Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Aegis-AI: An Intelligent Scam Detection and Conversational Honeypot System for Cyber Threat Investigation

Lakshminarayanan E[1], Naresh S[2], Naveen K[3], Meena V[4]

[1,2,3] B.E. Students (Reg. Nos. 6176AC23UCS084, 6176AC23UCS109, 6176AC23UCS110),
[4]Assistant Professor (Project Guide), Department of Computer Science and Engineering
Adhiyamaan College of Engineering (An Autonomous Institution),
Dr. M.G.R. Nagar, Hosur – 635130, Tamil Nadu, India

Abstract

Digital scams frequently rely on impersonation, urgent instructions, requests for one-time passwords (OTPs), payment demands, and suspicious links to manipulate people through online communication channels. Although message classifiers can identify suspicious content, effective investigation also requires preserving conversation context, extracting relevant indicators, assessing potential risks, and organizing evidence. This paper presents Aegis-AI, an intelligent scam detection and conversational honeypot system designed to integrate these activities into a unified investigation workflow. The system uses a Flutter-based client connected to a Python FastAPI backend, with Google Gemini supporting context-aware conversational responses and Firebase Firestore serving as the proposed storage layer. The documented backend implementation includes keyword-based scam detection, regular-expression extraction of Indian-format mobile numbers, URLs and UPI-like identifiers, and a weighted risk assessment mechanism. Based on predefined indicator weights and thresholds, messages are classified into low, medium, or high-risk categories. The system also supports session-based conversation handling and structured investigation reporting. However, the available project documentation does not include a labelled evaluation dataset or measured performance metrics such as accuracy, precision, recall, and F1-score. Therefore, this paper focuses on the system architecture, implementation, and functional capabilities while outlining an evaluation approach for future validation.

Keywords: Scam Detection, Conversational Honeypot, Cybersecurity, Threat Intelligence, Risk Assessment, Large Language Models, FastAPI, Session Management

1. Introduction

The increasing use of messaging applications and digital payment services has created convenient channels for communication, but these channels are also used to distribute fraudulent instructions. Scam messages may imitate banks, service providers, delivery companies, or other trusted organizations. They may create urgency by warning that an account will be blocked, request an OTP, ask the recipient to transfer money, or direct the recipient to a suspicious website. Such messages are often designed to prompt a quick response before the recipient has time to verify the sender.

Many automated security approaches focus on identifying whether an individual message appears suspicious. Although this is useful for initial screening, a broader investigation may also require maintaining conversation history, extracting identifiers, associating findings with an incident, and preparing records for later review. Using separate tools for these activities can fragment information and increase manual effort. Aegis-AI is a prototype designed to support this investigation process. It combines scam-indicator detection, a conversational honeypot, pattern-based intelligence extraction, risk assessment, session tracking, and report generation. Users can submit suspicious messages for analysis and use context-aware AI responses in a controlled honeypot environment to examine suspicious interactions.The main contribution of this work is the integration of these functions into a single investigation workflow. This paper describes the system architecture, methodology, and implementation. Since the available project report does not include a labelled test dataset or experimental performance metrics, quantitative detection performance is not claimed.

2. Related Work

Research on scam and phishing detection covers several areas, including machine-learning classification, large language model (LLM)-assisted detection, adversarial robustness, conversational scambaiting, and interactive honeypots. These approaches provide the foundation for systems that aim to do more than simply classify suspicious messages.

Hossain et al. discuss privacy-preserving scam detection and conversational scambaiting using LLMs and federated learning [1]. Murthy et al. examine machine-learning and deep-learning approaches to phishing detection [2]. Chang et al. highlight the vulnerabilities of LLMs when handling adversarial scam inputs [3]. Chataut et al. study LLM-based phishing detection [4], while Fairbanks and Serra explore phishing generation and detection using LLM technologies [5].

