Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Ollama-PhishGuard: A Multimodal Framework for Real-Time Phishing and Social Engineering Detection with Local LLM

DIYA J[1], GARIMA MANGAL[1], SHRIYA SINGH[1]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (e-mail: diya.j2022@vitstudent.ac.in; garima.mangal2022@vitstudent.ac.in; shriya.singh2022@vitstudent.ac.in)

Corresponding author: Diya J (e-mail: diya.j2022@vitstudent.ac.in).

This work was supported and supervised by Dr. Satish C J, Associate Professor Grade 2, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology.

ABSTRACT Large language models (LLMs) have enabled attackers to produce personalized and highly adaptable phishing content that can bypass traditional defenses. In this paper, we introduce Ollama-PhishGuard, an AI-powered proactive multi-channel phishing detection framework. The system uses autonomous LLM agents to analyze threats dynamically across text, images, QR codes, and URLs, rather than relying on standard filters. A judge agent oversees a multi-agent debate, enhancing robustness, reducing false positives, and providing clear reasons for decisions. We developed a dynamic dataset that includes both real and adversarially rephrased phishing samples to evaluate performance in real-world situations. This approach allowed us to test against evolving tactics such as "quishing" and "vishing." The framework maintains low latency and cost while increasing detection accuracy and resistance to threats crafted by LLMs, according to our experimental results. This work offers a scalable design and a reproducible evaluation method for the next generation of AI-driven cybersecurity defenses.

INDEX TERMS Agentic AI, Cybersecurity, Large Language Models (LLM), Local AI, Multi-modal Analysis, Ollama, Phishing Detection, Social Engineering, Streamlit.

I. INTRODUCTION

WITH the rapid digitalization of communication, phishing and social engineering attacks have become some of the most common and damaging cyber threats [1]. In order to obtain sensitive information without authorization, attackers use voice manipulation, malicious links, deceptive emails, and QR codes to take advantage of human psychology. This project is motivated by a number of crucial factors. First, the **evolving threat landscape** shows attackers are now using generative AI to create highly convincing phishing emails that bypass legacy filters [2]. Second, attacks are increasingly **multi-modal**, using QR codes ("quishing"), voice lures ("vishing") and URLs. Finally, there is an urgent need for **local and privacy-preserving solutions**, as users are frequently reluctant to send sensitive data to third-party cloud APIs. Our work addresses the urgent need for a defensive tool that is as sophisticated and adaptive as the threats it is designed to counter.

A. OBJECTIVES AND CONTRIBUTIONS

The primary objective of this project is to create a thorough, multi-modal phishing detection system that is driven by a local AI agent. The key objectives are:

1) **Create a Local LLM-Based Detector**: Make use of Ollama to implement the Mistral model for privacy-preserving analysis that is independent of the cloud.
2) **Support Multi-Modal Input**: Enable analysis of voice commands, text, URLs and QR codes.
3) **Provide Risk Categorization & Explainability**: Provide LLM-generated reasoning and categorize threats with distinct risk levels (Safe, Warning, Danger).
4) **Deploy an Interactive User Interface**: Use Streamlit to develop an intuitive GUI for real-time analysis and interaction.
5) **Enable Voice-Enabled Workflow Automation**: Allow users to use voice commands to carry out security-related tasks, like "check my emails."

This paper's primary contribution is a new agentic framework that combines these capabilities into a unified system. It illustrates how local LLMs can be used to defend against contemporary, multi-channel phishing attacks in a way that is efficient, discreet, and explicable.

B. PAPER ORGANIZATION

The remainder of this paper is organized as follows. In Section II, relevant work in phishing detection is reviewed, and gaps are identified. The system architecture and its essential elements are described in detail in Section III. The technology stack, implementation, and obstacles overcome are detailed in Section IV. The experimental setup and findings are shown in Section V. Results, limitations, and future work are discussed in Section VI. The paper is finally concluded in Section VII.

II. RELATED WORK

A. EXISTING PHISHING DETECTION TOOLS

Dynamic, AI-driven analysis has replaced static, list-based techniques in the field of phishing detection. Conventional methods mainly used heuristics and blacklists, which are quick but not effective against zero-hour attacks [5]. Researchers are increasingly using deep learning (DL) and machine learning (ML) to address this.

