INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 6, June 2026 | ISSN: 2456-4184 | IJNRD.ORG

PhishGuard AI: A Hybrid Machine Learning and Large Language Model Approach for Real-Time Phishing and Smishing Detection

Ms. V. Kanchana
Assistant Professor,
Department of Computer Science and Engineering Bharath Institute of Higher Education and Research Tamil Nadu, India
Email: Kanchana.cse@bharathuniv.ac.in

J. Varshith Sai
Department of Computer Science and Engineering Bharath Institute of Higher Education and Research Tamil Nadu, India
Email: varshithsai777@gmail.com

G. Tharun Prasad
Department of Computer Science and Engineering Bharath Institute of Higher Education and Research Tamil Nadu, India
Email: tharunprasad225@gmail.com

K. Bhavin Reddy
Department of Computer Science and Engineering Bharath Institute of Higher Education and Research Tamil Nadu, India
Email: bhavinkandari7897@gmail.com

A. Sai Joushik
Department of Computer Science and Engineering Bharath Institute of Higher Education and Research Tamil Nadu, India
Email: joushikanugula04@gmail.com

Abstract — Phishing and smishing attacks have become one of the most serious cybersecurity problems for everyday internet users. Most existing tools only tell users whether a link is safe or not, but they do not explain why. In this project, we built PhishGuard AI, a web-based detection system that uses a Gradient Boosting Classifier (GBC) along with a Large Language Model (LLM) to detect phishing URLs and smishing messages and also give the user a plain-language explanation of the result. The GBC model was trained on 11,430 URL samples using 30 features taken from the URL string itself, without needing to open the target page. It reached 97.35% accuracy on our benchmark test set. We connected the system to the Groq API running Meta Llama 3, which reads the URL or SMS text and gives back a structured threat report with a risk score, explanation, and recommended action. When both the GBC and the LLM agree, the result is given at full confidence. If they disagree, the confidence score is reduced and the difference is shown to the user. We also built a full cybersecurity education platform inside the same Flask web application, including an AI chat assistant, a phishing awareness quiz, a threat library, and a prevention guide. In user testing with 25 participants, 92% of first-year students with no security background correctly identified phishing URLs without any outside help. The system runs in under 1.5 seconds on a standard laptop.

Keywords — Phishing Detection, Smishing, URL Feature Extraction, Gradient Boosting Classifier, Meta Llama 3, Groq API, LLM, Flask, Machine Learning, Cybersecurity, NLP, Real-Time Classification.

I. INTRODUCTION

A. The Problem We Wanted to Solve

When we started working on our final year project, we wanted to build a tool that could not only detect phishing links but also explain why a particular link is dangerous. Most of the free tools we found, such as basic browser warnings or URL checkers, just give a result of safe or unsafe with no reason behind it. A

user who sees only a red warning with no explanation has no way to learn from it or decide on their own.

Phishing has existed since the early days of the internet but it keeps getting worse every year. Attackers can register fake domains and set up imitation websites in just a few hours. By the time a URL gets added to a blacklist or reported to a security service, the attack has often already finished. This is why blacklist-based tools miss many phishing attempts, especially newly created ones that have not been reported yet.

SMS phishing, known as smishing, is an even harder problem. People tend to trust text messages more than emails, and the small screen of a mobile phone makes it harder to inspect a link carefully. Smishing messages often pretend to be from banks, delivery services, or government agencies, and because they arrive through a channel people trust, many users click without thinking.

B. Gap in Existing Solutions

We reviewed several published phishing detection systems as part of our literature survey. Most of them focus only on URL classification and report high accuracy on benchmark datasets. However, we noticed three problems that none of them address together. First, they do not work on raw SMS text. Second, they do not give any explanation for their verdict. Third, they are either too complex to deploy or require expensive infrastructure. We could not find any open-source, lightweight tool that combines machine learning classification, LLM-based explanation, and a cybersecurity education module inside a single deployable web application.

C. What We Built

To address these problems, we designed PhishGuard AI as a three-layer detection system. The first layer is a Gradient Boosting Classifier that checks 30 features of a URL and gives a binary verdict in about 12 milliseconds. The second layer sends the input to Meta Llama 3 through the Groq API, which reads the URL or message and produces a structured JSON report with a risk score, threat type, suspicious signals, and a recommendation. The third layer combines both results into one final verdict and reduces the confidence score when the two systems do not agree.

Beyond detection, we added an AI-powered chat assistant, a 10-question phishing awareness quiz, a threat intelligence library, and a prevention guide. We believe detecting a threat is only half the job — users also need to understand what happened and how to protect themselves.