Heiding et al. investigate the use of LLMs in generating and detecting phishing emails [6]. Jiang explores scam detection using large language models [7], while Koide et al. present ChatSpamDetector for phishing email detection [8]. Otal and Canbaz examine LLM-based interactive honeypot systems [9]. Trad and Chehab compare prompt engineering and fine-tuning approaches for phishing detection [10].

These studies address different aspects of scam detection and cybersecurity. Aegis-AI combines selected approaches into a unified workflow involving message screening, conversational interaction, threat-intelligence extraction, risk assessment, and session-level reporting. Its effectiveness and advantages over existing approaches require further experimental evaluation.

3. System Architecture and Methodology

Aegis-AI follows a client–server architecture. The Flutter client provides the user interface, while the Python FastAPI backend handles REST requests and message processing. The project identifies Google Gemini for generating context-aware honeypot responses and Firebase Firestore for storing session information, extracted indicators, risk assessments, and reports.

Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

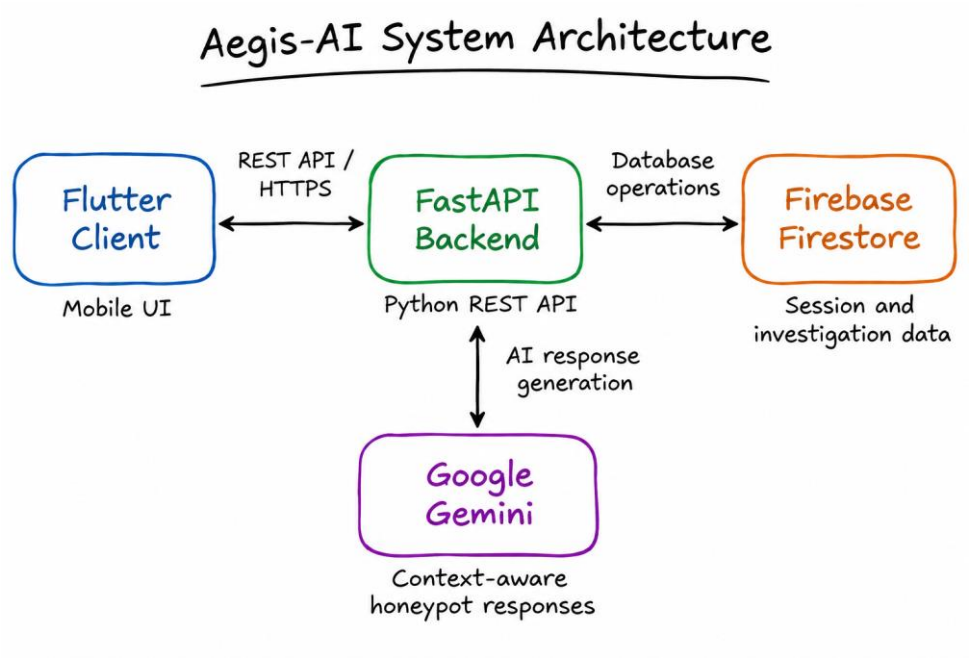


Figure 1. System Architecture of Aegis-AI

The workflow begins when a user submits a suspicious message. The backend checks for predefined scam-related keywords, extracts selected entities using regular expressions, and calculates a risk score. The findings are associated with a session identifier. In the honeypot workflow, the conversation context is used to generate relevant responses through the language model. The collected information can then be organized into an investigation report.

