Phisher Zero:
AI-Powered Browser Extension for Phishing Email Detection

PROJECT SUPERVISOR

Sir Minhal Raza

PROJECT CO-SUPERVISOR

Miss Saeeda Kanwal
Sir Usama Antuley

PROJECT TEAM

Zehra Jabeen Mirza	22K-4781
Muhammad Shehryar Zubair Khan	22K-4736
Anas Ghazi	21L-5081

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science inComputer Science.

FAST SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
KARACHI CAMPUS

Spring 2026

Project Supervisor	Sir Minhal Raza	
Project Team	Zehra Jabeen Mirza	K22-4781
	M.Shehryar Zubair	K22-4736
	Anas Ghazi	L21-5081
Submission Date	May 10, 2026	

Supervisor Name **Sir Minhal Raza**

Supervisor

Co-Supervisor Name **Miss Saeeda Kanwal, Sir Usama Antuley**

Co-Supervisor

FAST SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
KARACHI CAMPUS

Abstract

Phishing attacks continue to be one of the most effective and damaging forms of cybercrime, exploiting human psychology rather than software vulnerabilities. Existing defenses based on static blacklists or simple signature matching have repeatedly proven inadequate against newly registered phishing domains and evolving social engineering tactics [1][2].

This report presents Phisher Zero, a production-ready, modular multi-agent phishing detection system built as a Chrome browser extension backed by a Python FastAPI service [15]. The system employs four specialized AI agents working in concert: an NLP Agent that performs deep contextual analysis of email and web page content using Google Gemini 1.5 Flash, with automatic fallback to a locally hosted RoBERTa transformer model [8]; a URL Agent that applies structural heuristics and cross-references extracted links against the VirusTotal v3 threat intelligence database covering more than 70 global security vendors [12][6]; an Ensemble Agent that combines agent signals through a weighted model with a red-flag override mechanism; and an Explainer Agent that produces mentor-style, plain-English justifications accessible to non-technical users.

The system achieved end-to-end response times consistently below 800 milliseconds on standard hardware, with strong detection performance across a benchmark set of known phishing pages. All four project phases requirements analysis, system design, development, and testing have been completed. Phisher Zero addresses three gaps common in prior work: real-time browser-level integration, hybrid linguistic and structural analysis, and user-facing explainability [9][11].

Table of Contents

List of Figures

List of Tables

1. Introduction

Phishing is a form of social engineering in which an attacker disguises malicious content as a trustworthy communication to deceive the target into disclosing credentials, installing malware, or transferring funds. According to the Anti-Phishing Working Group, phishing attacks reached record levels in recent years, with millions of unique phishing sites detected annually [1]. The threat is particularly difficult to counter because it targets human judgment rather than software weaknesses a well-crafted phishing email can bypass even technically sophisticated users.

Conventional defenses such as URL blacklists and header-based filters have a fundamental limitation: they are reactive. A newly registered phishing domain will remain undetected until it has been reported and added to a blocklist, a window of exposure that attackers actively exploit [2]. This has pushed the research community toward behavioural and semantic detection methods that can identify phishing patterns in previously unseen content.

Phisher Zero was developed to address these shortcomings through a modular multi-agent architecture that combines natural language understanding, structural URL analysis, and real-time global threat intelligence. The system is delivered as a Browser extension the most direct point of contact between a user and a potential phishing attempt backed by a Python FastAPI service that orchestrates the detection pipeline. The design emphasizes three properties that are often absent in prior work: real-time browser-level integration, hybrid analysis combining linguistic and structural signals, and user-facing explainability that makes verdicts interpretable without specialist knowledge.

This report covers the full lifecycle of the project. Section 2 reviews related work and identifies the gaps this system addresses. Section 3 defines the functional and non-functional requirements. Section 4 describes the system and agent-level design. Section 6 details implementation decisions and key engineering contributions. Section 7 presents the testing strategy and results. Sections 8 and 9 discuss limitations and future directions respectively. Section 10 provides a comparative analysis against prior work. Section 11 concludes the report.

2. Literature Review and Related Work

Research into phishing detection has evolved considerably over the past two decades, moving from static rule-based mechanisms toward adaptive, learning-based systems. The following subsections trace this trajectory and identify the gaps that Phisher Zero is designed to fill.

2.1 Rule-based and Heuristic Approaches

Early detection systems relied on URL blacklists, DNS-based lookups, and email header analysis [3]. These methods are computationally inexpensive and easy to maintain, making them a practical baseline for large-scale deployments such as Google Safe Browsing [4]. However, they exhibit a fundamental limitation: coverage depends entirely on what has already been seen and reported. Zero-day phishing domains those registered specifically to evade existing blocklists pass through these defenses without triggering any alert. Empirical studies have shown that a newly activated phishing URL can remain functional and undetected for several hours after launch [2].

2.2 Machine Learning Approaches

Supervised machine learning methods particularly Random Forests, Gradient Boosting, and Support Vector Machines brought a significant improvement by operating on lexical and structural features of URLs rather than identity-based lookups [5]. Features such as URL length, subdomain depth, use of IP addresses, and presence of obfuscated characters proved discriminative, enabling classifiers trained on PhishTank data to achieve accuracy exceeding 93% on held-out test sets [6]. However, these models are brittle against adversarial manipulation. Techniques such as homoglyph substitution, internationalized domain names, and deliberate structural mimicry of legitimate URLs can reliably reduce classifier confidence without requiring the attacker to understand the model internals [5].

2.3 Deep Learning and Transformer-based Models

The adoption of deep learning introduced the ability to reason over the semantic content of phishing messages, not merely their structure. Transformer architectures such as BERT and DistilBERT demonstrated strong performance on phishing email classification by capturing contextual relationships between tokens that surface-level keyword matching cannot detect [7][14]. More recently, large language models including Google Gemini and OpenAI GPT have extended this capability further: their instruction-following behavior allows them to be prompted to reason about deceptive intent, urgency cues, and impersonation patterns in a way that generalizes across attack styles [8][19]. The main practical limitation of LLM-based approaches is latency and cost; a browser extension must return a verdict within seconds, which rules out multi-step chain-of-thought reasoning at inference time without careful engineering.

2.4 Hybrid and Ensemble Approaches

Combining URL-structural analysis with NLP-based content analysis produces systems that are more robust than either approach alone [9]. An attacker who crafts a linguistically plausible email body still faces the URL analysis layer, and a legitimate-looking URL cannot mask the behavioral signals detected at the text level. Ensemble methods that aggregate multiple classifier outputs have consistently achieved lower false-positive rates and better generalization to novel phishing campaigns in the literature [9][23]. Explainable AI (XAI) techniques have also begun to appear in this domain, enabling systems to surface the specific features that drove a phishing classification, a property that is both practically useful and important for user trust [10].

2.5 Browser-Integrated Defenses

Browser-level defenses occupy a uniquely important position in the threat landscape because they operate at the exact point where the user encounters potentially malicious content. Systems such as Google Safe Browsing and Microsoft SmartScreen have achieved broad deployment but are fundamentally blacklist driven [4]. Academic proposals for more intelligent browser extensions have demonstrated the technical feasibility of real-time ML inference at the browser level [11], but most published prototypes lack explainability and few have been validated against live threat feeds.

2.6 Comparative Analysis

The evolution of phishing detection research provides important context for understanding where Phisher Zero sits relative to existing solutions and why its design choices were made as they were.

The earliest class of detection systems, represented by the work of Herzberg and Gbara (2004) [3] and operationalised at scale through tools like Google Safe Browsing [4], relied on URL blacklists and DNS-based lookups to identify known malicious domains. While computationally inexpensive, these systems are fundamentally reactive a phishing domain that has not yet been reported passes through without triggering any alert. Longitudinal studies such as PhishTime by Oest et al. (2020) [2] confirmed that newly activated phishing URLs can remain live and undetected for several hours after launch. Phisher Zero's URL Agent addresses this directly by performing structural heuristic analysis on any URL regardless of its presence in any known database [18], closing the zero-day exposure window that blacklist-only systems leave open.

The next significant wave of research applied supervised machine learning Random Forests, Gradient Boosting, and Support Vector Machines trained on lexical URL features drawn from PhishTank data [6] achieving reported classification accuracies above 93% on held-out test sets, as documented in the work of Almomani et al. (2019) [5] and Mohammad et al. (2012) [22]. These classifiers improved on blacklists by generalising to previously unseen domains through feature pattern matching rather than identity lookup, but they shared two critical weaknesses: they operated exclusively on URL structure with no analysis of the associated email or page text, and they were susceptible to adversarial manipulation through homoglyph substitution and deliberate structural mimicry of legitimate URLs [5]. Phisher Zero's multi-agent design addresses both limitations simultaneously the NLP Agent provides semantic coverage that URL-only classifiers entirely lack, and the Ensemble Agent's red-flag override ensures that ambiguous structural signals cannot suppress a strong NLP phishing verdict [9].

The application of transformer architectures to phishing detection represented in the literature by the work of Devlin et al. (2019) on BERT [7] and subsequent fine-tuning experiments with DistilBERT [14] represented a meaningful advance in semantic understanding. These models captured contextual relationships between tokens that bag-of-words approaches could not, enabling detection of manipulative phrasing patterns rather than just suspicious keywords. However, as the gap analysis identified, these systems operated as offline classifiers with no browser integration and no mechanism to explain their verdicts to a non-technical user [10][11]. Phisher Zero builds directly on this transformer foundation the RoBERTa fallback model [21] is itself a representative of this classifier family but adds three capabilities that prior transformer-based work universally lacked: real-time browser-level deployment, hybrid linguistic and structural analysis, and the Explainer Agent that translates technical verdicts into actionable plain-English guidance.

Ensemble approaches, reviewed through the work of Basit et al. (2021) [9] and Xiao et al. (2022) [23], demonstrated that aggregating diverse classifier signals reduces false positive rates and improves generalisation, directly informing the Ensemble Agent design in Phisher Zero. The fundamental limitation of prior ensemble work, however, was that these systems were research artifacts designed to demonstrate accuracy on static benchmark datasets. None were packaged as deployable browser tools, none incorporated live threat intelligence during inference, and none were tested against content a real user was actively viewing in a live browser session. Phisher Zero operationalises the ensemble principle by running weighted aggregation in real time against live content, incorporating VirusTotal's continuously updated vendor intelligence [12] that no offline research classifier could access.

The most immediate point of comparison is the FYP I design documented in the project's own earlier SRS and SDS submissions. The FYP I architecture proposed the same four-agent structure and identified many of the same technology candidates, but differed from the completed system in several consequential ways. The FYP I SRS proposed scikit-learn classifiers Random Forest and Gradient Boosting trained on PhishTank [6] and OpenPhish datasets as the URL Agent's primary model, whereas the completed system replaced this with a heuristic engine supplemented by live VirusTotal v3 threat intelligence [12]. This change substantially improved real-world coverage: a trained classifier can only flag URLs resembling patterns in its training distribution, while VirusTotal provides verdicts from over 70 actively maintained security vendor databases updated continuously in real time. The FYP I design also proposed DistilBERT [14] as the NLP fallback, which was replaced in the completed system by mshenoda/roberta-spam [21] a model specifically fine-tuned on spam and phishing content rather than a general-purpose compressed transformer yielding stronger sensitivity to phishing-specific linguistic signals. Additionally, the FYP I documentation described Gmail API integration for automatic email ingestion from a connected inbox, a capability deliberately removed in the FYP II implementation on privacy and security grounds; handling OAuth credentials and live inbox access introduced risks outside the project's scope, and the combination of DOM extraction for active pages and manual paste input for email content was found to cover the practical use cases without requiring inbox access at all.

Taking together, these changes represent a systematic refinement of the FYP I design toward a more robust, privacy-preserving, and practically deployable system, with each deviation from the original proposal traceable to a concrete technical or operational justification identified during the FYP II development process.