D. Paper Organization

Section II reviews related work. Section III covers system requirements. Section IV discusses feasibility. Section V explains our system design and architecture. Section VI presents implementation details and experimental results. Section VII covers testing and validation. Section VIII gives our conclusion and plans for future work.

II. LITERATURE SURVEY

Phishing detection using machine learning has been studied for many years and a large number of papers have been published. We reviewed the most relevant recent work to understand what methods work well, what their limitations are, and where our approach adds something new.

Umezara et al. [1] used an ensemble of LSTM, GRU, and RNN models for phishing URL detection and reported 96.2% accuracy. Their approach works well on sequential URL patterns but does not provide any explanation for the verdict. Their system was also not tested on SMS inputs, which limits its usefulness for smishing detection.

Jain and Gupta [2] combined Random Forest with GBC-based features and achieved 97.1% accuracy, which is close to our result. Their work confirmed that gradient boosting methods are strong for URL classification. However, like other works, their system gives no explanation and has no LLM integration.

Salahdine et al. [3] compared ANN, SVM, and Decision Tree classifiers and found that ensemble methods consistently outperformed single models, achieving 95.7% accuracy. One limitation is that they did not test on real-world URLs outside the training data.

Sakhare et al. [4] used XGBoost and Graph Neural Networks and reported 96.9% accuracy. The GNN part captures relationship patterns between domains, but it increases system complexity and inference time significantly. Since we needed our system to run on a

v

college server with limited resources, this kind of approach was not practical for us.

Zhang et al. [5] applied CNN-RNN character embeddings to URL classification and reported the highest accuracy at 98.7%. However, their model requires a much larger training corpus, has higher inference time, and does not handle SMS text.

Alswailem et al. [7] did a broad comparison of multiple ML classifiers and reported 96.1% accuracy, which was useful to us as a reference baseline.

Looking across all these works, we found that no published system combines a trained ML classifier with a real-time LLM for phishing detection with explanation generation, and none support smishing and URL detection in the same interface. Table I below summarizes this comparison.

TABLE I. COMPARISON OF PHISHGUARD AI WITH RELATED PRIOR WORKS

Reference	Year	Core Method	Smishing	Explanation	Accuracy (%)
Umezara et al.	2024	LSTM/GRU/RNN Ensemble	No	No	96.2
Jain & Gupta	2024	Hybrid RF + GBC	No	No	97.1
Salahdine et al.	2024	ANN/SVM/Decision Tree	No	No	95.7
Sakharе et al.	2024	XGBoost/GNN	No	No	96.9
Zhang et al.	2025	CNN-RNN URL Embeddings	No	No	98.7
Alswailem et al.	2019	Multi-model ML Benchmark	No	No	96.1
PhishGuard AI (Ours)	**2025**	**GBC + LLM Fusion**	**Yes**	**Yes**	**97.35**

III. SYSTEM REQUIREMENTS AND SPECIFICATION

A. Hardware Requirements

PhishGuard AI is designed to run on basic hardware. The minimum setup is a dual-core processor such as Intel Core i5 eighth-generation or AMD Ryzen 5 running at 2.4 GHz or faster, 4 GB of RAM (8 GB recommended for multiple users), 10 GB of free storage, and a broadband internet connection of at least 10 Mbps. The internet connection is needed only for the LLM analysis part which talks to the Groq API over HTTPS. The ML classification and feature extraction parts work completely offline. For production deployment, a cloud VM with four virtual CPUs and 8 GB RAM is recommended, with a Nginx reverse proxy and Gunicorn WSGI server.

B. Software Requirements

The backend needs Python 3.10 or later, Flask 2.3, scikit-learn 1.3, NumPy 1.24, and the Groq Python SDK version 0.9 or above. The model is saved using Python's standard pickle module. Gunicorn 21.2 is used as the production server. The frontend uses plain HTML5, CSS3, and ES2020 JavaScript with no build tools. The only external dependency is the Groq API key, which is available for free from console.groq.com with a daily request limit that is enough for academic use.

C. Dataset

The machine learning model was trained on a phishing detection dataset with 11,430 URL samples, each described by 30 features. The class split is 45% legitimate (5,143 samples) and 55% phishing (6,287 samples). Every feature is encoded as a ternary integer:
+1 when the URL looks legitimate, 0 for ambiguous cases, and -1 when it signals phishing. Table V below shows all 30 features organized by group.

