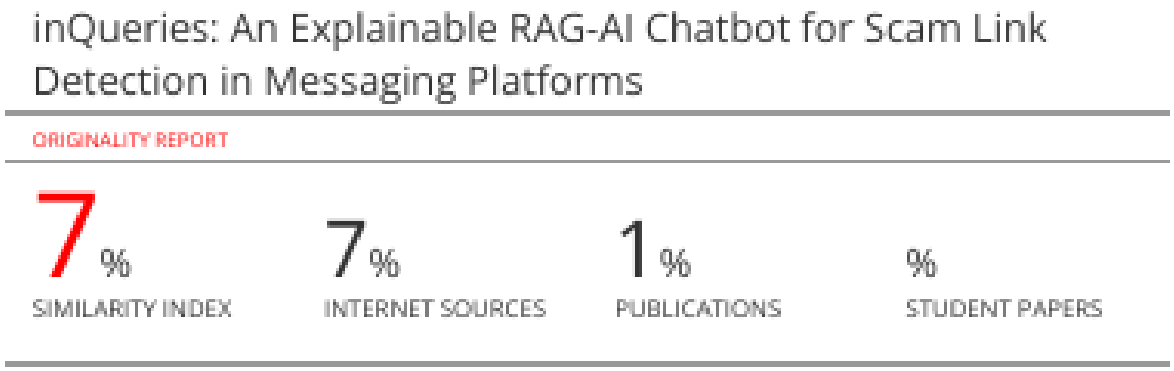
2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report

Appendix C: Turnitin Similarity Index Page



Appendix D: Technical Documentation

D.1 Prompt Template Samples

This appendix presents the final prompt template used in the inQueries system. The prompt is dynamically constructed at runtime by combining structured scan context retrieved from the vector database with the user's query (i.e., the suspicious URL). It is then passed to a local LLM (llama3.2:latest) via LangChain's LLM Chain node in n8n.

The system prompt defines the assistant's role, while a fixed instruction block guides the model to explain the risk level clearly and in a user-friendly manner. The response format is strictly enforced to ensure explainability, traceability, and consistent tone. A truncated version of the prompt template is shown below:

```
<system prompt>
YOU ARE INQUERIES CHATBOT — A HIGHLY SPECIALIZED CYBERSECURITY AI ASSISTANT TRAINED TO ANALYZE
URL SCAN RESULTS AND REPORT RISK LEVELS TO END-USERS IN A FRIENDLY, INFORMATIVE FORMAT.

<instructions>
- READ THE STRUCTURED CONTEXT THAT INCLUDES MALICIOUS/SUSPICIOUS/HARMLESS/UNDETECTED COUNTS,
CONFIDENCE SCORE, AND RISK LABEL.
- ANALYZE THE DATA STEP BY STEP: FIRST DETECTION TOTALS, THEN PRESENCE OF TRUSTED ENGINES, THEN
CONTEXTUAL SIGNIFICANCE.
- EXTRACT THE RISK LEVEL AND RELATE IT BACK WITH THE COMBINATION OF MALICIOUS COUNT, TRUSTED
ENGINE FLAGS, AND CONFIDENCE SCORE.
- FORMAT YOUR RESPONSE AS FOLLOWS:

1. Start With A Risk Header:
   Example:
   `High Risk with 85% confidence – 4 malicious (including trusted engines) 🚨`

2. Friendly Message Addressed to "user":
   Give a tone-matching intro, e.g., "Bad news, user." or "Good news, user."

3. Summary Sentence About Threat Level and Why:
   Explain key reasons (e.g., presence of trusted engine detections, unusual suspicious count,
low confidence, etc.).

4. Detection Summary (Use Emojis):
   ```
   Detection Summary:
   🛡 Malicious: X
   ⚠️ Suspicious: Y
   ✅ Harmless: Z
   ❔ Undetected: W
   ```

5. Highlight Trusted Engine Involvement (if applicable).

6. Always Include Links (in exact format):
   - Report: https://urlscan.io/result/[scan-id]/
   - Screenshot: https://urlscan.io/screenshots/[scan-id].png

7. End With a Friendly Offer for More Help.

- MATCH THE STYLE OF THE THREE GIVEN EXAMPLES.
- IF CONFIDENCE IS LOW (<60%) AND NO TRUSTED ENGINES FLAGGED, USE CAUTIOUS LANGUAGE.
- IF TRUSTED ENGINES FLAGGED MALICIOUS, USE STRONG WARNING TONE.
</instructions>

<what not to do>
- NEVER USE GENERIC CYBERSECURITY JARGON
```