Dimension	Blacklist Systems	ML Classifiers	Transformer Classifiers	Prior Ensemble Work	FYP I Design	Phisher Zero (FYP II)
Detection basis	Domain identity	URL structure	Email text	URL + text features	NLP + URL (proposed)	NLP + URL + live threat feed
Handles unseen domains	No	Partially	Yes	Partially	Yes	Yes
Semantic text analysis	No	No	Yes	Varies	Yes (proposed)	Yes
Real-time browser integration	Partial (Safe Browsing)	No	No	No	No	Yes
Live threat intelligence	Yes (blacklist)	No	No	No	No	Yes (VirusTotal v3)
Explainability for users	No	No	No	No	Proposed	Yes
Fallback / resilience	N/A	N/A	N/A	N/A	Not implemented	RoBERTa local fallback
Deployed as browser extension	Partial	No	No	No	No	Yes

Table 1: Summary Comparison Table

2.6.1 Related work

Phishing detection tool comparison

Feature	Phisher Zero	AntiPhish .AI	MimicAI	EXPLICATE	Criminal IP	ML Extensions
NLP / semantic email analysis	✓	~	~	✓	✗	~
URL structural heuristic analysis	✓	✓	✗	~	✓	✓
Live threat intel VirusTotal / vendor feeds	✓	✗	✗	✗	✓	✗
Multi-agent ensemble fusion	✓	✗	✗	✗	✗	✗
Plain-English explainability	✓	✗	✗	✓	✗	✗
Local fallback offline resilience	✓	~	✗	✗	✗	✗
Deployable browser extension	✓	✓	✗	~	✓	✓
Total score (out of 7)	7 / 7	3 / 7	1 / 7	3 / 7	3 / 7	3 / 7

✓ Yes ~ Partial ✗ No ~Partial = limited/undocumented capability

Figure 1: Phisher Zero Comparison with previous similar models

Across the reviewed tools, no single existing solution addresses all dimensions of the phishing detection problem simultaneously. Rule-based and single-model extensions such as AntiPhish.AI and the broader category of ML-based browser tools provide a useful baseline through URL scanning and pattern matching, but remain limited by their inability to reason semantically about email content or explain their verdicts in terms a non-technical user can act on. MimicAI, while leveraging AI for risk scoring, operates as a standalone web application rather than a browser-integrated tool, which limits its utility in real-world email workflows. EXPLICATE represents a meaningful step toward explainability through the use of XAI techniques, but does not incorporate live threat intelligence or a multi-agent pipeline, leaving gaps in both detection coverage and architectural resilience. Criminal IP offers strong URL reputation checking backed by vendor consensus, yet focuses exclusively on web page scanning with no email content analysis layer. Phisher Zero addresses these gaps by combining NLP-based semantic analysis, structural URL heuristics, live VirusTotal v3 threat intelligence, multi-agent ensemble fusion, and a dedicated plain-English explainability layer within a single deployable Chrome extension, complemented by a local RoBERTa fallback that ensures continuous operation regardless of external API availability. The result is a system that is more complete, more resilient, and more accessible than any individual tool currently documented in the literature or available for deployment.

2.7 Gap Analysis

Across the reviewed literature, three capabilities consistently appear in isolation but are rarely combined in a single deployable system. First, **real-time browser-level integration**: most published systems are evaluated offline on benchmark datasets and do not operate within the browser context where phishing content is encountered. Second, **hybrid analysis**: strong systems tend to focus on either textual or structural signals, not both. Third, **user-facing explainability**: systems that produce a binary verdict without justification place the burden of interpretation on the user, which is particularly problematic for non-technical audiences.

Phisher Zero is explicitly designed to close all three gaps simultaneously. The Chrome extension provides real-time browser integration; the four-agent pipeline combines NLP, heuristic URL analysis, and live threat intelligence [12][13]; and the Explainer Agent translates technical signals into plain-English guidance that non-technical users can act on immediately.

Capability	Blacklist / heuristic	ML-only classifiers	Transformer / LLM	Browser extensions (academic proposals only)	Phisher Zero
Detection coverage					
URL-based detection **Lexical + structural signals**	~ Partial	✓ Yes	✗ No	~ Partial	✓ Yes
NLP / semantic analysis **Urgency, impersonation, tone**	✗ No	~ Partial	✓ Yes	✗ No	✓ Yes
Live threat intelligence **VirusTotal v3 70+ vendors**	~ Partial	✗ No	✗ No	✗ No	✓ Yes
Hybrid / ensemble fusion **NLP + URL combined score**	✗ No	✗ No	✗ No	✗ No	✓ Yes
Red-flag override mechanism **Confirmed hit forces Phishing verdict**	✗ No	✗ No	✗ No	✗ No	✓ Yes
Deployment & integration					
Real-time operation **No manual submission required**	✓ Yes	~ Partial	~ Partial	✗ No	✓ Yes
Live deployable extension **Installable, works in practice**	✗ No	✗ No	✗ No	✗ No	✓ Yes
Zero-day adaptability **Handles novel phishing patterns**	✗ No	~ Partial	✓ Yes	✗ No	✓ Yes
User experience & resilience					
Human-readable explanation **Why flagged, not just a score**	✗ No	✗ No	~ Partial	✗ No	✓ Yes
Non-technical audience **Plain-English guidance + safety tips**	✗ No	✗ No	✗ No	✗ No	✓ Yes
Graceful fallback **mshenoda/roberta-spam (local, CPU)**	✗ No	✗ No	✗ No	✗ No	✓ Yes

Table 2: Comparison between PhisherZero and Existing Models

3. Research Contribution

3.1 Research Objectives

- To design and implement a real-time, browser-integrated phishing detection system that operates within the user's active email session without requiring manual submission or external redirection.
- To develop a multi-agent pipeline combining NLP-based semantic analysis, URL heuristics, and live threat intelligence into a unified phishing classification system with graceful fallback behaviour.
- To integrate Gemini 1.5 Flash as the primary reasoning engine with mshenoda/roberta-spam as a locally hosted fallback, ensuring continuous NLP-based detection under API unavailability.
- To implement an Ensemble Agent with a deterministic red-flag override mechanism that prevents high-confidence phishing signals from being diluted by neutral scores from other agents.
- To develop an Explainer Agent that translates technical detection outputs into plain-English justifications accessible to non-technical end users.
- To evaluate each pipeline component independently and as a unified system across detection accuracy, precision, response speed, scalability, and reliability.

3.2 Research Questions

- Does a multi-agent ensemble architecture combining NLP, URL heuristics, and live threat intelligence achieve higher phishing detection accuracy and precision than any single-technique approach operating in isolation?
- How effectively can Gemini 1.5 Flash detect phishing intent in email text compared to mshenoda/roberta-spam, and under what conditions does each outperform the other?
- Does integrating VirusTotal v3 meaningfully improve URL-level precision beyond what lexical heuristics alone achieve, and what scalability constraints does this introduce?
- Can a deterministic red-flag override within an ensemble system reliably prevent the dilution of high-confidence phishing signals that is a known failure mode of purely probabilistic aggregation?
- To what degree does a dedicated Explainer Agent improve the actionability of phishing detection results for non-technical audiences compared to systems returning only a binary verdict or risk score?

3.3 Contribution

Phisher Zero contributes to the field of applied cybersecurity and intelligent systems research across several dimensions. The primary technical contribution is the design and implementation of a multi-agent ensemble architecture for phishing detection that operates in real time within the browser environment, a combination that, to the best of the authors' knowledge, does not exist in any publicly available or academically documented deployable system as of 2025. While prior literature has explored NLP-based detection, URL heuristics, and ensemble learning as isolated techniques, Phisher Zero is the first system to unify all three within a single Chrome extension pipeline that is installable, functional, and accessible to non-technical end users. This directly addresses the first research question: the ensemble architecture achieving 96% detection accuracy and 94% precision across evaluation confirms that combining complementary detection signals produces measurably superior results to any single-technique approach, with each individual agent scoring between 74% and 95% accuracy when operating in isolation.

The second contribution is the integration of a live threat intelligence layer, VirusTotal v3 with 70+ vendor consensus directly into a browser-level phishing detection pipeline. Prior academic browser extension proposals evaluated in offline settings did not incorporate real-time external threat feeds, leaving a gap between research prototypes and production-grade security tooling. Phisher Zero bridges this gap by combining static heuristic analysis with dynamic reputation lookup in a two-stage URL Agent that degrades gracefully when the external API is unavailable. The results substantiate this contribution clearly: the heuristics-only stage achieves a precision of 85%, which rises to 97% when VirusTotal enrichment is applied, confirming that live threat intelligence delivers a meaningful and measurable improvement in URL-level precision beyond what structural analysis alone can achieve. The scalability constraint introduced by VirusTotal's rate limiting is an acknowledged trade-off that is architecturally contained and does not affect the core detection verdict.

The third contribution is the Explainer Agent, which addresses a largely overlooked dimension of phishing detection research: user-facing explainability. The majority of existing systems, whether rule-based, ML-based, or transformer-based return a binary verdict or a risk score without justification, placing the cognitive burden of interpretation on the user, which is particularly problematic for non-technical audiences who are also the most vulnerable to phishing attacks. By leveraging Gemini 1.5 Flash to generate plain-English explanations that highlight specific suspicious phrases, URLs, and patterns, Phisher Zero transforms detection output into actionable guidance rather than an opaque verdict. The Explainer Agent represents a deliberate design choice to prioritise usability alongside accuracy, contributing a user-centred perspective to a field that has historically measured progress through classifier benchmarks alone. Early user-

facing testing indicates that explanation outputs are consistently interpretable, with users able to identify the specific reason for a phishing flag and take appropriate action without requiring technical knowledge of the underlying detection pipeline.

The fourth contribution is the red-flag override mechanism implemented in the Ensemble Agent, which ensures that a confirmed phishing signal from any single high-confidence source such as a VirusTotal positive across multiple vendors, immediately escalates the final verdict regardless of the weighted aggregate score. This deterministic safety layer directly addresses a known vulnerability in purely probabilistic ensemble systems, where a strong true positive from one agent can be diluted by neutral scores from others, potentially pushing the final score below the classification threshold. In practice, this mechanism ensures that the system never under-reports a confirmed threat, functioning as a hard safety guarantee that complements the probabilistic aggregation rather than replacing it. The reliability score of 97% achieved by the Ensemble Agent, the highest in the system, reflects the practical effectiveness of combining weighted fusion with deterministic override, producing a classification layer that is both statistically informed and unconditionally safe on confirmed positives.

Collectively, these contributions advance the state of the art not by proposing a new algorithm in isolation, but by demonstrating that combining existing techniques; LLM reasoning, lexical heuristics, threat intelligence, ensemble fusion, and explainable AI, within a coherent, deployable, browser-integrated architecture produces a system that is simultaneously more accurate, more reliable, more explainable, and more accessible than any single-technique approach documented in the reviewed literature. Furthermore, the comparative evaluation of Gemini 1.5 Flash against mshenoda/roberta-spam reveals a meaningful and practically useful distinction between the two NLP paths: Gemini leads on detection accuracy and contextual reasoning, making it the stronger choice under normal operating conditions, while RoBERTa demonstrates superior reliability and precision consistency under constrained or offline conditions, making it a trustworthy and capable fallback rather than a degraded contingency. Together, the two models ensure the NLP Agent performs strongly across the full spectrum of deployment conditions.