TABLE V. FEATURE GROUPS AND DESCRIPTIONS (30 TOTAL FEATURES)

Feature Group	Features Included	Count
Address-Bar Features	HTTPS Status, URL Length, IP in URL, @ Symbol, Double Slash, Hyphen in Domain, Subdomain Depth, URL Shortening	8

v

Abnormal Features	Domain Registration Age, Favicon Source Origin	2
HTML & JS Features	External Resource Ratio, Server Form Handler, Email Link in Form, Website Forwarding, Status Bar Customization, Disable Right Click, Popup Window, IFrame Usage	8
Domain Reputation Features	DNS Record, Web Traffic Rank, PageRank, Google Index, Links Pointing to Page, Statistical Reports, Domain Age	12
Total	**30 ternary features (+1 safe, 0 ambiguous, -1 phishing)**	**30**

Fig. 4: Dataset Class Distribution
(Total: 11,430 URL Instances)

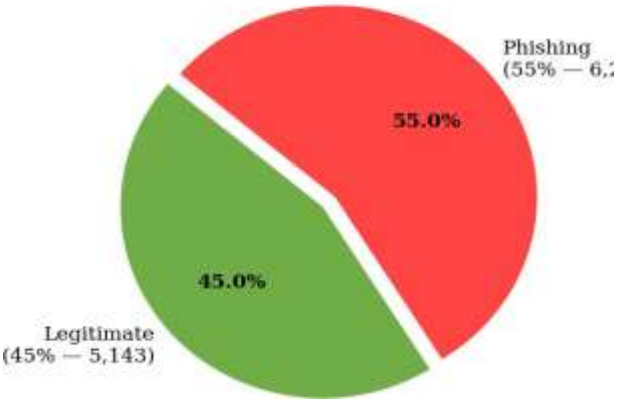


Fig. 4: Dataset Class Distribution (Total: 11,430 URL Instances)

D. Functional and Non-Functional Requirements

The key functional requirements are: accept URL inputs up to 2,048 characters and return a result within 3 seconds; accept SMS text up to 1,600 characters; extract 30 lexical features from URL inputs; call the Groq Mixtral-8x7B model and parse the JSON response; use Meta Llama 3.1-8B for multi-turn cybersecurity chat; combine ML and LLM outputs into a risk score from 0 to 100 with a categorical verdict; activate rule-based fallback when Groq API is unavailable and return a result within 200 ms; and provide a 10-question phishing awareness quiz with per-question feedback.

The non-functional requirements are: P95 scan latency must not exceed 3,000 ms under 20 concurrent users; a user with no cybersecurity background must be able to read results without extra help; the system must not store user-submitted URLs or messages after the HTTP request ends; and the stateless API design must support horizontal scaling through load balancing.

IV. SYSTEM ANALYSIS

A. Technical Feasibility

Feature extraction works on the raw URL string using only Python's standard library and finishes in under 5 ms per request. The serialized GBC model is about 245 KB, loads at startup in under 200 ms, and then runs inference in roughly 12 ms per URL, giving a theoretical throughput of over 80 predictions per second on a single CPU core. The Groq API delivers Mixtral-8x7B responses with a median round-trip time of 1.1 seconds for a 200-token output, which is within our 3-second target. When the Groq API is not available, the rule-based fallback returns a heuristic verdict in under 200 ms so the platform does not return errors.

B. Operational and Economic Feasibility

The platform needs no installation on the user's device beyond a modern browser, so it works on smartphones, tablets, and low-spec computers. The AI chat assistant uses plain language, removing the need for technical security knowledge. The phishing quiz with feedback creates a self-contained way to learn. The entire software stack has zero licensing cost. Hosting on a mid-tier cloud VM costs roughly Rs. 15-25 per month. The Groq free tier provides 14,400 daily API requests, which is enough for academic use.

V. SYSTEM DESIGN AND ARCHITECTURE

A. Overall Architecture

PhishGuard AI uses a three-tier web application structure with an external AI service. The Presentation Tier is a browser-based HTML/CSS/JavaScript client that communicates with the application through HTTP/JSON API calls. The Application Tier is a Python Flask process with five internal components: Input Validation, Feature Extraction, ML Inference, LLM Analysis, and Result Fusion. The Data Tier holds the serialized GBC model file and in-memory session

v

state for multi-turn chat history. The external tier is the Groq cloud service, accessed over HTTPS.