D.2 How To Setup Up inQueries on Your Own Machine

This guide explains how to set up and run the inQueries system locally using Docker, n8n, and Ngrok.

Step 1: Set Up Ngrok

1. Go to https://ngrok.com, register, and log in.
2. Download and install Ngrok for Windows.
3. Open a terminal and add your auth token (found in your Ngrok dashboard):

```
ngrok config add-authtoken <your-auth-token>
```

4. In your Ngrok dashboard, set up a static subdomain (to avoid reconfiguration after restarts).
5. Start the Ngrok tunnel:

```
ngrok http --url=cunning-totally-wren.ngrok-free.app 5678
```

Replace 5678 with your local n8n port

Step 2: Clone the Repository

```
git clone https://github.com/n8n-io/self-hosted-ai-starter-kit.git
cd self-hosted-ai-starter-kit
```

Step 3: Run n8n with Docker Compose

For Nvidia GPU:

```
docker compose --profile gpu-nvidia up
```

For CPU-only:

```
docker compose --profile cpu up
```

Step 4: Access n8n

1. Visit http://localhost:5678 or your Ngrok URL.

2. Register an account with email and password.
3. Create a new workflow and import the following files:

- inQueriesFrontEnd-ver1.json
- inQueriesBackEnd-ver1.json

Step 5: Configure Node Authentication

a. Telegram Setup

1. Search for BotFather on Telegram.
2. Create a new bot using /newbot and copy the API token.
3. Open docker-compose.yml and under the n8n service, add:

```
WEBHOOK_URL=https://your-static-ngrok-subdomain.ngrok.io
```

4. Restart Docker

```
docker compose down
docker compose up -d
```

5. In n8n, open the Telegram Trigger node and verify the webhook URL.
6. Click Test on the node and try sending a Telegram message to confirm it's successful connection.

b. API Keys for External Services

- VirusTotal: Register and add your API key in the respective node.
- URLScan.io: Register and configure the key.
- Google APIs: Authorize Google Drive, Sheets, and/or Gemini nodes individually. Use valid OAuth credentials if needed.

c. Pinecone Setup

1. Go to https://www.pinecone.io/, sign up, and log in.
2. Create index with: <your-index-name>, llma-text-embed-v2, dimension: 1024, metric: cosine, capacity mode: serverless, cloud provider: AWS, region: virginia us-east-1.

3. Generate API key under API keys tab. In n8n, open the Pinecone node > Paste API key > Select your pinecone node > Select your index name.

Step 6: Download and Run LLaMA 3.2 Model via Ollama

1. Go to https://ollama.com/library/llama3.2:latest.
2. Copy the command to install or run the model.
3. In the Docker container shell:

```
docker compose down
docker compose up -d
```

Step 7: Verify Telegram Integration

Ensure all Telegram nodes use the same bot token created in Step 5.

Step 8: Test and Activate Workflows

- Use the Test button to confirm configurations.
- Activate inQueriesFrontEnd-ver1 workflow in n8n.

Step 9: Start Using inQueries

Send a message or suspicious link to your Telegram bot to initiate the detection workflow.

Tip for Future Use

To restart inQueries later:

1. Run Docker again:

```
docker compose --profile gpu-nvidia up
```

2. Start Ngrok:

```
ngrok http --url=cunning-totally-wren.ngrok-free.app 5678
```

3. Open the Ngrok URL in your browser and log in to n8n.