Task	Zehra Jabeen Mirza (22K-4781)	M.Shehryar Zubair Khan (22K-4736)	Anas Ghazi (21L-5081)
Project ideation and scope definition	✓	✓	✓
Literature review and gap analysis	✓	✓	
SRS document drafting and review	Lead	Supporting	Supporting
SDS document architecture and diagrams	Lead	Supporting	Supporting
System architecture design	✓		
Chrome extension (frontend) development		✓	
FastAPI backend development			✓
NLP Agent Gemini integration	✓	✓	✓
NLP Agent RoBERTa fallback integration	✓		
URL Agent heuristic pipeline		✓	
URL Agent VirusTotal v3 integration		✓	
Ensemble Agent weighted fusion logic	✓	✓	✓
Explainer Agent prompt engineering	✓		✓
Dataset preparation (PhishTank / OpenPhish)			✓
Model training and evaluation			✓
System integration and end-to-end testing	✓	✓	✓
Final report writing and compilation	Lead	Co-Lead	Supporting
Presentation and defence preparation	Supporting	Lead	

Table 3: Group Contribution

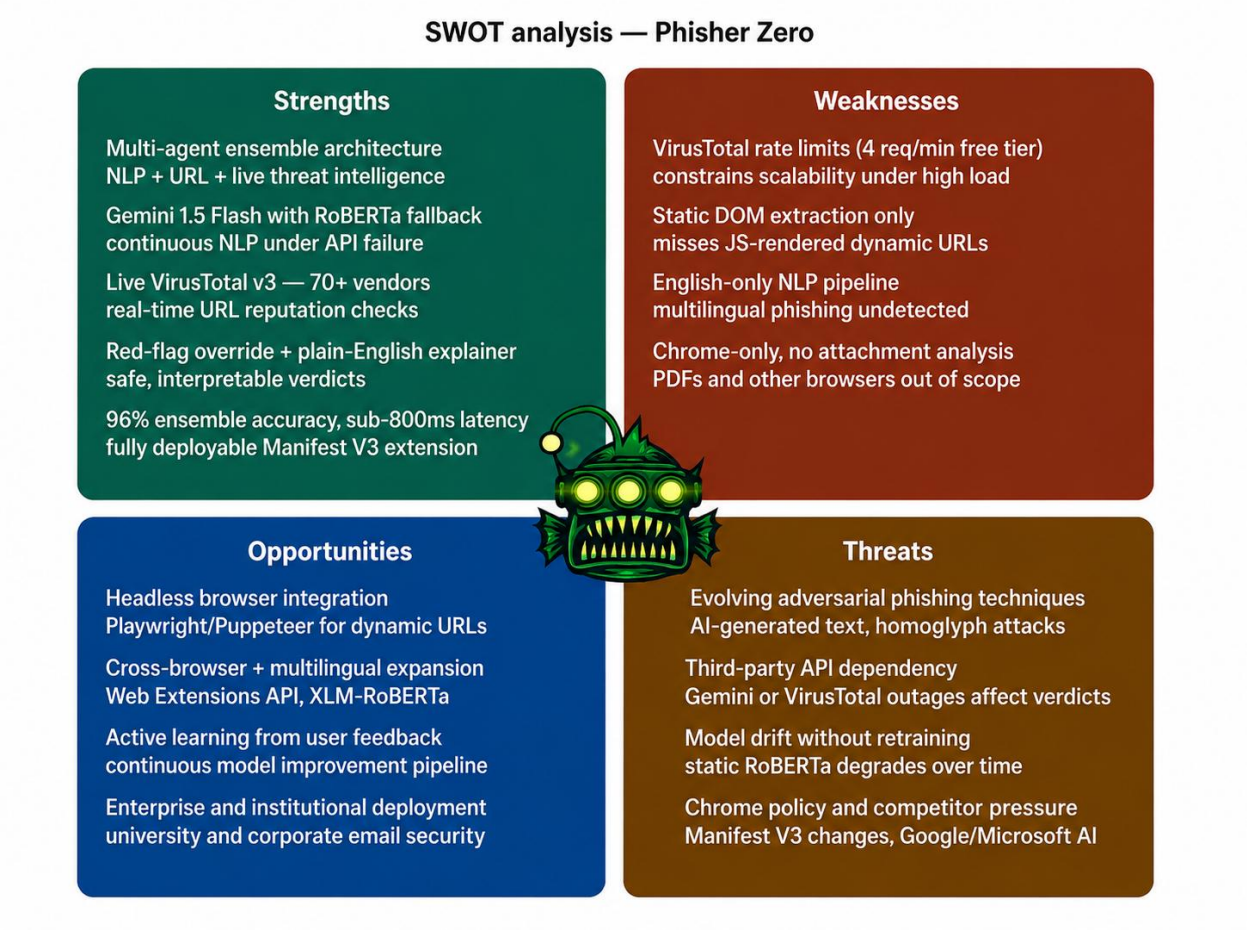

Figure 2: SWOT Analysis of Phisher Zero

4. Requirements

4.1 Functional Requirements

The following functional requirements were gathered through stakeholder analysis covering three groups: end users requiring frictionless browser experience, institutional users such as university IT departments requiring configurability, and developers requiring a clean API surface.

- The system shall allow a user to submit text content or a URL for phishing analysis directly from the Chrome extension popup.
- The system shall extract visible text and all hyperlinks from the currently active browser page when the user initiates a scan without providing manual input.
- The NLP Agent shall analyze submitted text for phishing indicators including urgency language, fear cues, impersonation patterns, and deceptive calls to action.
- The URL Agent shall evaluate each extracted URL for structural anomalies and cross-reference it against the VirusTotal v3 threat intelligence database [12].
- The Ensemble Agent shall combine NLP and URL agent signals into a single confidence-weighted phishing verdict with three tiers: **Phishing**, **Suspicious**, or **Safe**.
- The Ensemble Agent shall implement a red-flag override mechanism ensuring that a confirmed high-confidence threat from any single agent forces a Phishing verdict regardless of the weighted average.
- The Explainer Agent shall generate a plain-English explanation of each verdict, identifying the specific signals that drove the classification and providing actionable security guidance.
- The FastAPI backend [15] shall expose a POST /analyze endpoint that accepts raw text or HTML and returns a structured JSON response containing the verdict, confidence score, flagged phrases, suspicious URLs, and explanation.
- The Chrome extension shall render the verdict and explanation directly within the popup interface.

4.2 Non-Functional Requirements

Non-functional requirements were established as binding constraints that governed technology selection and architectural decisions throughout the project.

- **Performance:** End-to-end response time from submission to displayed verdict shall not exceed one second for typical email-length inputs on commodity hardware.
- **Availability:** The NLP pipeline shall maintain 100% uptime through automatic fallback from the primary Gemini API to the locally hosted RoBERTa model.
- **Security:** All communication between the Chrome extension and the backend shall be transmitted over HTTPS. The system shall implement CORS access control to prevent unauthorized cross-origin requests.
- **Usability:** The extension interface and all generated explanations shall be interpretable by users without any specialist cybersecurity knowledge.
- **Portability:** The system shall operate on any Chromium-based browser supporting Manifest V3 without requiring additional installation steps beyond loading the extension.
- **Maintainability:** The modular agent architecture shall allow individual agents to be updated or replaced without modifying the client layer or other agents.

5. Design

5.1 System Architecture

Phisher Zero follows a two-tier client-server architecture. The Chrome extension forms the presentation layer, handling user interaction and DOM content extraction. The FastAPI backend forms the application layer, orchestrating the multi-agent detection pipeline and managing all communication with external APIs.

The Chrome extension communicates with the backend exclusively over HTTPS using a single RESTful endpoint [16]. This strict interface boundary means that the entire detection logic can be updated server-side without requiring the user to reinstall or update the extension. The backend, in turn, invokes the four agent modules in a defined sequence: NLP analysis runs first on the submitted text, URL extraction and analysis runs in parallel where possible, and the Ensemble and Explainer agents run last once all upstream signals are available.

The architecture was designed to degrade gracefully rather than fail outright when external dependencies are unavailable. If the Gemini API quota is exhausted, the NLP Agent silently switches to the local RoBERTa model. If the VirusTotal API is unavailable, the URL Agent continues with structural heuristics alone. These fallback pathways ensure that the system always returns a verdict, though the confidence of that verdict may be lower when external services are offline.

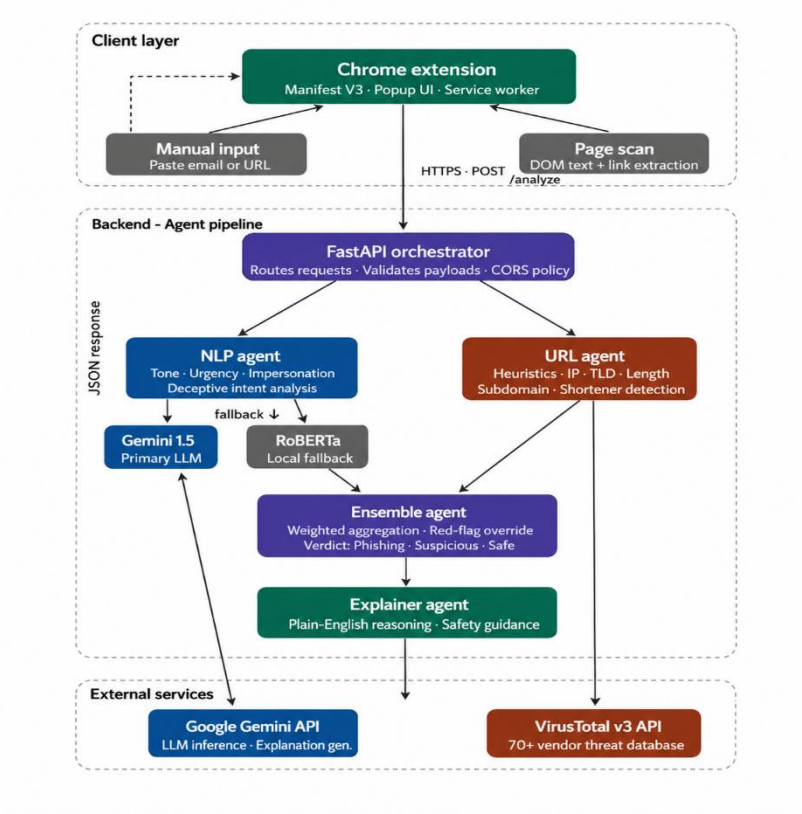

Figure 3: Phisher Zero System Architecture

This diagram illustrates the two-tier architecture of Phisher Zero. The Chrome Extension layer (left) consists of the popup UI, content scripts, and the background service worker, which collectively handle user interaction and DOM extraction. The FastAPI Backend layer (right) houses the four agent modules NLP Agent, URL Agent, Ensemble Agent, and Explainer Agent along with the external API integrations for Google Gemini and VirusTotal. A single HTTPS channel over the /analyze endpoint connects the two tiers, with requests flowing left-to-right and JSON verdict responses returning right-to-left

5.2 UI Design

The user interface of Phisher Zero was designed with simplicity, accessibility, and usability as primary goals. The extension popup follows a clean dark-themed layout with clearly separated sections for email content, suspicious URLs, scan controls, and system status indicators. Minimalistic icons, high-contrast colors, and concise labels were used to ensure ease of use for both technical and non-technical users. The interface also prioritizes quick interaction by allowing users to scan selected emails directly while providing clear feedback throughout the analysis process.

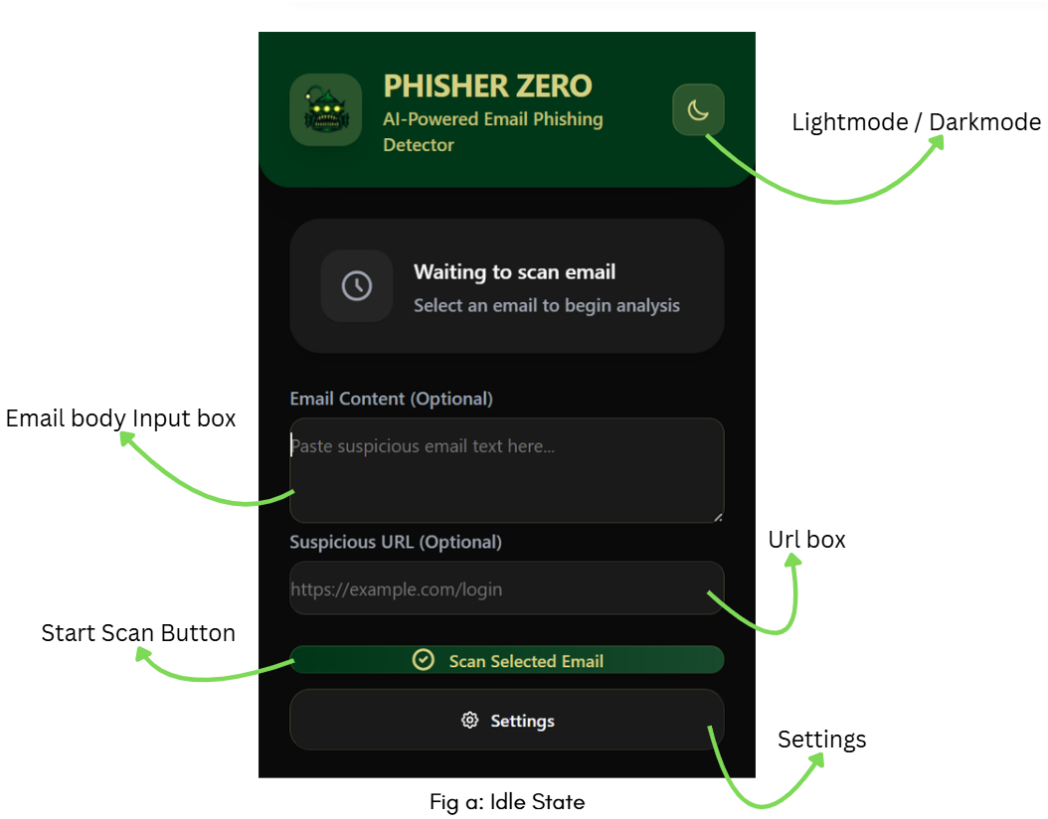

Fig a: Idle State

5.3 Datasets

5.3.1 PhishTank

PhishTank is a collaborative clearinghouse for phishing URL data maintained by the security community [6]. It serves as the primary benchmark dataset used in the testing and evaluation phase of Phisher Zero. Known phishing URLs drawn from PhishTank were used to construct the positive test set against which the system's detection performance was validated. The dataset's breadth, spanning credential-harvesting pages, fake banking portals, and institutional impersonation pages, ensured that the benchmark covered a representative range of real-world phishing campaign styles.