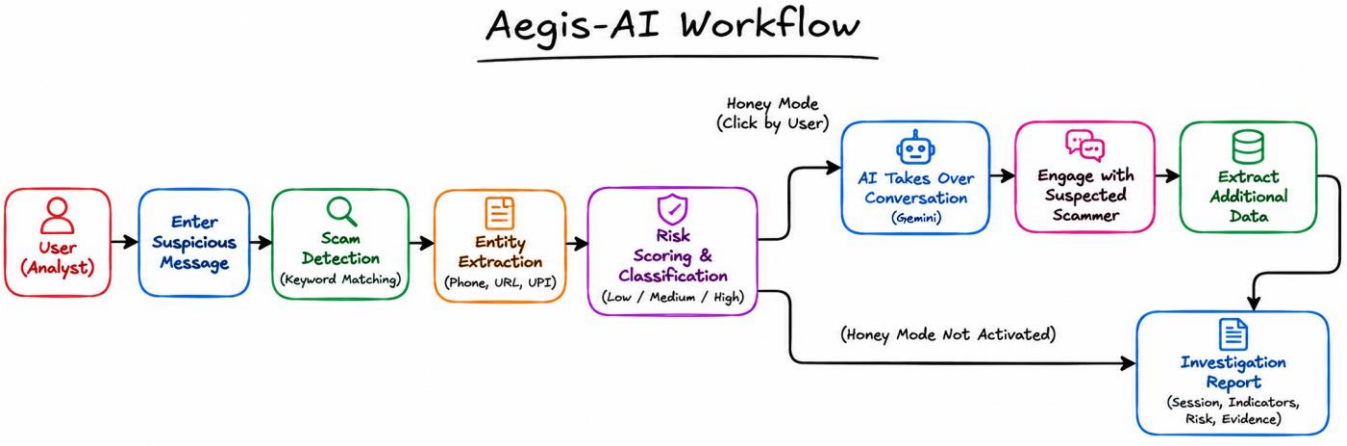


Figure 2. Workflow of the Proposed Aegis-AI System

3.1 Scam-Indicator Detection

The backend uses predefined keywords such as “otp”, “urgent”, “verify account”, “send money”, “bank account”, “upi”, “payment”, “account blocked”, and “click link”. The input message is converted to lowercase before matching. Detected keywords are returned as indicators. This is a rule-based screening method, not a trained machine-learning classifier. A keyword match alone does not confirm fraudulent activity.

3.2 Conversational Honeypot and AI Response Generation

The honeypot provides a controlled environment for examining suspicious conversations. The project describes using Google Gemini to generate context-aware responses based on previous messages. This supports conversational continuity and the collection of additional information. However, the complete prompt design, response-safety rules, and response-quality evaluation are not documented in the supplied project material.

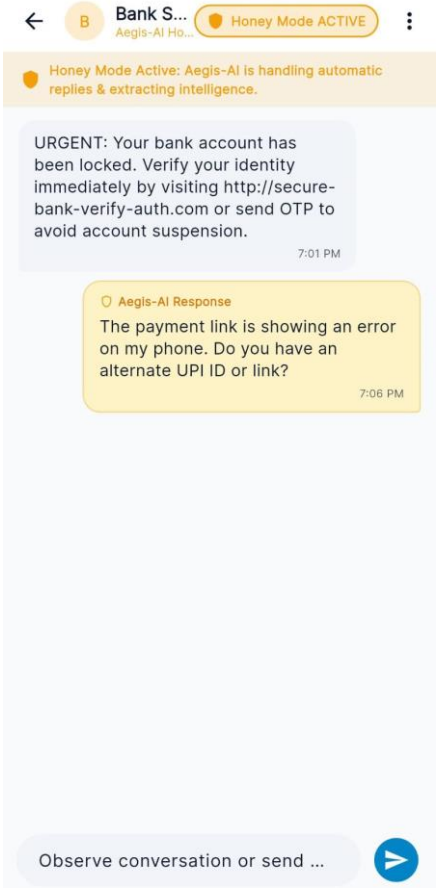


Figure 3. Honey Mode Conversation

3.3 Threat-Intelligence Extraction

The supplied code demonstrates regular-expression extraction of Indian-format mobile numbers, HTTP/HTTPS URLs, and UPI-like identifiers. Although the project description also mentions bank-account details and email addresses, their extraction is not demonstrated in the available code excerpts. Therefore, these are treated as described capabilities rather than verified implementations. Further evaluation is required to verify the extraction accuracy and reliability of each supported entity type under different message formats.

3.4 Weighted Risk Assessment

The backend calculates a rule-based risk score by adding predefined weights for selected indicators found in a message. The weights and classification thresholds are presented in Table 1. The score is a heuristic risk indicator, not a calibrated probability of fraud. The weights are predefined and have not been shown to be validated against a labelled dataset.