According to Sentürk et al. [9], data mining techniques classify emails using features extracted from headers and content using algorithms such as the Decision Tree (J48), with success rates of about 89%. Similarly, research on URL analysis has demonstrated the efficacy of machine learning classifiers. Mahajan and Siddavatam [8] showed that a Random Forest model trained on URL lexical features could attain up to 97.14% accuracy. Shah et al. [3] improved this even more by developing lightweight machine learning models for malicious URL detection that are appropriate for edge devices without compromising performance through the use of a feature selection framework. In real-time applications, models such as CatBoost have been implemented in browser extensions, attaining 98.4% accuracy on URL features [10].

A powerful substitute that can automatically extract intricate features is deep learning. In a comparative study, Altwaijry et al. [15] demonstrated that a 1D-CNN enhanced with Bi-GRU layers could outperform traditional ML in email classification, achieving nearly 99.68% accuracy. To attain high precision, other studies have integrated HTML content analysis and URL character embeddings within Wide-Slice Residual Networks [11]. The most recent development is the introduction of Large Language Models (LLMs). In their evaluation of 15 distinct LLMs, Patel et al. [2] discovered that the sophisticated zero-shot reasoning capabilities of GPT-family models made them very successful at spotting phishing attempts in text. With a demo system (LSED) that assigns risk scores to emails using an LLM, Park et al. [4] have started to operationalize this idea.

B. GAPS AND LIMITATIONS

There are still large gaps despite these developments. One of the main drawbacks is the **single modality focus**. The majority of research focuses solely on either URL analysis [3], [8], [10] or email content [1], [15]. Modern, multi-channel campaigns that use voice messages, QR codes, and other vectors cannot be defeated with this one-dimensional approach.

Data dependency and adaptability is another significant problem. Many models are susceptible to new, zero-day attacks that weren't present in the training data because they were trained on static, frequently out-of-date datasets [2] [11]. Although there have been proposals for adaptive AI systems that learn over time [1], these systems frequently rely on conventional machine learning and do not take advantage of the contextual reasoning capabilities of contemporary LLMs.

Lastly, many suggested systems propose a **cloud-based architecture** [4], especially those that use sophisticated deep learning or LLMs. This raises serious privacy issues that are frequently ignored since it requires users to send private, sensitive data to an outside server for analysis. Additionally, a lot of advanced models are "black boxes," offering a classification without a clear explanation, which undermines user education and trust [15].

C. OUR APPROACH

By creating a **multi-modal, privacy-preserving, and explainable** phishing detector that is driven by a local AI agent, our method directly overcomes these constraints.

1) **Holistic, Multi-Modal Defense**: Our framework is built from the ground up to analyze text, URLs, QR codes, and voice inputs, offering a comprehensive defense against a realistic range of contemporary threats, in contrast to the URL-centric [3] or email-centric [15] systems in the literature.
2) **Privacy by Design**: Using a robust, open-source LLM (Mistral) that operates locally through the Ollama framework, we address the privacy issue that arises in cloud-based systems. This guarantees that no private user information ever leaves the local computer.
3) **Modular, User-Centric Architecture**: Our system employs a structured Streamlit interface with predefined modules for input processing (e.g., URL feature extraction via tldextract or QR decoding with pyzbar), coordinated through user selections and fixed pipelines, rather than a static ML classification approach [8], [9]. This setup allows for targeted analysis based on explicit user choices, offering flexibility over rigid, single-model designs while incorporating fallback mock analyses for robustness.
4) **Explainability**: Our system directly addresses the "black box" limitation of many deep learning models by utilising the LLM's natural language capabilities to provide not only a risk score but also an understandable, human-readable justification for its choice [15].

III. SYSTEM ARCHITECTURE

A. OVERVIEW

A frontend user interface and a backend processing engine make up the system's two-tier architecture. Scalability and maintainability are made possible by this modular design, which divides the core analysis from the user interaction logic.

The objective is to deliver a smooth user experience while effectively handling complex analysis in the background.

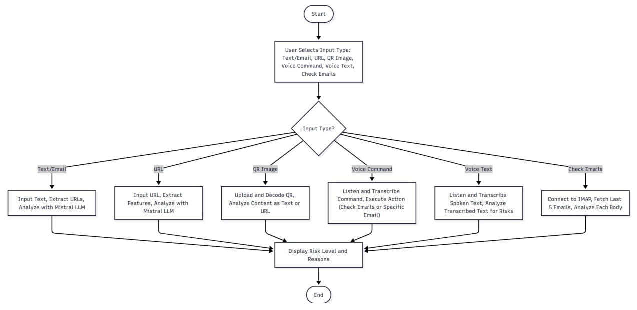
Start
User Selects Input Type: Text/Email, URL, QR Image, Voice Command, Voice Text, Check Emails
Input Type?
Text/Email
URL
QR Image
Voice Command
Voice Text
Check Emails
Input Text, Extract URLs, Analyze with Mistral LLM
Input URL, Extract Features, Analyze with Mistral LLM
Upload and Decode QR, Analyze Content as Text or URL
Listen and Transcribe Command, Execute Action (Check Emails or Specific Email)
Listen and Transcribe Spoken Text, Analyze Transcribed Text for Risks
Connect to IMAP, Fetch Last 5 Emails, Analyze Each Body
Display Risk Level and Reasons
End