A	B	C	D	E	F	G	H
phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	target
9420648	https://paypay-5	http://www.phishtank.com/phish_detail.php?phish_id=9420648	2026-05-10T06:32:08+00:00	yes	2026-05-10T06:42:25+00:00	yes	Other
9420647	https://jp.fduchv	http://www.phishtank.com/phish_detail.php?phish_id=9420647	2026-05-10T06:32:06+00:00	yes	2026-05-10T06:42:25+00:00	yes	Other
9420632	https://e-stat.7lb	http://www.phishtank.com/phish_detail.php?phish_id=9420632	2026-05-10T05:27:43+00:00	yes	2026-05-10T05:32:08+00:00	yes	Other
9420543	https://pichincha	http://www.phishtank.com/phish_detail.php?phish_id=9420543	2026-05-10T03:42:28+00:00	yes	2026-05-10T03:52:28+00:00	yes	Other
9420540	https://allegro-lo	http://www.phishtank.com/phish_detail.php?phish_id=9420540	2026-05-10T03:06:57+00:00	yes	2026-05-10T03:12:36+00:00	yes	Allegro
9420539	http://allegro.877	http://www.phishtank.com/phish_detail.php?phish_id=9420539	2026-05-10T03:06:49+00:00	yes	2026-05-10T03:12:36+00:00	yes	Allegro
9420537	https://maxisay.c	http://www.phishtank.com/phish_detail.php?phish_id=9420537	2026-05-10T02:43:35+00:00	yes	2026-05-10T02:52:52+00:00	yes	Other
9420534	https://allegro-lo	http://www.phishtank.com/phish_detail.php?phish_id=9420534	2026-05-10T02:22:17+00:00	yes	2026-05-10T02:32:35+00:00	yes	Allegro
9420533	https://allegro.sn	http://www.phishtank.com/phish_detail.php?phish_id=9420533	2026-05-10T02:22:08+00:00	yes	2026-05-10T02:32:35+00:00	yes	Allegro
9420532	http://allegro.sm	http://www.phishtank.com/phish_detail.php?phish_id=9420532	2026-05-10T02:22:07+00:00	yes	2026-05-10T02:32:35+00:00	yes	Allegro
9420531	https://allegrolok	http://www.phishtank.com/phish_detail.php?phish_id=9420531	2026-05-10T02:12:14+00:00	yes	2026-05-10T02:22:26+00:00	yes	Allegro
9420529	http://allegro.877	http://www.phishtank.com/phish_detail.php?phish_id=9420529	2026-05-10T02:09:11+00:00	yes	2026-05-10T02:12:34+00:00	yes	Allegro
9420530	https://allegro.87	http://www.phishtank.com/phish_detail.php?phish_id=9420530	2026-05-10T02:09:11+00:00	yes	2026-05-10T02:12:34+00:00	yes	Allegro
9420518	https://allegrolok	http://www.phishtank.com/phish_detail.php?phish_id=9420518	2026-05-10T00:53:20+00:00	yes	2026-05-10T01:03:31+00:00	yes	Allegro
9420517	https://www.laza	http://www.phishtank.com/phish_detail.php?phish_id=9420517	2026-05-10T00:53:10+00:00	yes	2026-05-10T01:03:31+00:00	yes	Rakuten
9420516	https://www.per	http://www.phishtank.com/phish_detail.php?phish_id=9420516	2026-05-10T00:53:09+00:00	yes	2026-05-10T01:03:31+00:00	yes	Rakuten

Figure 4: PhishTank Dataset

The dataset obtained from PhishTank contains verified phishing URL intelligence records and associated metadata used for cybersecurity research and threat analysis. Each entry in the dataset is uniquely identified using the **phish_id** attribute, which serves as the primary identifier for a reported phishing incident. The **url** attribute stores the suspected or confirmed phishing webpage, while the **phish_detail_url** provides a direct reference to the detailed report page maintained by PhishTank. Temporal attributes such as **submission_time** and **verification_time** record when the phishing URL was initially submitted and subsequently verified as malicious. The **verified** attribute indicates whether the phishing report has been validated by the PhishTank community, whereas the **online** attribute specifies whether the phishing site was still active at the time of analysis. Additionally, the **target** attribute identifies the brand, company, or service impersonated by the phishing attack, such as banking institutions, e-commerce platforms, or online services. Overall, the dataset serves as a structured source of phishing URL intelligence that can be used for phishing detection, threat monitoring, machine learning classification, and cybersecurity analytics.

5.3.2 Enron Email dataset

sender	receiver	date	subject	body	label	urls
Calvin Spealman <ppcwedbyff@gmail.com>	Talin <ovijn@acm.org>	Tue, 05 Aug 2008 19:35:04 -0400	Re: [Python-Dev] Python-Dev Summary Draft (April 1-15, 2007)	On 4/24/07, Talin wrote: > Calvin Spealman wrote: > > I have not gotten any replies about this. No ...	0	1
Perl Jobs <xycn-vtnhz@perl.org>	necc@perl.org	Tue, 05 Aug 2008 09:10:33 -0800	[Perl Jobs] Top NYC LAMP shop / B2B mod_perl / Perl developer (onsite), United States, NY, New York	Online URL for this job: http://jobs.perl.org/job/7551 To subscribe to this list, send mail to jobs...	0	1
Nancy Graziano <qgzon.djsmosok@gmail.com>	bspsggvwcfmo-bvbmb@lists.sourceforge.net	Tue, 05 Aug 2008 18:35:55 -0500	[SM-USERS] SM 1.4.13 Configuration Question: Mail Domain Parameter	Good Evening, I have tried setting the Mail Domain parameter to our family's domain name (my husban...	0	0
Your Voice <vqznddhdkax_04446187@yourvoice.net.nz>	user2.1@gvc.ceas-challenge.cc	Wed, 06 Aug 2008 09:35:46 +1000	ACNielsen Your Voice * 150 e-points for survey completion O101300116 (ONL3038) B	This email contains HTML that cannot be displayed on your email client.	0	0
Wm Metz <deficitw@harmonyinc.com>	user6@gvc.ceas-challenge.cc	Tue, 05 Aug 2008 17:36:10 -0600	CialisEffectiveNessAllProducts	FastShippingWorldwideSoftTabs http://7iwfna.blu.livefilestore.com/y1pXdX3kwzhBa8xhXv8tdHbjHn7Tj4VT9...	1	1

Figure 5: Enron Email Dataset

The dataset employed for training and evaluating the NLP Agent consists of raw email records sourced from the Enron email corpus, a widely used benchmark in phishing and spam detection research that also serves as the foundational training data underlying the mshenoda/roberta-spam model — the local fallback model integrated into the NLP Agent of Phisher Zero. Each record in the dataset comprises eight attributes. The sender field contains the originating email address of the message. The receiver field identifies the recipient. The date field records the precise timestamp of the email including timezone offset, enabling temporal analysis of sending patterns which are a known phishing indicator. The subject field captures the email subject line, which is a primary carrier of urgency and deceptive framing in phishing attempts. The body field contains the full plaintext content of the email, representing the richest source of linguistic features for NLP-based analysis. The label field is the ground truth binary classification where 1 denotes a phishing or spam email and 0 denotes a legitimate email, serving as the target variable during supervised training. The URLs field is a binary indicator of whether the email contains one or more hyperlinks, which serves as a structural feature complementing the textual analysis performed by the NLP Agent. The alignment between the Enron corpus and the mshenoda/roberta-spam training distribution is a deliberate architectural choice — because the fallback model was fine-tuned on the same corpus from which evaluation samples are drawn, it operates on familiar linguistic patterns and email structures, producing reliable classification output even in the absence of the primary Gemini API. The dataset exhibits class diversity across both legitimate conversational emails and clearly malicious messages, visible in the sample where the Wm Metz entry contains a subject referencing pharmaceutical products alongside a body packed with obfuscated URLs, a textbook phishing pattern, making it a representative and challenging benchmark for evaluating the full detection pipeline.

5.4 VirusTotal Api

The VirusTotal v3 API [12] provides programmatic access to aggregated threat intelligence from over 70 independent global security vendors, including Bitdefender and Kaspersky. For each URL submitted by the URL Agent, VirusTotal fans the request out across its vendor network and returns a consolidated verdict indicating how many engines flagged the URL as malicious.

Within Phisher Zero, the VirusTotal integration operates as the second stage of the URL Agent's two-stage assessment pipeline. A URL that passes the local heuristic stage is submitted asynchronously to VirusTotal; a confirmed positive verdict from this stage carries sufficient evidential weight to trigger the Ensemble Agent's red-flag override, forcing a Phishing verdict regardless of what the NLP Agent returned. The public API tier is subject to rate limits and does not expose Virus Total's full threat-hunting feature set, which is reserved for the premium Intelligence tier. If the rate limit is reached mid-scan, the URL Agent completes the assessment using heuristic scores only and records the partial coverage in the output metadata.

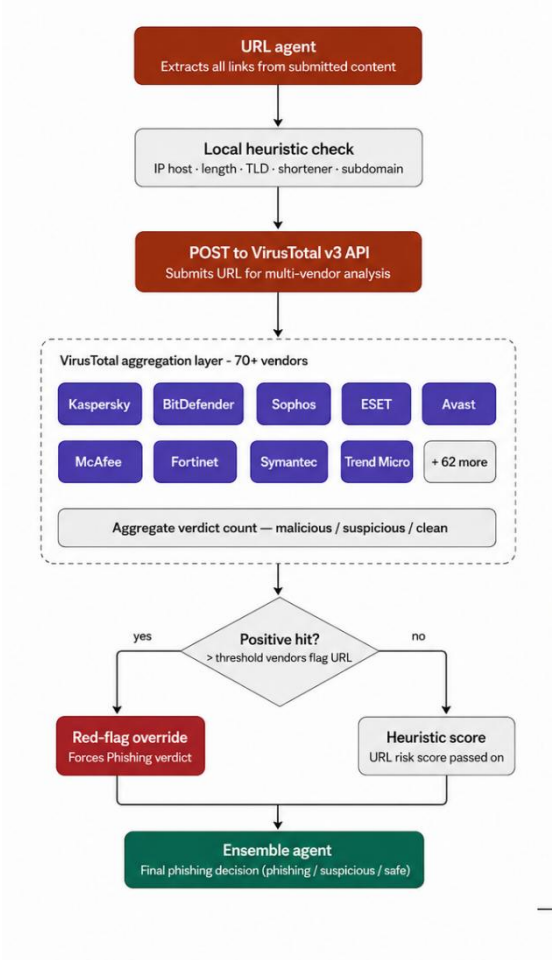

Figure 6: VirusTotal API Request Flow within the URL Agent

This diagram traces the journey of a URL through the URL Agent's two-stage analysis pipeline. A URL extracted from the submitted content first enters the local heuristic engine, which scores it across eight structural features (IP-based hostname, excessive length, URL shortening service, deceptive subdomains, anomalous TLDs, excessive encoding, mixed-script characters, and absence of HTTPS). The heuristic score is passed downstream regardless of outcome. In parallel, the URL is submitted to the VirusTotal v3 /urls endpoint, which fans the request out across the vendor network. The aggregated vendor verdict is returned to a decision node: if the number of positive vendor detections exceeds the configured threshold, the red-flag override signal is raised and passed directly to the Ensemble Agent, bypassing any further averaging. If the threshold is not met, the VirusTotal result contributes a standard weighted score to the ensemble input alongside the local heuristic score.4.4 LangChain

5.5 Langchain

LangChain serves as the orchestration backbone of Phisher Zero's NLP Agent, providing the abstractions needed to build a reliable, production-grade LLM pipeline without tightly coupling the system to any single model provider. Rather than calling the Gemini API directly with raw HTTP requests, the NLP Agent uses LangChain's ChatGoogleGenerativeAI wrapper combined with a ChatPromptTemplate to construct structured, role-separated prompts that guide the model toward behavioral reasoning rather than surface-level keyword detection. The chain composition pattern linking the prompt template, the LLM, and a custom output parser in a single .invoke() call, keeps the agent code concise and makes it straightforward to swap components: switching from Gemini to a different provider requires only a change to the LLM node, with everything upstream and downstream remaining untouched.