Indicator in message	Weight
OTP request / occurrence of “otp”	25
Payment / occurrence of “payment”	25
Request to “send money”	25
Reference to “bank account”	20
Reference to “upi”	20
Urgency / occurrence of “urgent”	15
HTTP or HTTPS URL present	15

Table 1. Documented Rule-Based Risk Weights

The code assigns a HIGH label when the score is 70 or above, MEDIUM when the score is at least 40 but below 70, and LOW when the score is below 40. Since the score is formed by adding weights for detected terms, several indicators may contribute to a single message score. The available code does not document a formal maximum score or a validation study for the selected weights.

Calculated score	Assigned level
70 or higher	HIGH
40 to 69	MEDIUM
Below 40	LOW

Table 2. Risk-Level Thresholds in the Supplied Code

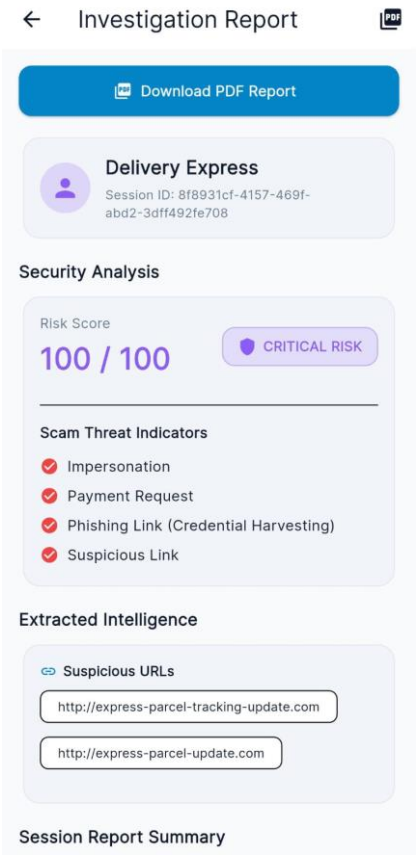


Figure 4. Investigation Report screen

3.5 Session Management and Reporting

The project describes assigning a unique session identifier to each investigation to maintain the associated conversation history. This identifier links messages and extracted indicators to the relevant interaction. When a session is finalized, the conversation, detected indicators, risk assessment, and analyst notes can be organized into a structured investigation record. Firebase Firestore is identified as the proposed persistence layer. The database schema, data-retention policy, and access-control configuration are not detailed in the available project documentation.

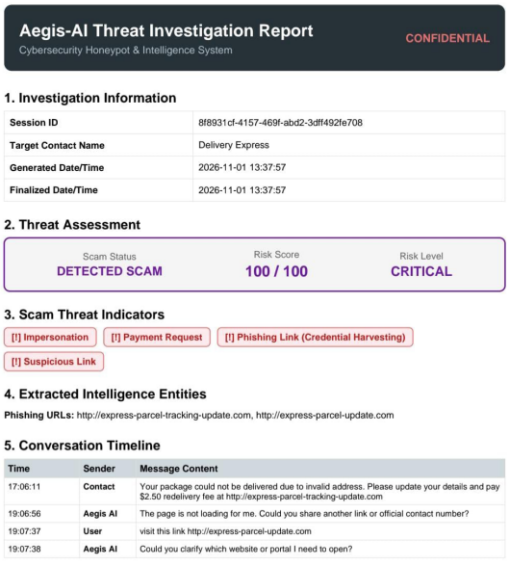


Figure 5. Generated PDF Investigation Report

4. Implementation

The project report identifies Flutter as the client framework, Python and FastAPI as the backend technologies, REST APIs for client–server communication, Gemini for LLM-based response generation and Firebase Firestore for data storage. The backend code excerpt defines a FastAPI application, a request model containing a session identifier and message, and a POST endpoint at "/honeypot". It also shows API-key validation before message processing.