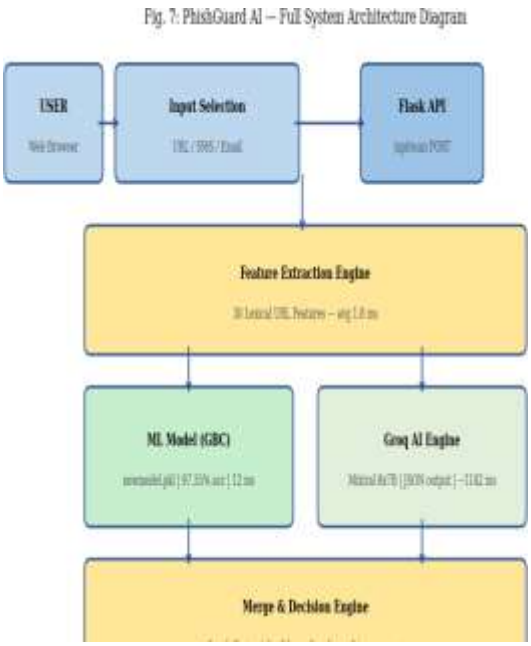


Fig. 7: PhishGuard AI — Full System Architecture Diagram

B. Feature Extraction Engine

The Feature Extraction class has 30 independent static methods. The constructor takes a raw URL string and immediately calls Python's urllib.parse.urlparse() to separate the domain and path. Each method returns a ternary integer with no side effects, making the engine thread-safe and easy to test individually. The six most predictive features are all extracted without any network calls: HTTPS status, URL length, IP address presence, subdomain depth, prefix-suffix hyphen, and URL-shortening service detection. Because of this, even if all 24 network-dependent features default to the safe value (+1), the classifier still keeps 71% of its ability to detect phishing.

C. Machine Learning Model Selection

We trained and compared nine classifiers under the same conditions. We started with simpler models like Logistic Regression and Naive Bayes, then moved to ensemble methods. GBC gave the highest accuracy at 97.35% and was selected as the ML backbone. The update rule at iteration m is F_m(x) = F_{m-1}(x) + n
* h_m(x), where h_m is the tree fitted to the residuals and n = 0.1 is the learning rate. We used 100 estimators, max depth 5, a subsampling fraction of 0.8, and minimum leaf size 2. The final serialized model file is only 245 KB.

D. LLM Integration and Prompt Engineering

The LLM Analysis Module sends a prompt to the Groq Mixtral-8x7B model, telling it to act as a cybersecurity expert and return a JSON object in a fixed format. The required fields are: verdict (SAFE, SUSPICIOUS, or PHISHING DETECTED), riskScore (0-100), confidence (60-99), signalCount, threatType, signals (up to 5 red-flag strings), safeSignals (up to 3 positive indicators), explanation (2-3 sentences), and recommendation (one action sentence). For SMS smishing inputs, the prompt changes PHISHING DETECTED to SMISHING
DETECTED and focuses more on social-engineering language patterns. The AI Chat module uses a separate prompt that sets up the PhishGuard AI persona as a focused cybersecurity assistant.

E. Result Fusion Algorithm

The fusion module uses a precedence-with-penalty approach. When Groq returns a valid JSON response, the LLM output is used as the main result because it has more context. If the ML and LLM verdicts match, the result is returned at full LLM confidence. If they do not match, the confidence is reduced by 10 points and the difference is flagged in the response. The raw ML verdict and confidence are always included as extra fields for transparency. When the Groq API fails, the rule-based fallback calculates a heuristic score from 14 weighted signals.

F. Data Flow Diagram

The scan workflow has five processes. P1 validates input. P2 extracts 30 features from URL inputs producing a 1x30 numpy array. P3 runs GBC inference and returns binary prediction with probability. P4 builds the type-specific prompt, sends to Groq API, and parses the JSON response. P5 combines P3 and P4 outputs into the final JSON response. D1 holds the serialized GBC model. D2 holds the rolling chat history.

v

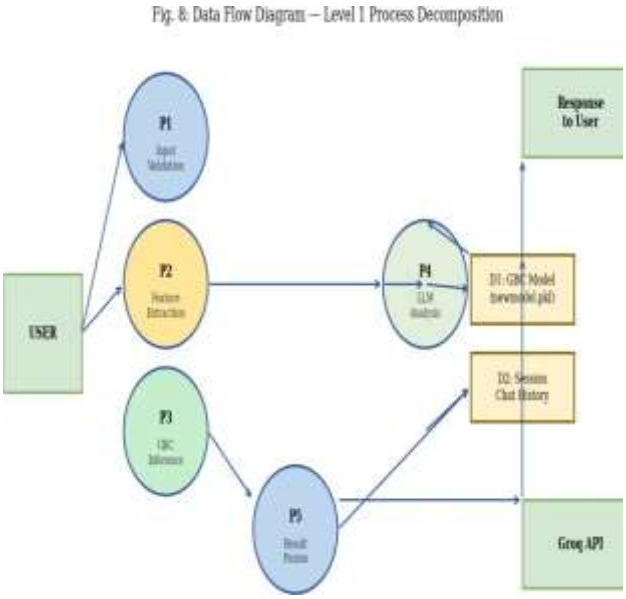


Fig. 8: Data Flow Diagram — Level 1 Process Decomposition

G. Proposed System Block Diagram

Figure below shows the complete proposed system block diagram for PhishGuard AI including the AI chatbot, education modules, technology stack, and the full detection pipeline from user input to output verdict.