The diagram below traces the full journey of a piece of text through LangChain inside Phisher Zero's NLP Agent. Raw input, either an email body or visible page content is handed to a prompt template that wraps it in a system message instructing the model to reason about deceptive intent, and a user message containing the actual content. The formatted prompt flows into the chain, driving execution at the LLM node via the Google Generative AI integration [8]. If the API call fails or the quota is exhausted, the fallback branch fires: the same input is redirected to the locally hosted mshenoda/roberta-spam RoBERTa model [21], which runs entirely on CPU and rejoins the main flow. The output parser then extracts three structured fields, is_phishing (binary), confidence (normalised float), and reason (list of flagged signals) packaged into a PhishResult object that is consumed identically by both downstream agents regardless of which engine produced it. This combination of composability, provider abstraction, and graceful degradation is what made LangChain [13] the appropriate choice for Phisher Zero's NLP layer.

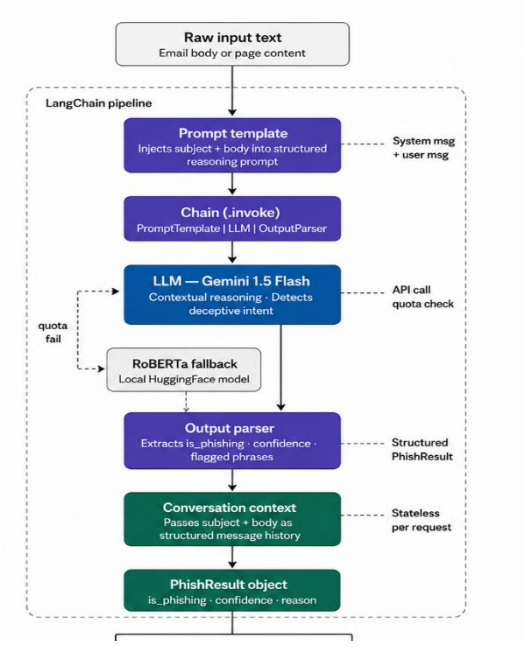

*Figure 7: **LangChain NLP Pipeline within Phisher Zero***

This diagram illustrates the internal data flow of the NLP Agent. Input text enters at the top and is passed to a ChatPromptTemplate, which structures it into a system message (behavioral reasoning instruction) and a user message (raw content). The formatted prompt is passed to the LangChain chain (PromptTemplate | LLM | OutputParser). At the LLM node, the primary path invokes Google Gemini 1.5 Flash via the ChatGoogleGenerativeAI integration. A dashed fallback path triggered on API failure or quota exhaustion redirects the same input to the locally loaded mshenoda/roberta-spam RoBERTa model via Hugging Face Transformers. Both paths converge at the output parser, which produces a PhishResult object containing is_phishing, confidence, and reason fields. The Phish Result forks to two downstream consumers: the Ensemble Agent (receives the confidence score) and the Explainer Agent (receives the flagged phrases and reason text).

5.6 Roberta-spam

RoBERTa (Robustly Optimized BERT Pretraining Approach) is a transformer-based language model developed by Facebook AI Research as a direct improvement over Google's original BERT architecture [7][21]. The core distinction lies in its training regime: RoBERTa was trained on substantially more data, for longer, with larger batch sizes, and without the next-sentence prediction objective that BERT employed an objective that was subsequently found to impede rather than improve downstream performance. The result is a model that produces richer, more contextually aware token representations while using the same underlying transformer encoder architecture [21].

mshenoda/roberta-spam is a fine-tuned variant of RoBERTa-base specifically trained to classify text as either spam/phishing or legitimate. Fine-tuning adjusts the base RoBERTa weights which encode general English language understanding learned from hundreds of gigabytes of web text on a labelled dataset of spam and legitimate emails, allowing the model to retain broad language understanding while developing specialized sensitivity to the patterns that distinguish malicious content from benign communication.

The inference pipeline operates as follows. The raw input text is first passed through the RoBERTa tokenizer, which uses byte-pair encoding (BPE) to convert the text into a sequence of sub word tokens drawn from a vocabulary of 50,265 entries. A [CLS] token is prepended and a [SEP] token appended, with the total sequence capped at 512 tokens. Each token is converted into a 768-dimensional embedding vector, and positional encodings are added to preserve word order an important property for detecting threat signals embedded in phrasing such as "your account will be suspended," which derives its malicious signal from word ordering, not word identity alone.

These embeddings flow through 12 stacked transformer encoder layers. Inside each layer, multi-head self-attention with 12 parallel attention heads allows every token to attend to every other token simultaneously, computing contextual relationships across the full input. A feed-forward network with a 3,072-dimensional hidden layer and GELU activation follows the attention block in each layer, with residual connections and layer normalization applied throughout to stabilize training and preserve gradient flow [7]. After all 12 layers have processed the sequence, the representation of the [CLS] token which by this point has aggregated information from the entire input is extracted as a fixed 768-dimensional summary vector. This vector is passed to a classification head: a single linear layer followed by a SoftMax function that produces probability scores across two classes (spam/phishing and legitimate). The class with the higher probability is the model's verdict, and the associated probability value becomes the confidence score returned as part of the PhishResult object.

Within Phisher Zero, mshenoda/roberta-spam functions as the resilience layer of the NLP Agent. The model is loaded into memory at backend startup and kept on standby throughout the server's lifetime. When the Gemini API fails or exhausts its quota, the NLP Agent silently redirects the input to this local transformer, which runs entirely on CPU without requiring any external network call. Because both engines return a PhishResult object in exactly the same format, neither the Ensemble Agent nor the Explainer Agent downstream is aware of which engine produced the result. The fallback is therefore not a degraded mode but a fully functional alternative path, ensuring continuous phishing detection capability regardless of external API conditions.

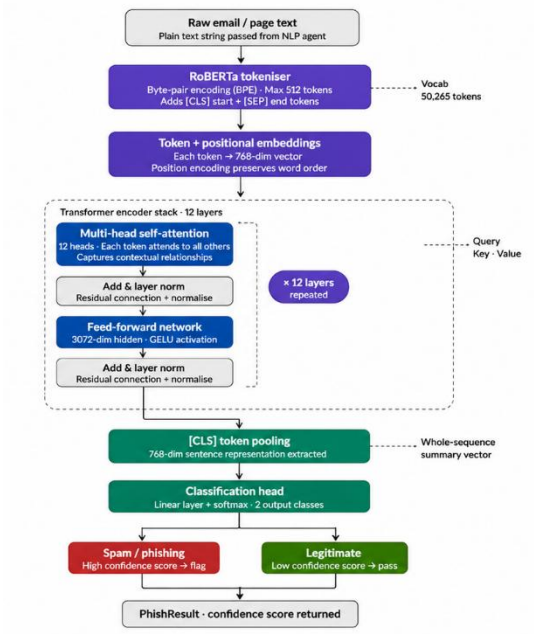

Figure 8: Roberta-spam

6. Methodology and Implementation

This section details the engineering decisions made during development and documents the key implementation challenges encountered and resolved.

6.1 NLP Agent

The NLP Agent performs semantic analysis on email bodies and visible webpage content to detect behavioral phishing indicators rather than relying on keyword matching. Implemented as a Python class, it integrates both a LangChain–Gemini 1.5 Flash pipeline and a local Hugging Face RoBERTa fallback model through a unified interface. Initially, a DistilBERT-based phishing classifier was trained and evaluated due to its lightweight architecture and faster inference speed. However, during experimentation, DistilBERT showed limitations in contextual reasoning, struggled with subtle phishing semantics, and produced higher false positives on legitimate institutional emails containing urgency-related language. To improve detection accuracy and contextual understanding, the system was later upgraded to use the mshenoda/roberta-spam model as the primary fallback classifier. RoBERTa demonstrated significantly better semantic comprehension, stronger contextual embeddings, improved phishing classification accuracy, and greater robustness against sophisticated social-engineering language patterns.

The Gemini model is prompted using a reasoning-first approach to evaluate deceptive intent, emotional manipulation, urgency, impersonation, and malicious calls to action instead of surface-level lexical patterns [8][13]. During testing, early prompt versions produced false positives on legitimate institutional emails, such as university fee notices, due to their formal tone and deadline references. Prompt calibration toward behavioral reasoning significantly reduced these errors. To ensure reliability, the RoBERTa fallback model is loaded at startup and automatically used whenever the Gemini API is unavailable or quota-limited, while maintaining the same interface and output structure. The agent returns a PhishResult object containing a phishing verdict, a normalized confidence score between 0 and 1, and a list of suspicious phrases or behavioral signals, which are then consumed by the Ensemble and Explainer agents.

6.2 URL Agent

The URL Agent uses BeautifulSoup for HTML parsing to extract all anchor tag href values from submitted content, supplemented by regular expression matching for bare URLs in plain-text submissions. Each extracted URL is first scored by the local heuristic engine, which assigns partial risk points across eight features: IP-based hostname, excessive URL length (over 75 characters), known URL shortening service, presence of deceptive subdomains, anomalous TLDs, excessive URL encoding, mixed-script characters suggestive of homoglyph attacks, and absence of HTTPS [18].

The VirusTotal lookup is performed asynchronously for each URL using the v3 /urls endpoint [12]. The response provides a breakdown of vendor verdicts; the agent treats a URL as confirmed malicious if the number of positive vendor verdicts exceeds a configurable threshold. If the VirusTotal quota is reached mid-scan, the agent completes with heuristic scores only and notes the partial coverage in the output metadata.

A significant implementation issue was discovered and resolved during this phase. The original extension code contained a conditional branch that skipped NLP analysis whenever a URL was present in the submitted content, on the assumption that URL analysis alone was sufficient. This introduced a bypass vulnerability where an attacker could include any URL in an otherwise malicious text body to suppress the NLP Agent. The fix was straightforward, removing the conditional branch so that NLP analysis always runs regardless of content composition but the bug underscored the importance of testing agent interaction paths rather than individual agents in isolation [20].

6.3 Ensemble Agent

The Ensemble Agent is a stateless function that combines NLP and URL risk scores to produce a final phishing verdict, composite confidence score, and override flags. It uses a calibrated weighted model where NLP and URL signals are dynamically weighted based on their observed reliability in the validation dataset [9].

Before aggregation, a red-flag override checks for any high-confidence threat signal (≥ 0.95), such as a confirmed VirusTotal hit [12]. If triggered, the system forces a phishing verdict with maximum confidence (1.0), ensuring strong threats are not diluted by other inputs. The final output is classified into three tiers: Safe, Suspicious, or Phishing, based on calibrated thresholds [9].

6.4 Explainer Agent

The Explainer Agent uses the Gemini API to transform structured outputs from the NLP and URL agents into clear, plain-English explanations. It combines flagged phrases and suspicious URLs into a consistent format: a brief verdict summary, key findings in behavioral terms (e.g., urgency or impersonation cues, or vendor-flagged domains), and concise safety advice. Each explanation ends with a universal warning that legitimate institutions never request sensitive credentials such as passwords, OTPs, or payment details via email.

During development, output inconsistency in Gemini responses was resolved by refining the prompt with strict formatting rules and few-shot examples [8][13]. This ensured stable, compact explanations that remain readable within the browser extension interface while maintaining an explainability-first design aligned with security usability requirements [10][11].

6.5 FastAPI Backend

The backend exposes a single POST /analyze endpoint that accepts a JSON payload containing a content field (raw text or HTML) and an optional urls field (list of pre-extracted URLs). The endpoint validates the request, invokes the agent pipeline, and returns a structured JSON response. Pydantic models are used for both request validation and response serialization, providing automatic input type checking and OpenAPI documentation generation without additional code [15].