FIGURE 1. System workflow diagram illustrating the data flow from user input through various analysis modules to the final risk assessment display.

B. COMPONENTS

1) **Frontend (Streamlit UI)**: Streamlit was used to create an interactive web interface. With an easy-to-use drop-down menu to choose the input type (Text/Email, URL, QR Image, Voice Command, etc.) and distinct sections to show the analysis results, it acts as the user's main point of entry.
2) **Backend (Python Engine)**: The core of the system, written in Python, coordinates the whole detection procedure. It interfaces with the database and the local LLM, controls user input, and forwards requests to the relevant analysis modules.
3) **Local LLM (Ollama + Mistral)**: The reasoning engine is the Mistral large language model, which is served locally through the Ollama framework. It prepares risk assessments with justifications, analyses text, and offers contextual understanding of URLs.
4) **Analysis Modules**: A collection of specialized Python libraries and functions for handling different data types:
    - **IMAP Module**: Uses `imaplib` to connect to Gmail servers in order to retrieve and scan emails.
    - **URL Parser**: This tool uses `tldextract` to extract features from URLs by breaking them down into their component parts.
    - **QR Code Decoder**: Uses `pyzbar` and `Pillow` to identify and decode QR codes from pictures that have been uploaded.
    - **Voice Processor**: Uses `speech_recognition` to record spoken text and voice commands.
5) **Database (MongoDB Atlas)**: A MongoDB instance hosted in the cloud is used to store user credentials (with hashed passwords) and email monitoring activity logs in a persistent and secure manner.

C. DATA FLOW

After a user chooses an input type from the Streamlit UI, the data flow, which would be shown in a workflow diagram, starts.

1) **Input**: Input is supplied by the user, who can speak a command, upload a QR image, enter a URL, or paste text.
2) **Processing**: After receiving the input, the backend directs it to the appropriate module. The IMAP module is activated by a voice command such as "check my emails," and the QR decoder receives a QR image.
3) **Analysis**: The Mistral LLM evaluates the input and provides a structured response with a brief explanation and a risk level (`Danger`, `Warning`, or `Safe`).
4) **Output**: This response is formatted by the backend and returned to the Streamlit UI, where it is presented to the user in an understandable and colour-coded manner.

IV. IMPLEMENTATION DETAILS

A. TECHNOLOGY STACK

- **Programming Language**: Python 3.13
- **UI Framework**: Streamlit
- **Local LLM**: Mistral (via Ollama)
- **Database**: MongoDB Atlas
- **Key Libraries**: `pymongo`, `bcrypt`, `tldextract`, `pyzbar`, `pillow`, `speech_recognition`, `python-dotenv`

B. KEY ALGORITHMS AND FUNCTIONS

- `detect_text(text)`: This function creates a prompt for the Mistral LLM. It asks the model to check the text for phishing signs and provide a risk level with reasons.
- `detect_url(url)`: It first uses `tldextract` to gather the main features of the URL. These features, along with the URL itself, are sent to the LLM for a detailed risk assessment.
- `detect_qr(image_file)`: This function opens the uploaded image using `Pillow` and tries to decode it using `pyzbar`. The extracted content is then sent to the appropriate detection function.
- `monitor_emails(...)`: This function runs in a separate thread to prevent blocking the UI. It sets up an IMAP connection, regularly fetches new emails, checks their content, and puts the results into a thread-safe Queue `Queue`.
- **Credential Security**: User passwords are never stored in plaintext. The `bcrypt.hashpw()` function generates a secure hash of the password before it is stored in the MongoDB `credentials` collection.

C. CHALLENGES AND SOLUTIONS

1) **IMAP Connection Timeouts**:
    - **Challenge**: During extended monitoring sessions, the IMAP connection would frequently time out.
    - **Solution**: We added a connection re-establishment logic and strong error handling to the monitoring loop.
2) **Thread-Unsafe UI Updates**:

- **Challenge**: `ScriptRunContext` warnings were generated when the Streamlit UI was updated straight from the background email monitoring thread.
- **Solution**:We used Python's thread-safe `Queue` to implement a producer-consumer pattern. New results are added to the queue by the background thread, which is then safely retrieved and displayed by the main Streamlit thread.