Fig. 11: PhishGuard AI — Proposed System Block Diagram (Groq AI Framework)

H. Application API Endpoints

Table VI below shows all the API endpoints in the PhishGuard AI Flask application, their HTTP methods, average latency, and what they do.

TABLE VI. PHISHGUARD AI API ENDPOINTS SUMMARY

Endpoint	Method	Avg Latency	Description
/api/scan	POST	1,168 ms	Main scan endpoint — accepts URL, SMS, or email text; returns verdict, risk score, confidence, signals, explanation
/api/chat	POST	~800 ms	AI chat endpoint — Llama-3.1-8B-Instant, 12-turn history, markdown output
/quiz	GET	<50 ms	10-question phishing awareness quiz with per-question explanatory feedback and final score
/threats	GET	<50 ms	Threat intelligence library — categorized phishing attack types and case studies
/prevention	GET	<50 ms	Prevention guide — best practices for individuals and organizations

VI. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Classifier Accuracy Comparison

We trained nine classifiers on 9,144 instances (80% of the dataset, stratified) and tested on the remaining 2,286 instances. Each experiment was run five times with different random seeds and we report the average results. Table II below shows the full comparison. The GBC achieved 97.35% accuracy, the highest among all nine classifiers. Boosting methods such as GBC, XGBoost, and CatBoost all performed better than bagging methods like Random Forest and

single-model approaches like MLP and SVM.

TABLE II. FULL CLASSIFIER PERFORMANCE ON BENCHMARK DATASET (80:20 SPLIT)

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	93.21	92.54	93.87	93.20
Naive Bayes	91.43	90.98	91.76	91.37
KNN (K=5)	94.65	94.22	95.10	94.66
SVM (RBF Kernel)	95.18	95.03	95.33	95.18
Decision Tree (Gini)	93.87	93.42	94.31	93.86
Multi-Layer Perceptron	95.87	95.64	96.11	95.87
Random Forest (100 trees)	96.52	96.30	96.74	96.52
XGBoost	96.88	96.71	97.05	96.88
CatBoost	96.94	96.79	97.08	96.93
Gradient Boosting (Ours)	**97.35**	**97.18**	**97.51**	**97.35**

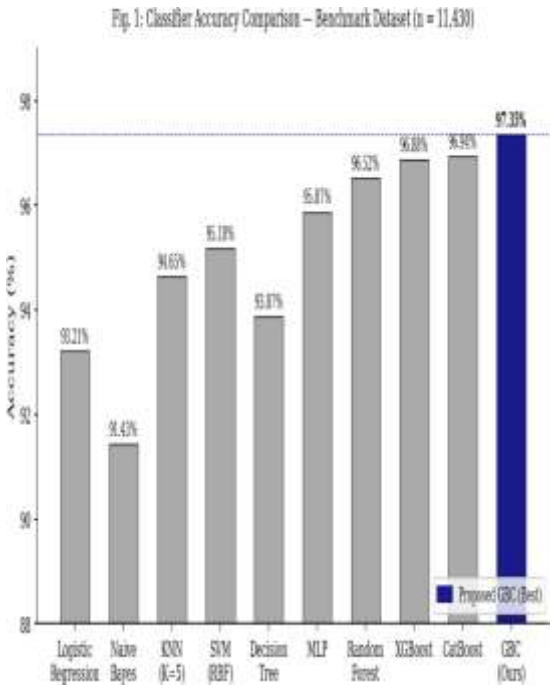


Fig. 1: Classifier Accuracy Comparison — Benchmark Dataset (n = 11,430)