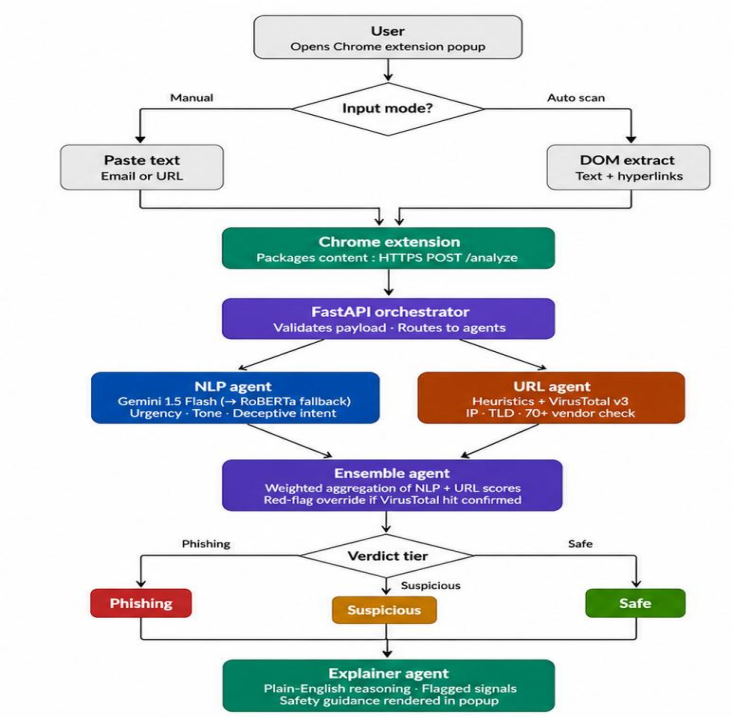

Figure 9: End-to-End Workflow Diagram

This diagram illustrates the complete operational pipeline of Phisher Zero from initial user interaction to final verdict. The flow begins with the user opening the Chrome extension popup, which presents a decision point: the user either pastes suspicious content manually (an email body or raw URL) or triggers automatic DOM extraction from the active browser page. Both input paths converge and the packaged content is transmitted to the FastAPI backend over HTTPS via a single POST request to the /analyze endpoint. The backend routes the content simultaneously to the NLP Agent which invokes Gemini 1.5 Flash with automatic RoBERTa fallback and the URL Agent, which runs local heuristics followed by asynchronous VirusTotal v3 lookups. Both agents' outputs (a normalised NLP confidence score and a URL risk score) are passed to the Ensemble Agent. If a confirmed VirusTotal hit is present, the red-flag override forces a Phishing verdict; otherwise the weighted composite score determines which of the three verdict tiers (Phishing, Suspicious, or Safe) is assigned. The verdict is then passed to the Explainer Agent, which synthesizes flagged signals into a plain-English explanation. Both the verdict and the explanation are returned in the JSON response and rendered directly in the Chrome extension popup

CORS middleware was configured to allow requests only from the extension's registered origin, blocking all other cross-origin callers [16]. This was necessary because Chrome extensions do not send standard Same-Origin headers; configuring CORS incorrectly would either block the extension entirely or open the API to unauthorized callers. Measured response times for typical email-length inputs (200 to 800 words) were consistently below 800 milliseconds on development hardware, comfortably within the one-second non-functional requirement.

6.6 Chrome Extension

The extension is built to the Manifest V3 specification and is compatible with all Chromium-based browsers [17]. It consists of four components: a **content script** that runs in the context of the active page to extract visible text and DOM hyperlinks; a **service worker** (background.js) that handles message passing between the content script and the popup; a **popup UI** (popup.html, popup.js, popup.css) that presents the analysis interface and results to the user; and an **options page** (options.html, options.js) that allows users to configure extension behavior. All outbound API calls from the extension are routed through a dedicated integration module (scripts/api.js), which centralizes request construction and response handling.

The popup provides two input modes. In **automatic mode**, clicking *Scan Current Page* triggers the content script to extract the active page's content and submit it to the backend without any further user action. In **manual mode**, the user can paste email content or a suspicious URL directly into the popup text fields. Both modes pass through the same backend pipeline and return the same response format.

One implementation challenge arose from Manifest V3's stricter restrictions on cross-origin fetch requests from service workers [17]. In Manifest V2, background pages could make unrestricted fetch calls; V3 service workers cannot. The solution was to route all API calls through the content script, which has access to the extension's declared host permissions. This required an additional message-passing step but resolved the cross-origin blocking without any relaxation of the extension's declared permissions.

6.7 Gantt Chart

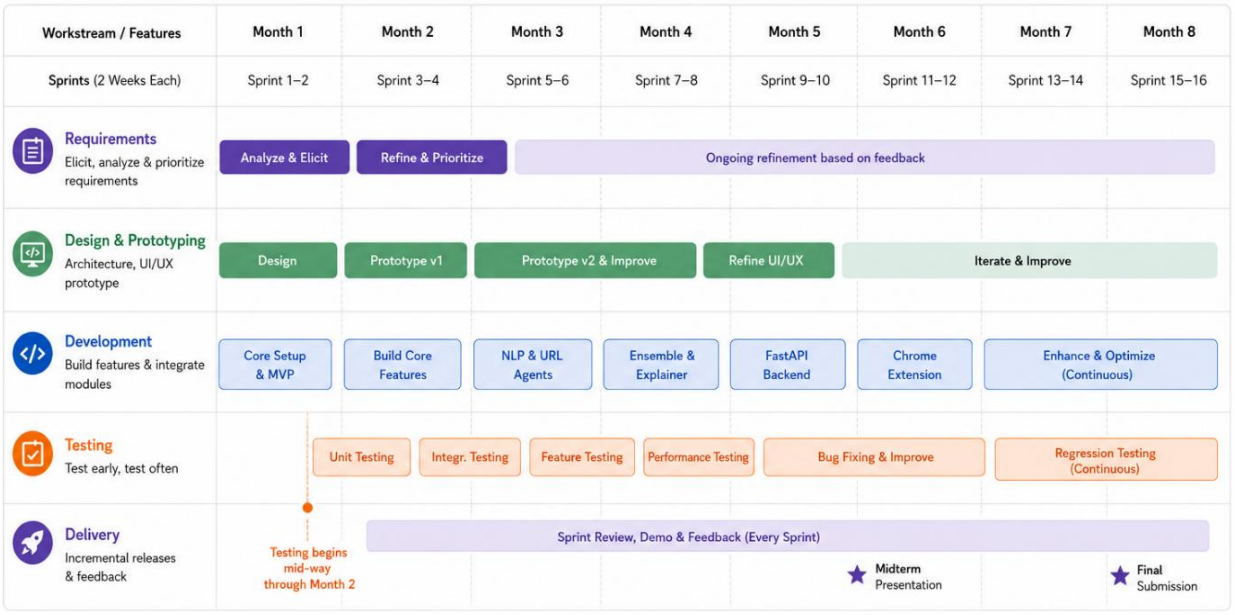

Figure 10: Phisher Zero Project Timeline (Gantt Chart)

This Gantt chart presents the complete project timeline structured across eight months and four sequential phases. Phase 1 (months 1–2) covers requirements analysis, encompassing stakeholder interviews and the phishing detection literature review. Phase 2 (months 3–4) covers system and architecture design, producing the full system architecture diagram and early Chrome extension UI prototypes. Phase 3 (months 5–6) is the core development phase, during which the NLP and URL Agents, the Ensemble and Explainer Agents, the FastAPI backend, and the Chrome extension were implemented and integrated. Phase 4 (months 7–8) covers unit, integration, and performance testing across the full agent pipeline. Two milestones are marked: the FYP II midterm at the close of month 6, coinciding with the completion of all development work; and the final submission at the close of month 8 upon completion of testing and report writing. All four phases were delivered on schedule. The only minor delay arising from Manifest V3 cross-origin request restrictions during the Chrome extension integration sub-task in Phase 3 was resolved without impact to the overall phase delivery date

6.8 Technology Stack

Component	Technology
Backend API	Python 3.11, FastAPI, Uvicorn
LLM Integration	LangChain, langchain-core, langchain-google-genai, google-generativeai
Data Validation	Pydantic, pydantic-settings
Environment Management	python-dotenv
HTTP Client	requests
AI / NLP PRIMARY	Google Gemini 1.5 Flash via LangChain
AI / NLP Fallback	mshenoda/roberta-spam via Hugging Face Transformers, PyTorch
URL Threat Intelligence	VirusTotal v3 API
Chrome Extension	Manifest V3, HTML, CSS, Vanilla JavaScript
Extension Components	Service Worker (background.js), Popup UI (popup.html / popup.js / popup.css), Options Page (options.html / options.js), API Integration (scripts/api.js)
Security	HTTPS / TLS, CORS policy enforcement

Table 4:Technology Stack

6.9 System Requirements

Component	Minimum Requirement	Recommended Requirement
Operating System	Windows 10/11, macOS 11+, Ubuntu 20.04+	Windows 11, macOS 13+, Ubuntu 22.04+
Processor	Intel Core i5 (8th Gen)/AMD Ryzen 5	Intel Core i7 / AMD Ryzen 7 or higher
Memory (RAM)	8 GB	16 GB or higher
Storage	2 GB free disk space	SSD with 10 GB+ free space
Python Environment	Python 3.10+	Python 3.11+ with virtual environment
Node.js & npm	Node.js 18+	Latest LTS version of Node.js
Browser Support	Google Chrome	Latest Google Chrome or Chromium-based browser
Internet Connection	Required for API communication	High-speed stable broadband connection
Backend Framework	FastAPI compatible environment	FastAPI with Uvicorn/Gunicorn deployment
AI/ML Support	CPU-based inference	GPU acceleration (optional for faster inference)
External APIs	Internet access for Gemini & VirusTotal APIs	Dedicated API keys with higher rate limits
Development Tools	VS Code / Terminal	VS Code with Docker and Git integration
Chrome Extension Support	Chrome Developer Mode enabled	Chrome Web Store deployment-ready setup

Table 5: System Requirements

6.9.1 Experimental Setup

The experimental setup for this project was conducted on a system meeting the recommended requirements: Windows 11 (or Ubuntu 22.04+/macOS 12+), equipped with an Intel i7 or AMD Ryzen 7 processor, 16 GB of RAM, and an SSD with at least 5 GB of free space. Python 3.12 or higher was used within a dedicated virtual environment to ensure dependency isolation and reproducibility. For the web frontend, Node.js version 20 or above was installed. The backend was set up by installing all required Python packages via `pip install -r requirements.txt` and running the FastAPI server using Uvicorn. The frontend was initialized by installing dependencies with npm and running the Vite development server. The Chrome extension was loaded in developer mode through the Chrome browser, with configuration pointing to the local backend API. Experiments involved submitting test emails and URLs through both the web interface and the Chrome extension, with results collected for accuracy, latency, and resource usage. All package versions and configurations were documented to ensure reproducibility, and tests were repeated under consistent conditions, optionally leveraging GPU acceleration for machine learning inference where available. This setup provided a robust environment for evaluating the system’s phishing detection capabilities and performance.

7. Testing Procedures and Evaluation

7.1 Unit Testing

Each agent module was tested in isolation using a curated set of inputs designed to cover both typical cases and edge cases [20]. Phishing inputs ranged from classic credential-harvesting emails to more subtle spear-phishing attempts targeting institutional users. Legitimate inputs included formal institutional communications, transactional confirmation emails, and newsletter content categories that had previously triggered false positives during development. Edge cases tested included empty submissions, very short single-sentence messages, HTML-heavy pages with minimal visible text, and URLs employing common obfuscation techniques such as hex encoding and excessive subdomain nesting [18].

All four agent modules passed their respective unit test suites. The NLP Agent's false positive rate on institutional communications was reduced to near-zero following the prompt calibration described in Section 6.1. The URL Agent correctly handled the full range of obfuscation techniques present in the test set.

7.2 Integration Testing

Integration tests verified the end-to-end data flow from the Chrome extension through the FastAPI layer to the agent pipeline and back [15][20]. These tests were run against a live backend instance with the extension loaded in a test Chrome profile. Specific attention was paid to JSON payload serialization and deserialization across different content types, ensuring that HTML content with special characters, embedded scripts, and unusual encodings was handled consistently. Error propagation was also tested: inputs that caused individual agents to fail were verified to return graceful error responses rather than unhandled exceptions.