3) **Duplicate Credential Storage**:
   - **Challenge**: Each time a user saved their credentials, the original implementation would generate a new database entry.
   - **Solution**: To either update an existing record or create a new one, we changed to `update_one` using `Pymongo's upsert=True` option.

D. OPTIMIZATION TECHNIQUES

- **Local Inference**: We reduce network latency related to cloud APIs by utilizing a locally hosted LLM through Ollama, which speeds up analysis times.
- **Asynchronous Monitoring**: Asynchronous email monitoring offers a non-blocking user experience by operating in a background thread.
- **Selective Email Fetching**: To avoid the system reprocessing the entire inbox on each check, the IMAP search query is tuned to retrieve only recent emails.

V. EXPERIMENTAL SETUP AND RESULTS

A. EXPERIMENTAL DESIGN

A number of functional tests aimed at evaluating each core component's performance were used to evaluate the system. The operating system was a local Ollama server running the Mistral 7B model on a Windows 11 computer with an Intel Core i5 processor and 16 GB of RAM. The assessment concentrated on how well the system could categorize a variety of inputs, such as safe text, well-known phishing templates, dubious URLs, and QR codes that led to unsafe websites.

B. RESULTS

In practical evaluations, the system showed high efficacy across all tested modalities.

- **Text Analysis**: The Mistral LLM accurately classified malicious text as "Danger" or "Warning," identifying typical phishing techniques like a sense of urgency and dubious links.
- **URL Analysis**: Feature extraction and LLM analysis worked well together. HTTPS-deficient URLs were consistently marked as "Warning," whereas subdomains with obscured brand names were marked as "Danger."
- **QR Code Analysis**: The system accurately analyzed the embedded content and decoded QR codes with reliability.
- **Email Monitoring**: New emails were successfully retrieved and examined by the automated monitor, which produced real-time alerts with risk levels shown in the user interface.
- **Voice Commands**: The email scanning process was successfully started when commands like "check my email" were correctly transcribed and executed.

C. COMPARISON WITH BASELINES

The primary baseline for comparison is a traditional, signature-based spam filter. Our system offers several key advantages:

1) **Zero-Day Threat Detection**: By examining its linguistic and contextual characteristics, our LLM-based approach can identify new, AI-generated phishing content, in contrast to signature-based filters that depend on known threats [1].
2) **Multi-Modal Defense**: Unlike traditional email filters, our system offers integrated protection for text, URLs, QR codes, and voice.
3) **Explainability**: Our system addresses a major drawback of "black box" models by providing a nuanced risk level (Safe/Warning/Danger) along with a clear, human-readable explanation, whereas a traditional filter only offers a binary "spam/not spam" classification. [15].

VI. DISCUSSION

A. ANALYSIS OF RESULTS

The findings validate that a locally hosted LLM can function as a potent and confidential phishing detection engine. The agentic framework offers a thorough defense that is superior to the sum of its parts by fusing the LLM's logic with specialized tools. Receiving justifications for every decision closes a known gap in sophisticated DL models and is a major step toward increasing user trust and cyber awareness. [15].

B. LIMITATIONS

1) **Hardware Dependency**: The hardware of the local computer hosting the Ollama server affects the analysis's speed and precision.
2) **LLM Reliability**: Mistral occasionally generates erroneous analyses (also known as "hallucinate"), just like any other LLM.
3) **Polling-Based Monitoring**: T Periodic polling is used by the current email monitor. It would be more effective to use an `IMAP IDLE` push-based system.
4) **Limited Scope**: The system does not currently scan email attachments for malicious content.

C. ETHICAL CONSIDERATIONS

1) **Data Privacy**: By exclusively using a local LLM, we ensure that the content of user emails and other inputs is never transmitted to a third-party service, preserving user privacy.
2) **Credential Security**: We adhere to security best practices by never storing passwords in plaintext. All passwords are securely hashed using `bcrypt`.

D. FUTURE WORK

Addressing the present constraints and enhancing the system's functionality will be the main goals of future work. Some of the planned improvements are:

- **Implement Real-time Monitoring**: Switch from polling to an `IMAP IDLE`-based method for immediate email alerts
- **Attachment Scanning**: Include tools for examining email attachment content.
- **Multi-Account Support**: Redesign the backend to enable users to keep an eye on several email accounts at once.
- **Enhanced Voice Commands**: Increase the voice commands' vocabulary to accommodate more intricate inquiries.