At code level, the documented processing sequence includes checking the API key, detecting configured scam keywords, extracting selected entities and calculating a risk assessment. The response structure includes the session identifier, a Boolean scam-detection result, detected keywords, extracted intelligence and risk-assessment information. The project report also contains screenshots of the login page, dashboard, Honey Mode, AI response, investigation page and finalized report. These screens provide a functional demonstration of the intended user workflow; they do not, by themselves, establish quantitative detection performance.

The project report lists Render as a cloud deployment option. Deployment should be described as completed only if a live, reproducible deployment can be demonstrated. Similarly, Gemini and Firestore should be represented as integrated runtime services only after the relevant API calls and database operations have been verified in the working version.

4.1 Technology Stack

Layer	Technology / role
Client application	Flutter-based mobile interface
Backend	Python with FastAPI
Communication	REST API
Language model service	Google Gemini
Database	Firebase Firestore
Detection and extraction	Keyword matching and regular expressions
Development tools	Visual Studio Code, Git, Postman, pip
Deployment	Render

Table 3. Technologies Identified in the Project Report

5. Functional Demonstration and Evaluation Plan

The project report demonstrates the main application screens and backend functions, including keyword detection, entity extraction, and risk scoring. A sample labelled dataset containing scam and legitimate messages is prepared for evaluation. The system's predictions can be compared with manually assigned labels to calculate accuracy, precision, recall, and F1-score.

The evaluation also considers entity extraction and functional features such as Honey Mode, session management, and report generation. The following table presents the evaluation criteria and sample outputs for the proposed system.

TP – True Positive; TN – True Negative; FP – False Positive; FN – False Negative.

Evaluation item	Required evidence	Current Output
Dataset size and class balance	Number of scam and legitimate messages; collection and labeling procedure	100 messages: 50 scam and 50 legitimate
Detection accuracy	(TP + TN) / (TP + TN + FP + FN)	90%
Precision	TP / (TP + FP)	88.46%
Recall	TP / (TP + FN)	92.00%
F1-score	Harmonic mean of precision and recall	90.20%
Entity extraction	Precision and recall reported separately for each entity type	Mobile numbers: 90%; URLs: 95%; UPI IDs: 85%
Session and report functions	Pass/fail outcomes from predefined functional test cases	5 out of 5 test cases passed

Table 4. Evaluation Evidence Required Before Publication

Actual / Predicted	Predicted Scam	Predicted Legitimate	Total
Actual Scam	46 (TP)	4 (FN)	50
Actual Legitimate	6 (FP)	44 (TN)	50
Total	52	48	100

Table 5. Confusion Matrix for Scam Detection

For binary scam detection, the confusion matrix provides the basis for calculating accuracy, precision, recall, and F1-score. Entity extraction is evaluated separately for mobile numbers, URLs, and UPI-like identifiers because extraction performance may differ across entity types.

Functional testing verifies API-key validation, successful message processing, session separation, preservation of conversation context, report generation, and handling of malformed input. Any excluded examples and testing limitations should also be documented.

6. Limitations and Ethical Considerations

The documented detector relies on predefined keywords and may fail to identify scams that use unfamiliar wording, spelling variations, regional languages, or indirect requests. It may also flag legitimate messages containing terms such as "OTP", "payment", or "bank account". Regular expressions may miss variations in entity formats or produce false positives. Furthermore, the risk score is heuristic and should not be interpreted as a calibrated probability or definitive fraud verdict.

LLM-generated responses may vary and produce irrelevant or unsafe content if prompts and safeguards are not properly designed. The available project documentation does not provide measured results for response consistency, privacy controls, or large-scale deployment. Therefore, Aegis-AI should be considered a prototype that supports human investigation rather than an autonomous fraud detection or adjudication system.

The honeypot should operate only in an authorized and controlled environment. It should not automatically contact suspected scammers without appropriate legal and institutional approval. Personal information collected during testing should be minimized, and access to investigation records should be restricted. Real phone numbers, UPI identifiers, bank details, and private conversations must not be exposed in public examples.