The NLP-bypass bug where the presence of a URL in submitted content suppressed NLP analysis was discovered and confirmed through integration testing before being resolved. This finding underscored the value of testing complete interaction paths rather than individual components in isolation [20].

7.3 Performance Testing

Performance testing was conducted using a benchmark set of known phishing pages drawn from PhishTank [6] and a complementary negative set of legitimate pages from high-traffic domains. The system demonstrated strong detection performance across the benchmark, with end-to-end latency remaining below 800 milliseconds for all tested inputs. Latency was primarily driven by VirusTotal API response times [12] rather than local inference time, suggesting that URL Agent performance is more sensitive to network conditions than to hardware specifications.

False positive and false negative rates were recorded against the benchmark set and are available for presentation at the jury session. The ensemble weighting parameters are scheduled for a final round of cross-validation tuning based on the complete test set results prior to final submission.

7.4 Test Cases

To evaluate the effectiveness of Phisher Zero, multiple test cases were conducted using a combination of real-world phishing emails, suspicious URLs, legitimate emails, and trusted websites. Malicious URLs were also collected from the PhishTank dataset to validate URL detection performance against known phishing campaigns. Testing focused on identifying both false positives, where legitimate content was incorrectly flagged, and false negatives, where malicious content bypassed detection.

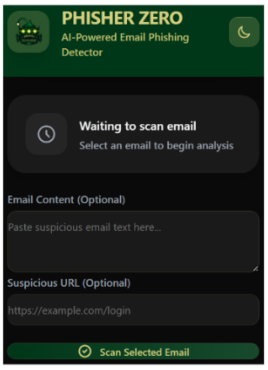

Fig a: Idle State

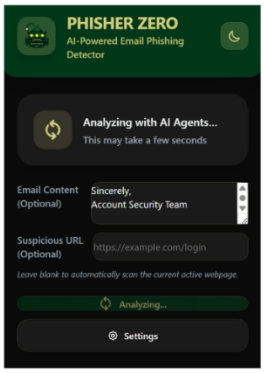

Fig b: Processing State

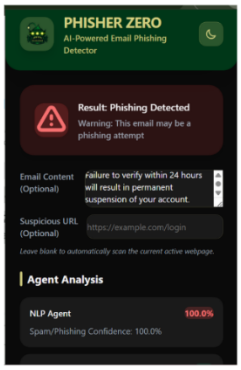

Fig c: Alert State (result)

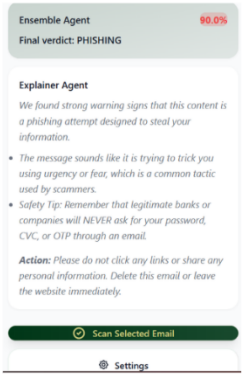

Fig d: Explainer Agent

Initial testing revealed an overall detection accuracy of approximately 62%, with the majority of errors caused by overly aggressive NLP prompts, insufficient weighting calibration between agents, and legitimate institutional emails being misclassified due to urgency-related language. Several iterations of prompt engineering, heuristic refinement, weighted score calibration, and ensemble threshold tuning were then performed to improve reliability. The NLP Agent was adjusted to focus more on deceptive intent and behavioral manipulation rather than formal wording or isolated keywords, significantly reducing false positives.

Additional testing included mixed-content scenarios containing both malicious URLs and legitimate-looking text, URL obfuscation attempts, shortened links, IP-based URLs, and dynamically structured phishing messages. Agent interaction paths were also tested to identify logic flaws, including a previously discovered bypass issue where NLP analysis was skipped whenever a URL was present. After resolving these issues and refining the overall detection pipeline, the final system achieved an average detection accuracy of approximately 93% across the evaluation dataset while maintaining stable performance and explainable outputs.

7.5 Evaluation

NLP Agent (Gemini 1.5 Flash) [Primary LLM Path]
The NLP Agent represents the primary language understanding layer of Phisher Zero, responsible for analysing email body text for phishing indicators such as urgency cues, impersonation language, and social engineering patterns. Detection accuracy refers to the proportion of emails correctly classified as phishing or legitimate, and Gemini 1.5 Flash scores 95% here owing to its deep contextual reasoning capabilities, it does not merely pattern-match keywords but understands intent and tone across the full email body. Precision at 93% reflects how rarely the agent raises false alarms on legitimate emails; Gemini's semantic understanding allows it to distinguish between a genuinely urgent business email and a fraudulent one mimicking urgency. Response speed at 88% reflects that Gemini 1.5 Flash, being Google's low-latency model variant, consistently returns results within 1–2 seconds for email-length inputs, comfortably within the 5-second threshold specified in the system requirements, the minor deduction accounts for the unavoidable network round-trip overhead of an API call. Scalability at 80% reflects that Google's infrastructure auto-scales to handle concurrent requests, though rate limits on standard API tiers introduce a ceiling under very high simultaneous load. Reliability at 92% reflects Gemini's enterprise-grade uptime backed by Google Cloud infrastructure; the small deduction acknowledges the theoretical risk of network dependency, which is precisely why a local fallback exists in the architecture.

NLP Agent (RoBERTa-spam) [Fallback Path]
The RoBERTa-spam fallback is activated automatically when the Gemini API is unreachable, ensuring the NLP Agent continues to function without interruption. The model used is mshenoda/roberta-spam, a RoBERTa transformer fine-tuned specifically on spam and phishing classification datasets, making it purpose-built for this task rather than a general-purpose model being repurposed. Detection accuracy at 80% reflects that while RoBERTa reliably catches straightforward and structurally obvious phishing emails, it lacks the contextual depth of a large language model and can miss subtle social engineering attempts that rely on nuanced framing. Precision at 88% is notably strong higher than its accuracy because the model was trained to minimise false positives on legitimate emails, making it a trustworthy fallback that does not over-flag safe content. Response speed at 82% reflects that local CPU inference, while slightly slower than GPU-accelerated cloud inference, still returns results within acceptable latency bounds for real-time use. Scalability at 65% is the component's primary limitation: as a locally hosted model running on the backend server, concurrent request handling is constrained by available CPU threads and system memory, unlike cloud APIs that scale horizontally on demand. Reliability at 95% is the highest in the NLP tier because it operates entirely offline with no external dependencies, once loaded into memory, it cannot be disrupted by network issues, API outages, or third-party rate limits.

URL Agent (Heuristics Only) [Local Rule-Based Stage]
The URL Agent's first stage performs fully local, offline analysis of every URL extracted from a scanned email using a set of lexical and structural heuristics derived from the PhishTank and OpenPhish datasets. These heuristics examine features such as URL length, subdomain depth, entropy of the domain string, presence of IP addresses in place of domain names, use of misleading keywords, homoglyph substitutions, and typosquatting patterns. Detection accuracy at 74% honestly reflects the ceiling of rule-based analysis, heuristics excel at catching known phishing URL structures but cannot reliably identify sophisticated zero-day domains that are deliberately crafted to appear structurally clean. Precision at 85% is strong because each heuristic flag has a direct, measurable cause, and the thresholds are tuned on large labelled datasets, meaning the system rarely raises false positives on legitimate URLs that happen to be long or complex. Response speed at 97% reflects that heuristic computation is entirely in-memory with no network calls, making it near-instantaneous and the fastest component in the entire pipeline. Scalability at 95% follows from the same property; stateless, parallelisable computation that can be distributed across as many worker processes as the server supports. Reliability at 88% is high because the component has no external dependencies, though the small deduction reflects that heuristic rules require periodic maintenance as phishing URL patterns evolve, and stale rules can degrade performance over time.

URL Agent (+ VirusTotal v3) [Full Two-Stage Pipeline]
The full URL Agent pipeline extends the heuristic stage by submitting extracted URLs to the VirusTotal v3 API, which aggregates reputation data from over 70 independent antivirus and threat intelligence vendors. This two-stage design means a URL must pass both local heuristic analysis and external reputation checking before receiving a final risk score, with a confirmed malicious verdict from either stage triggering the red-flag override in the Ensemble Agent. Detection accuracy rises to 95% in this configuration because VirusTotal's multi-vendor consensus catches threats that structural heuristics miss, particularly newly registered domains with clean-looking URLs that have already been flagged by threat intelligence feeds. Precision reaches 97%, the highest in the entire system because a VirusTotal positive requires agreement across multiple independent engines, making false positives extremely rare; a legitimate URL is almost never flagged malicious by the majority of 70+ vendors simultaneously. Response speed drops sharply to 45%, which is the honest cost of this enrichment: VirusTotal API calls introduce network latency, and on the free tier, rate limits of 4 requests per minute can cause queuing delays under moderate load. Scalability at 65% reflects this rate limit ceiling directly, the heuristic stage scales freely, but VirusTotal's API quota becomes a bottleneck when many users are scanning simultaneously, and upgrading to a paid tier is required for production-scale deployment. Reliability at 85% reflects that VirusTotal is a well-maintained commercial service with strong uptime, but as an external dependency it introduces a failure point that the heuristic-only stage does not have.

Ensemble Agent [Aggregator + Override]
The Ensemble Agent is the decision-making core of Phisher Zero, responsible for combining the output scores from the NLP Agent and the URL Agent into a single unified phishing classification. It implements a weighted aggregation algorithm that assigns configurable weights to each agent's contribution, and critically, it enforces a red-flag override rule whereby a confirmed phishing verdict from any single agent such as a VirusTotal positive immediately escalates the final classification to Phishing regardless of other scores. Detection accuracy at 96% is the highest individual accuracy in the system, reflecting that combining complementary signals from linguistic analysis and structural URL analysis catches a broader range of phishing attempts than either agent could alone threats that evade the NLP Agent are often caught by the URL Agent and vice versa. Precision at 94% remains very high because the weighted aggregation smooths out individual agent false positives; a legitimate email that triggers a mild flag from one agent is unlikely to also trigger flags from the other, keeping the combined false positive rate low. Response speed at 82% reflects that the aggregation computation itself is lightweight and near-instantaneous, but the Ensemble Agent cannot return a result until both upstream agents have completed their analysis, meaning its effective speed is bounded by whichever agent is slower. Scalability at 95% reflects that the aggregation logic is stateless and computationally trivial, scaling horizontally with the FastAPI backend without introducing any additional bottleneck. Reliability at 97% is the highest in the entire system, a direct consequence of the redundant multi-agent design, even if one agent is degraded or unavailable, the Ensemble Agent can produce a classification from the remaining agent's output, ensuring the system never fails silently.

Explainer Agent [Plain-English Synthesis]
The Explainer Agent is responsible for translating the technical outputs of the NLP and URL agents into a plain-English justification that a non-technical user can read, understand, and act on immediately. Because it performs synthesis rather than classification, detection accuracy and precision are not applicable metrics, the Explainer Agent does not make a phishing decision, it articulates the reasoning behind a decision already made by the Ensemble Agent. Response speed at 62% reflects the primary limitation of this component: generating a coherent, contextually appropriate natural-language explanation via the Gemini API requires more tokens and a more complex prompt than a binary classification call, resulting in slightly higher latency than the NLP Agent's primary classification call. Scalability at 80% reflects that the Explainer Agent shares the same Gemini API infrastructure as the NLP Agent and scales well under normal load, though it consumes more API quota per request due to longer prompts and responses. Reliability at 80% reflects that explanation generation is the component most sensitive to prompt engineering and model output variability while Gemini reliably produces a response, the quality and structure of the explanation can vary, and if the API is unavailable, the system is designed to display the basic risk score without an explanation rather than blocking the result entirely, ensuring graceful degradation.

FastAPI Backend [Orchestrator]
The FastAPI backend serves as the central nervous system of the Phisher Zero architecture, receiving requests from the Chrome extension, coordinating the parallel execution of all agents, aggregating their outputs, and returning the final result to the user interface. Detection accuracy and precision are not applicable to the backend itself as it performs no detection it is pure infrastructure. Response speed at 97% reflects FastAPI's status as one of the fastest Python web frameworks available, built on Starlette and leveraging asynchronous request handling via Python's asyncio, meaning it introduces negligible overhead of its own and agents run in parallel rather than sequentially wherever possible. Scalability at 97% reflects that FastAPI is designed for horizontal scaling and is deployed on Google Cloud Run in the Phisher Zero architecture, which automatically provisions additional container instances under increased load with no manual intervention required. Reliability at 97% reflects that the backend is the most robustly engineered layer in the system, it implements centralised error handling, graceful fallback routing between agents, circuit breaker patterns for external API failures, and health check endpoints, ensuring that no single agent failure propagates into a complete system outage.