VII. CONCLUSION

This paper has presented a novel, multi-modal phishing and social engineering detection system powered by a local AI agent. By integrating text, URL, QR code, and voice analysis with a privacy-preserving LLM, we have created a robust tool that overcomes many limitations of traditional and single-modality security solutions. The successful implementation of automated email monitoring, secure credential management, and an interactive Streamlit UI demonstrates the viability of this agentic approach. This work lays the foundation for more advanced, autonomous cybersecurity systems that can adapt to the ever-evolving landscape of digital threats.

ACKNOWLEDGMENT

The authors express their sincere gratitude to their supervisor, Dr. Satish C J, for his invaluable guidance, insightful feedback, and constant support throughout this research. This work was made possible by the resources and academic environment provided by the School of Computer Science and Engineering at the Vellore Institute of Technology (VIT), Vellore.

REFERENCES

[1] A. S. K. Joseph and S. Srinivasan, "Anti-Phishing Adaptive AI Systems: Efficiently Countering Social Engineering Attacks by Real-Time Analysis of Email Content," in *Proc. 2025 Int. Conf. Comput. Innovations Eng. Sustainability (ICCIES)*, Coimbatore, India, 2025, pp. 1–6, doi: 10.1109/ICCIES63851.2025.11032758.

[2] H. Patel, U. Rehman and F. Iqbal, "Evaluating the Efficacy of Large Language Models in Identifying Phishing Attempts," in *Proc. 2024 16th Int. Conf. Human System Interaction (HSI)*, Paris, France, 2024, pp. 1–7, doi: 10.1109/HSI61632.2024.10613528.

[3] S. H. Shah, A. Garu, D. N. Nguyen and M. Borowczak, "Feature Selection Framework for Optimizing ML-Based Malicious URL Detection," in *Proc. 2024 Cyber Awareness Res. Symp. (CARS)*, Grand Forks, ND, USA, 2024, pp. 1–6, doi: 10.1109/CARS61786.2024.10778786.

[4] A. Park, B. Jaculina and D. Dimitrov, "Demo: LLM-Based Social Engineering Detection System (LSED)," in *Proc. 2025 Silicon Valley Cybersecurity Conf. (SVCC)*, San Francisco, CA, USA, 2025, pp. 1–3, doi: 10.1109/SVCC65277.2025.11133640.

[5] M. Khonji, Y. Iraqi and A. Jones, "Phishing Detection: A Literature Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 2013, doi: 10.1109/SURV.2013.032213.00009.

[6] S. Salloum, T. Gaber, S. Vadera and K. Shaalan, "Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey," *Procedia Comput. Sci.*, vol. 189, pp. 19–28, 2021, doi: 10.1016/j.procs.2021.05.077.

[7] U. Zara, K. Ayub, H. Khan, A. Daud, T. Alsahfi and S. Gulzar, "Phishing Website Detection Using Deep Learning Models," *IEEE Access*, early access, pp. 1–1, 2024, doi: 10.1109/ACCESS.2024.3486462.

[8] R. Mahajan and I. Siddavatam, "Phishing Website Detection Using Machine Learning Algorithms," *Int. J. Comput. Appl.*, vol. 181, no. 23, pp. 45–47, Oct. 2018, doi: 10.5120/ijca2018918026.

[9] Ş. Şentürk, E. Yerli and İ. Soğukpınar, "Email Phishing Detection and Prevention by Using Data Mining Techniques," in *Proc. 2017 Int. Conf. Comput. Sci. Eng. (UBMK)*, Antalya, Turkey, 2017, pp. 707–712, doi: 10.1109/UBMK.2017.8093510.

[10] A. A. Alhemyari, Z. A. A. Hassan, M. A. Saeed, R. A. M. Alselwi and R. A. Alhemyari, "Real-Time Phishing Attack Detection Using Machine Learning," in *Proc. 2025 5th Int. Conf. Emerging Smart Technol. Appl. (eSmarTA)*, Ibb, Yemen, 2025, pp. 1–8, doi: 10.1109/eSmarTA66764.2025.11132204.

[11] P. Vidyasri and S. Suresh, "Automated Detection of Phishing Websites Using URL and HTML Structure and Behavioral Data Profiling," in *Proc. 2025 8th Int. Conf. Comput. Methodologies Commun. (ICCMC)*, Erode, India, 2025, pp. 1415–1421, doi: 10.1109/ICCMC65190.2025.11140804.