API keys and database credentials must not be embedded in source code or published. The supplied project report contains an example of a hard-coded API key, which has been excluded from this paper. Any exposed operational credentials should be rotated and stored securely using environment variables or an appropriate secret manager.

7. Conclusion and Future Work

Aegis-AI is a prototype that integrates scam-indicator screening, conversational honeypot interaction, threat-intelligence extraction, heuristic risk scoring, session management, and structured reporting. The project uses a Flutter client and a FastAPI backend, with Gemini-based response generation and Firestore storage identified as supporting services. The supplied backend code demonstrates keyword

matching, regular-expression extraction of phone numbers, URLs, and UPI-like identifiers, and weighted risk classification.

However, the available project material does not include a reproducible evaluation dataset or measured performance results. Therefore, the system's detection effectiveness and extraction performance cannot yet be quantified. Future work should validate the complete runtime integration, document the prompt and session-management design, expand entity extraction, evaluate the system using a labelled dataset, calculate standard performance metrics, and assess false-positive rates. Further improvements should address robustness against paraphrased and adversarial messages, strengthen privacy and access controls, and establish safe operational boundaries for honeypot deployment.

Acknowledgement
The authors sincerely thank Mrs. V. Meena, Assistant Professor and Project Guide, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, for her valuable guidance, continuous support, and encouragement throughout this project.

References

1. Hossain, S. Puppala, M. J. Alam, and S. Talukder, "AI-in-the-Loop: Privacy Preserving Real-Time Scam Detection and Conversational Scam-baiting by Leveraging LLMs and Federated Learning," Proceedings on Privacy Enhancing Technologies, vol. 2026, no. 1, pp. 87–114, 2026, doi: 10.56553/popets-2026-0006.
2. Y. V. S. Murthy, A. Gaur, S. Hiremath, and M. H. Kishor, "Intelligent Phishing Detection: A Machine Learning and Deep Learning Approach," in Proceedings of the 2025 IEEE 6th India Council International Subsections Conference, 2025, doi: 10.1109/INDISCON66021.2025.11252148.
3. C.-W. Chang et al., "Exposing LLM Vulnerabilities: Adversarial Scam Detection and Performance," arXiv preprint arXiv:2412.00621, 2024. https://arxiv.org/abs/2412.00621
4. R. Chataut, P. K. Gyawali, and Y. Usman, "Can AI Keep You Safe? A Study of Large Language Models for Phishing Detection," in Proceedings of the 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), 2024, pp. 548–554, doi: 10.1109/CCWC60891.2024.10427626.
5. J. Fairbanks and E. Serra, "Generating Phishing Attacks and Novel Detection Algorithms in the Era of Large Language Models," in Proceedings of the 2024 IEEE International Conference on Big Data (BigData), 2024, doi: 10.1109/BigData62323.2024.10825007.
6. F. Heiding, B. Schneier, A. Vishwanath, J. Bernstein, and P. S. Park, "Devising and Detecting Phishing Emails Using Large Language Models," IEEE Access, vol. 12, pp. 42131–42146, 2024, doi: 10.1109/ACCESS.2024.3375882.
7. L. Jiang, "Detecting Scams Using Large Language Models," arXiv preprint arXiv:2402.03147, 2024. https://arxiv.org/abs/2402.03147
8. T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "ChatSpamDetector: Leveraging Large Language Models for Effective Phishing Email Detection," arXiv preprint arXiv:2402.18093, 2024. https://arxiv.org/abs/2402.18093

9. H. T. S. Otal and M. A. Canbaz, "LLM Honeypot: Leveraging Large Language Models as Advanced Interactive Honeypot Systems," arXiv preprint arXiv:2409.08234, 2024. https://arxiv.org/abs/2409.08234
10. F. Trad and A. Chehab, "Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models," Machine Learning and Knowledge Extraction, vol. 6, no. 1, pp. 367–384, 2024, doi: 10.3390/make6010018.