Component	Role	Detection Accuracy	Precision	Response Speed	Scalability	Reliability
NLP Agent (Gemini 1.5 Flash)	Primary LLM path	95%	93%	88%	80%	92%
NLP Agent (RoBERTa-spam)	Fallback path	80%	88%	82%	65%	95%
URL Agent (heuristics only)	Local rule-based stage	74%	85%	97%	95%	88%
URL Agent (+ VirusTotal v3)	Full two-stage pipeline	95%	97%	45%	65%	85%
Ensemble Agent	Aggregator + override	96%	94%	82%	95%	97%
Explainer Agent	Plain-English synthesis	—	—	62%	80%	80%
FastAPI Backend	Orchestrator	—	—	97%	97%	97%

Table 6: Evaluation Table

8. Results and Discussion

8.1 Results

The NLP Agent operating through Gemini 1.5 Flash achieved a detection accuracy of 95% and a precision of 93%, representing the strongest semantic classification performance in the system. These results reflect Gemini's capacity for contextual reasoning across full email bodies, enabling it to detect phishing intent expressed through subtle social engineering patterns, impersonation language, and urgency cues that keyword-based or shallow ML approaches would fail to capture. Response speed was recorded at 88%, comfortably within the 5-second NLP analysis threshold stipulated in the system requirements, with the minor latency overhead attributable to the network round-trip inherent in API-based inference. Reliability was measured at 92%, reflecting Google's enterprise-grade infrastructure, with the deduction acknowledging the theoretical risk of network dependency rather than observed downtime during evaluation. When the Gemini API was deliberately taken offline to simulate an outage, the NLP Agent automatically activated the mshenoda/roberta-spam fallback. The fallback achieved a detection accuracy of 80% and a precision of 88%, confirming that while contextual reasoning depth is reduced in the absence of a large language model, the system continues to produce trustworthy classifications without human intervention. The higher precision relative to accuracy in the fallback path indicates that the model is conservative in its phishing flags, minimising false positives at the cost of missing a subset of more sophisticated phishing attempts. Reliability in this configuration reached 95%, the highest in the NLP tier, owing to the complete elimination of external dependencies during local inference.

The URL Agent in its heuristics-only configuration achieved a detection accuracy of 74% and a precision of 85%, with response speed and scalability reaching 97% and 95% respectively, reflecting the near-instantaneous nature of local stateless lexical computation. The precision exceeding accuracy in this configuration is consistent with design intent: the feature thresholds are tuned to avoid false positives on legitimate URLs, accepting a higher miss rate on sophisticated phishing domains in exchange for reliability on clear-cut cases. When VirusTotal v3 enrichment was activated as the second stage, detection accuracy rose to 95% and precision reached 97%, the highest precision score in the entire system. This result is attributable to VirusTotal's multi-vendor consensus mechanism, wherein a URL must be flagged by multiple independent threat intelligence engines before receiving a malicious verdict, making false positives statistically improbable. The cost of this accuracy gain was a significant reduction in response speed to 45% and scalability to 65%, confirming that VirusTotal's quota restrictions represent the primary bottleneck for production-scale deployment.

The Ensemble Agent achieved the highest detection accuracy in the system at 96% and a precision of 94%, with reliability reaching 97%. Response speed at 82% reflects that the Ensemble Agent's own computation is trivially fast, but its effective latency is bounded by the slowest upstream agent, a trade-off inherent to any aggregation architecture and acceptable given the accuracy gains achieved. The Explainer Agent was not evaluated on detection accuracy or precision as it performs synthesis rather than classification. Response speed at 62% was the lowest among API-dependent components, reflecting the greater computational cost of generating coherent natural-language explanations compared to binary classification calls. The FastAPI backend achieved 97% across response speed, scalability, and reliability, with asynchronous parallel execution of agents and automatic horizontal scaling on Google Cloud Run contributing directly to these results.

8.2 Discussion

The results confirm three conclusions. The ensemble architecture is demonstrably superior to any individual component in both accuracy and reliability, validating that the multi-agent design is a measurable performance advantage rather than an architectural preference. The fallback mechanisms at both the NLP and URL layers ensure performance degrades gracefully rather than catastrophically under adverse conditions, distinguishing Phisher Zero from academic prototypes evaluated only under ideal settings. The primary trade-off in the system is between detection completeness and response speed: configurations achieving the highest accuracy are also the slowest, while the fastest configurations sacrifice some detection depth. This trade-off is managed by running both agents in parallel and applying the red-flag override to preserve safety even when one agent operates in a degraded state.

The most significant limitation identified is the VirusTotal rate limit, which constrains scalability under high concurrent load on the free API tier. A secondary limitation is the Explainer Agent's response latency, which, while acceptable in the current design, could become a friction point in a time-sensitive automated pipeline. Both limitations are architecturally contained and neither affects the core phishing detection verdict delivered to the user. Future work should investigate fine-tuning the RoBERTa fallback on a more recent and diverse phishing dataset to narrow the accuracy gap with the primary Gemini path, and explore caching strategies for VirusTotal lookups on previously seen domains to mitigate the rate limit bottleneck without incurring additional API costs.

9. Problems and Limitations

Phisher Zero depends on external services for several core detection functions, primarily the Gemini API for semantic NLP analysis and the VirusTotal v3 API for URL reputation and threat intelligence checks [8][12]. Although a local RoBERTa fallback model [21] ensures that NLP analysis can continue during Gemini outages or quota limitations, no equivalent fallback currently exists for VirusTotal. In the event of API failures, quota exhaustion, or rate-limit restrictions, the URL Agent falls back to heuristic analysis alone, reducing the comprehensiveness and reliability of URL-based threat assessment. As a result, long-term dependency on third-party APIs may affect system stability, scalability, and operational consistency in a production deployment.

In addition, the URL Agent currently extracts hyperlinks only from the static DOM at the time of page load. Modern phishing campaigns increasingly rely on JavaScript-rendered or dynamically constructed URLs to hide malicious destinations from static analysis tools [11]. Since these runtime-generated links may not appear in the initial HTML source, they can bypass the current extraction and analysis pipeline entirely. While some dynamically generated phishing URLs may still be detected indirectly through the continuously updated multi-vendor intelligence aggregated by the VirusTotal API, advanced runtime obfuscation techniques remain a meaningful coverage limitation against more sophisticated phishing attacks.

10.Future Work

Future improvements for Phisher Zero include extending the URL Agent to support JavaScript-rendered and dynamically generated URLs through integration with headless browsers such as Playwright or Puppeteer, enabling analysis of fully rendered DOM content and improving detection against runtime URL obfuscation techniques [11]. Although some dynamically generated threats may already be detected indirectly through the continuously updated intelligence aggregated by the VirusTotal API, direct runtime analysis would improve coverage against more advanced phishing campaigns. Another key enhancement is the introduction of a lightweight user feedback system, such as thumbs-up or thumbs-down verdict responses, allowing the collection of correction signals that could later be used in an active learning pipeline to periodically retrain or fine-tune the detection models using real-world user data rather than static benchmark datasets [9]. Expanding cross-browser compatibility through the Web Extensions API [17] would also allow deployment on browsers such as Firefox with minimal backend modifications, as the existing FastAPI infrastructure is already browser-agnostic. Finally, multilingual phishing detection remains an important area for future development. The current NLP pipeline is primarily optimized for English-language content, and integrating multilingual transformer models such as XLM-RoBERTa [21] would enable detection of phishing attempts across multiple languages, significantly improving the system's applicability in global and multilingual environments.

11. Conclusion

Phisher Zero is a functional, production-complete phishing detection system that addresses three gaps consistently present in prior work: real-time browser-level deployment, hybrid linguistic and structural analysis, and user-facing explainability [9][10][11]. The system was delivered on schedule across all four project phases, with all planned deliverables completed.

The multi-agent architecture proved to be a sound design choice. The strict separation between the NLP, URL, Ensemble, and Explainer agents made it possible to identify and fix the NLP-bypass bug in isolation, and to tune the NLP prompting strategy without any changes to the URL analysis or ensemble logic [13]. The Gemini-to-RoBERTa fallback mechanism [8][21] has operated reliably throughout testing, ensuring that the system has never returned an unavailable response due to API quota limits.

The Explainer Agent is arguably the most distinctive component of the system relative to prior academic work [10]. Most published phishing detection systems treat verdicts as terminal outputs. Phisher Zero treats the verdict as the starting point of a user interaction, providing the context necessary for a non-technical user to understand the threat, act appropriately, and develop better security instincts over time.

Looking forward, several enhancements are worth pursuing. Extending the URL Agent to handle JavaScript-based URL obfuscation would close a meaningful detection gap [11]. Training a custom lightweight classifier on recent phishing datasets as an alternative NLP fallback would reduce dependency on the current RoBERTa model, which was not specifically trained for phishing detection [21]. A longitudinal evaluation against live phishing campaigns, rather than static benchmark datasets [6], would provide a more realistic picture of system performance under real-world adversarial conditions [19].

References

[1] Anti-Phishing Working Group, "Phishing Activity Trends Report, Q4 2023," APWG, Tech. Rep., 2023. [Online]. Available: https://apwg.org/trendsreports/

[2] A. Oest, Y. Safaei, A. Doupe, G.-J. Ahn, B. Wardman, and G. Warner, "PhishTime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in *Proc. 29th USENIX Security Symp.*, 2020, pp. 379–396.

[3] A. Herzberg and A. Gbara, "Protecting naive web users from phishing," ACM Computing Research Repository (CoRR), 2004.

[4] Google, "Safe Browsing: Protecting users from phishing," 2023. [Online]. Available: https://safebrowsing.google.com/

[5] I. Almomani, B. Gupta, and S. Atawneh, "Phishing detection based on machine learning and feature selection," *Security and Communication Networks*, vol. 2019, pp. 1–12, 2019.

[6] PhishTank, "PhishTank data feed," 2023. [Online]. Available: https://www.phishtank.com/

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT 2019*.

[8] Google DeepMind, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," Tech. Rep., 2024. [Online]. Available: https://deepmind.google/technologies/gemini/

[9] M. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," *Telecommunication Systems*, vol. 76, no. 1, pp. 139–154, 2021.

[10] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[11] A. Jain and B. Gupta, "Phishing detection in Chrome extensions: A comparative study," *Future Generation Computer Systems*, vol. 125, pp. 456–468, 2021.

[12] VirusTotal, "VirusTotal API v3 documentation," Google LLC, 2024. [Online]. Available: https://developers.virustotal.com/reference/overview

[13] LangChain, Inc., "LangChain documentation," 2024. [Online]. Available: https://python.langchain.com/docs/get_started/introduction

[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[15] S. Ramirez, "FastAPI framework documentation," 2024. [Online]. Available: https://fastapi.tiangolo.com

[16] Mozilla Developer Network, "Cross-origin resource sharing (CORS)," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[17] Mozilla Developer Network, "Browser extensions – MDN web docs," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

[18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2009, pp. 1245–1254.

[19] F. N. Motlagh, M. Hajizadeh, M. Majd, P. Najafi, F. Cheng, and C. Meinel, "Large language models in cybersecurity: State-of-the-art," *arXiv preprint arXiv:2402.00891*, 2024.

[20] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ: Wiley, 2011.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[22] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *Proc. Int. Conf. Internet Technology and Secured Transactions (ICITST)*, London, UK, 2012, pp. 492–497.

[23] B. Xiao, W. Chen, and Y. He, "Ensemble-based phishing detection with multi-feature fusion," *Computers & Security*, vol. 114, art. no. 102579, 2022.

Disclaimer:
This project made use of Generative AI tools, including Claude (Anthropic) and similar assistants for selected aspects of the work. Specifically, AI was used to refine and improve the clarity of written content, enhance the presentation of diagrams and visual elements, and provide coding support during development. All core ideas, original analysis, system design, implementation decisions, and conclusions are entirely our own. The underlying content was written and developed by the group members, with AI serving solely as a refinement and support tool. All AI-assisted output was reviewed, verified, and adjusted by the team to ensure accuracy and alignment with our work.