[12] N. Kumar, P. Sharma, S. Shilpa, K. Kaur, P. Singla and N. Goyal, "A Study on Phishing Detection Techniques," in *Proc. 2025 7th Int. Conf. Comput. Intell. Commun. Technol. (CCICT)*, Sonepat, India, 2025, pp. 313–318, doi: 10.1109/CCICT65753.2025.00056.

[13] M. K. S., Shankaramma, N. S. B. and S. L., "Evaluation and Performance of a Phishing Electronic Mail Detection Model," in *Proc. 2025 8th Int. Conf. Comput. Methodologies Commun. (ICCMC)*, Erode, India, 2025, pp. 1242–1246, doi: 10.1109/ICCMC65190.2025.11140705.

[14] Y. Lan, "Chat-Oriented Social Engineering Attack Detection Using Attention-Based Bi-LSTM and CNN," in *Proc. 2021 2nd Int. Conf. Comput. Data Sci. (CDS)*, Stanford, CA, USA, 2021, pp. 483–487, doi: 10.1109/CDS52072.2021.00089.

[15] N. Altwaijry, I. Al-Turaiki, R. Alotaibi and F. Alakeel, "Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models," *Sensors*, vol. 24, no. 7, pp. 2077, 2024, doi: 10.3390/s24072077.

APPENDIX A ADDITIONAL CODE SNIPPETS

The following Python code snippet illustrates the logic for analyzing text input using the local Ollama client.

```
# Excerpt from phishing_detector.py

def detect_text(text):
   """
   Analyzes text content for phishing indicators
   using a local LLM.
   """
   # Construct a prompt for the language model
   prompt = f"Analyze the following text for
        phishing or " \
         f"social engineering cues. Provide a risk
             level " \
         f"(Safe, Warning, or Danger) and a brief,
             " \
         f"one-sentence explanation for your
             assessment. " \
         f"Text: '{text}'"

   try:
      # Send the prompt to the local Ollama client
      response = ollama_client.chat(
         model='mistral',
         messages=[{"role": "user", "content":
             prompt}]
      )
      # Extract the content from the response
      text_result = response['message']['content']
      return text_result
   except Exception as e:
      # Fallback to mock analysis if the LLM call
          fails
      return f"LLM error: {str(e)}. Using mock
          analysis: " \
            f"{mock_phishing_analysis(text)}"
```

```
def extract_danger_level(analysis_result):
    """
    Extracts a standardized danger level from the
    LLM's text response.
    """
    analysis_lower = analysis_result.lower()
    if "danger" in analysis_lower:
        return "DANGER"
    elif "warning" in analysis_lower:
        return "WARNING"
    elif "safe" in analysis_lower:
        return "SAFE"
    else:
        return "UNKNOWN"
```

Listing 1. Excerpt from phishing_detector.py

APPENDIX B DETAILED TEST DATA

Below are examples of test inputs used to evaluate the system's performance.

- **Benign Text**: "Hi team, just a reminder about the meeting tomorrow at 10 AM. The agenda is attached. Please review it beforehand."
  - **Expected Output**: `SAFE`
- **Phishing Text**: "URGENT: Your bank account has been suspended due to suspicious activity. Please verify your identity immediately by clicking here to avoid permanent closure: http://bankofamerica.security-update.com/login"
  - **Expected Output**: `DANGER`
- **Suspicious URL**: http://paypal.com.login.support-id789.info/
  - **Expected Output**: `DANGER` (Reason: Misleading subdomains, lack of HTTPS)
- **Benign URL**: https://www.ieee.org/
  - **Expected Output**: `SAFE`

GARIMA MANGAL (22BCT0255) is a B.Tech. student in Computer Science and Engineering with an IoT Specialization at the Vellore Institute of Technology. Her interests lie in AI, IoT security, and embedded systems. She is a proficient full-stack developer, creating web applications using technologies like React, Node.js, and MongoDB.

SHRIYA SINGH (22BCE2939) is a B.Tech. student in Computer Science and Engineering at the Vellore Institute of Technology. Her focus areas include full-stack web development using React, Node.js, and MongoDB, data analysis, and computer vision with OpenCV.

DIYA J (22BCE3463) is a B.Tech. student in Computer Science and Engineering at VIT, Vellore. She has experience building data pipelines with Databricks DLT and PySpark. She is proficient in Java, Python, C/C++, and SQL , with experience in web applications. security, and full-stack web development.