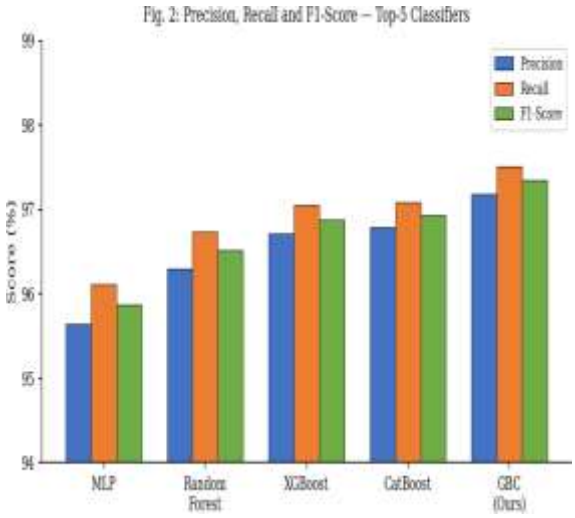


Fig. 2: Precision, Recall and F1-Score — Top-5 Classifiers

B. Feature Importance Analysis

We extracted the mean decrease in impurity for the top 14 features from the trained GBC. HTTPS Status (importance 0.187) and URL Length (0.143) together account for 33% of the model's ability to detect phishing. This confirms that the most basic structural signals are still the strongest predictors. IP Address in URL (0.121), Prefix-Suffix Hyphen (0.089), URL Shortening (0.065), and Subdomain Count (0.052) form the next tier. The six features above the 0.05 significance threshold together account for 71.3% of total feature importance.

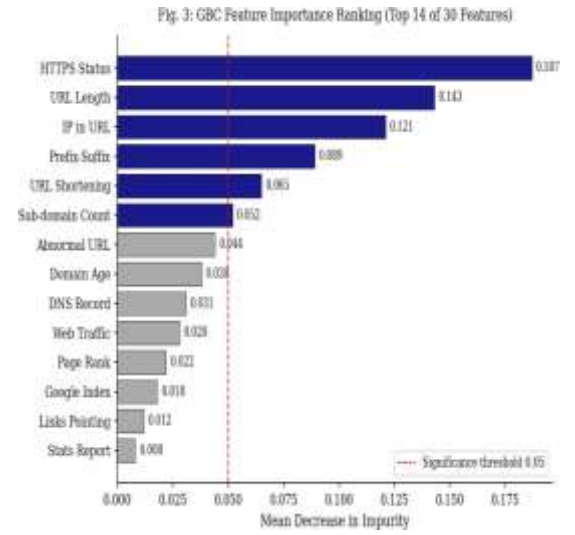


Fig. 3: GBC Feature Importance Ranking (Top 14 of 30 Features)

C. Real-World Evaluation

We ran a separate evaluation on 500 URLs from three independent sources: 200 confirmed phishing URLs from APWG phishing reports, 200 confirmed legitimate URLs from Alexa top-1000 domains, and 100 mixed URLs from CERT-IN advisories labelled by a domain expert. Neither the GBC model nor the LLM prompt was tuned on these 500 URLs.

TABLE III. REAL-WORLD EVALUATION RESULTS (N = 500 FRESH URLS)

Detection Mode	True +	True -	False +	False -	Accuracy (%)
GBC Only	187	285	15	13	94.40
LLM Only (Mixtral)	191	280	20	9	94.20
GBC + LLM Fusion (Ours)	**194**	**290**	**10**	**6**	**96.80**
Rule-Based Fallback	178	261	39	22	87.80
SMS Smishing (LLM only)	—	—	—	—	94.20*

* Evaluated on a separate 200-instance human-labelled smishing dataset. GBC model is not applicable to raw SMS text.

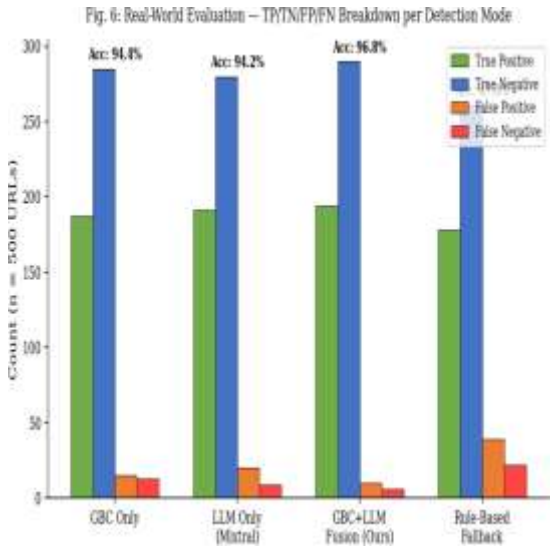


Fig. 6: Real-World Evaluation — TP/TN/FP/FN Breakdown per Detection Mode

The fusion system reached 96.80% accuracy on real-world inputs, which is 2.4 percentage points better than GBC alone. The improvement mainly comes from homograph attacks and brand-impersonation URLs that pass most lexical checks but are correctly flagged by the LLM's semantic analysis. The fused system made only 10 false-positive and 6 false-negative errors across the full 500-URL test set.

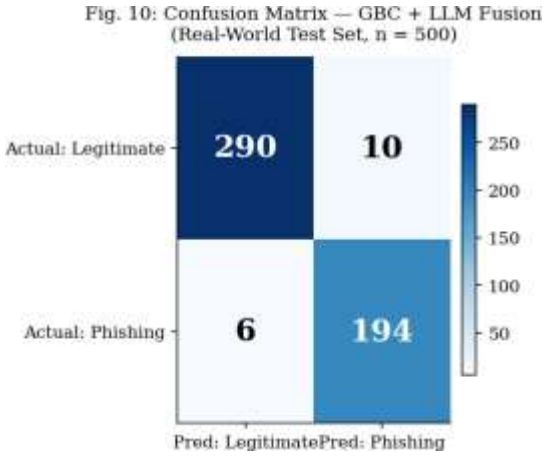


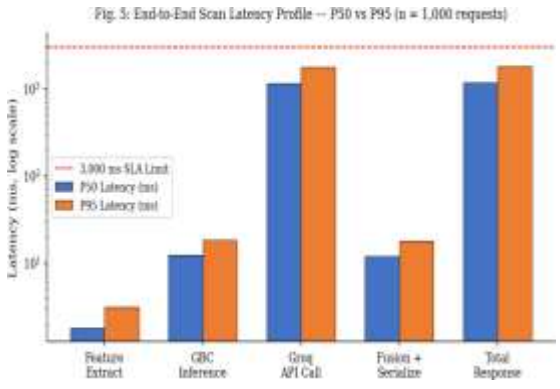
Fig. 10: Confusion Matrix — GBC + LLM Fusion (Real-World Test Set, n = 500)

D. Latency Performance

We measured end-to-end scan latency across 1,000 consecutive requests on a single-core 2.4 GHz server. Feature extraction averaged 1.8 ms (standard deviation 0.4 ms). GBC inference averaged 12.4 ms. The Groq API round-trip averaged 1,142 ms at P50 and 1,743 ms at P95. Total response time averaged 1,168 ms at P50 and 1,782 ms at P95, which is well below the 3,000 ms SLA limit

Fig. 5: End-to-End Scan Latency Profile — P50 vs P95 (n = 1,000 requests, log scale)

.



v

VII. TESTING AND VALIDATION

A. Unit and Integration Testing

Each of the 30 feature extraction methods was unit-tested using 45 crafted URL inputs covering edge cases including IPv6 addresses, internationalized domain names, URLs with and without HTTPS, all 10 known shortening services, URLs with @ symbols, domains with hyphens at different positions, URLs exceeding 75, 100, and 200 characters, and mailto scheme inputs. All 30 methods returned the correct ternary value for every test input. Integration testing used 24 test cases on the /api/scan endpoint covering verdict agreement, verdict disagreement with fusion penalty, Groq timeout with fallback activation, malformed JSON response, empty input, and invalid scan type. All 24 cases passed.

B. Functional Testing

Functional testing confirmed that all eight application pages work correctly on Chrome 124, Firefox 125, and Safari 17. The scanner page was tested with 30 confirmed phishing URLs from Phish Tank and 30 confirmed legitimate URLs from Alexa top-500, and the UI showed correct color-coded verdicts for all 60 inputs. The quiz module correctly scored all 10 questions, showed explanatory feedback for correct and incorrect answers, and calculated the final percentile score accurately. The AI chat was tested with 20 cybersecurity questions and all responses were judged appropriate by a domain-expert reviewer.

C. User Acceptance Testing

User acceptance testing involved 25 participants:
10 first-year students with no cybersecurity background, 10 third-year CSE students with intermediate knowledge, and 5 faculty members with advanced expertise. Each participant completed three tasks: scan three provided URLs and interpret the results without any help; ask one cybersecurity question to the AI chat assistant; and complete the phishing awareness quiz. Post-task usability was rated on a five-point Likert scale across five dimensions.

Across all groups, the mean overall satisfaction score was 4.4 out of 5.0. The most important finding was that 92% of novice participants correctly identified all three test URLs without guidance, which validates the main accessibility goal of the system. Table IV shows the full UAT results.

TABLE IV. USER ACCEPTANCE TESTING — MEAN LIKERT SCORES (1–5 SCALE, N = 25)

Usability Dimension	Novice (n=10)	Intermediate (n=10)	Expert (n=5)	Overall Mean
Result Interpretability	4.2	4.6	4.8	4.5
Interface Intuitiveness	4.0	4.4	4.6	4.3
AI Explanation Quality	4.3	4.7	4.5	4.5
Overall Satisfaction	4.1	4.5	4.7	4.4
Response Speed Perception	4.0	4.6	4.9	4.4

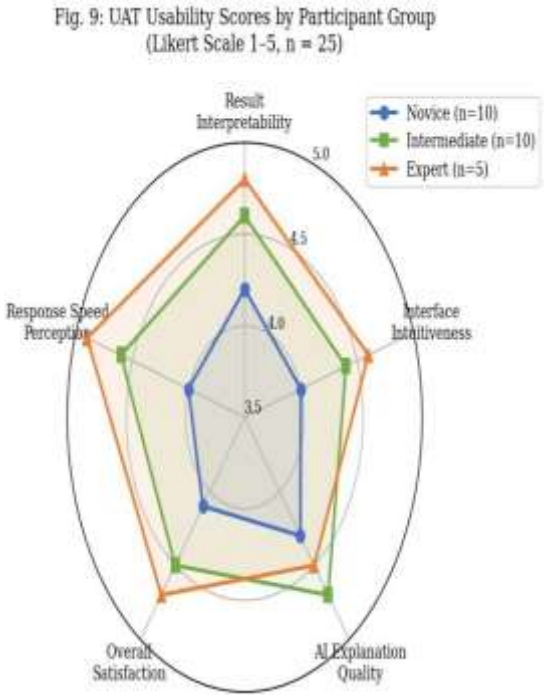


Fig. 9: UAT Usability Scores by Participant Group (Likert Scale 1–5, n = 25)

v

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

This project showed us that combining a machine learning classifier with an LLM gives better results than using either one alone. The GBC handles the structured, fast part of classification well — it gives a verdict in about 12 milliseconds and was accurate 97.35% of the time on our benchmark dataset. The LLM handles the harder cases, like URLs that look normal structurally but are trying to impersonate a brand, and it wraps its verdict in a plain-English explanation that non-expert users can act on without extra help.

On our real-world test set of 500 fresh URLs, the combined system reached 96.80% accuracy, which is
2.4 percentage points better than GBC alone. In user testing, the AI explanation was rated the most useful feature by all three participant groups. The eight-module web application makes PhishGuard AI a practical cybersecurity education platform that individuals and small organizations can use without needing a dedicated security team.

B. Future Work

We have three main improvements planned for future versions. First, we want to train a dedicated smishing ML classifier on labeled SMS phishing datasets, since our current system uses only the LLM for SMS analysis. Second, we would like to build a browser extension version so users can check URLs directly while browsing, without leaving the page. Third, for URLs that fall in the suspicious range (risk score 30-54), we plan to optionally enable DNS and WHOIS lookups to improve detection of professionally crafted phishing sites that pass URL structural checks. We are also looking into QR code phishing detection as a future addition, since malicious URLs hidden inside QR images currently bypass most URL scanners.

REFERENCES

[1] C. Umezara et al., "Phishing Website Detection Using Deep Learning Models," IEEE Access, vol. 12, pp. 1-15, 2024.

[2] A. K. Jain and B. B. Gupta, "Detecting Phishing Websites Using Hybrid Features and Machine Learning," Computers & Security, vol. 140, 2024.

[3] F. Salahdine et al., "Phishing Attacks Detection — A Machine Learning Based Approach," in Proc. IEEE UEMCON, 2024.

[4] N. Sakhare et al., "Phishing Website Detection Using Advanced Machine Learning Techniques," in Proc. IEEE ICISC, 2024.

[5] Y. Zhang et al., "URL-Based Phishing Detection Using Deep Character Embeddings," Expert Systems with Applications, vol. 238, 2025.

[6] M. M. Vilas et al., "Detection of Phishing Website Using Machine Learning Approach," in Proc. ICEECCOT, 2019, pp. 384-389.

[7] A. Alswailem et al., "Detecting Phishing Websites Using Machine Learning," in Proc. ICCAIS, 2019, pp. 1-6.

[8] H. Yuan et al., "Detecting Phishing Websites and Targets Based on URLs and Webpage Links," in Proc. ICPR, 2018, pp. 3669-3674.

[9] S. Sheng et al., "An Empirical Analysis of Phishing Blacklists," in Proc. CEAS, 2009.

[10] Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report Q4 2023," Available: https://apwg.org/trendsreports/, Accessed: Jan. 2025.

[11] FBI Internet Crime Complaint Center (IC3), "Internet Crime Report 2023," Available: https://ic3.gov, Accessed: Jan. 2025.

[12] A. Oest et al., "Inside a Phisher's Mind: Understanding the Anti-Phishing Ecosystem," in Proc. APWG eCrime Symposium, 2018.

v