iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84336

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Sentinel AI: An AI-Powered Visual Problem Detection and Intelligent Expert Recommendation System Using Computer Vision and Large Language Models

Sanket Kadhao[1], Abishai Amair[2], Bhairavi Sawale[3], Viddhi Menghare[4], Prof. Priya Khotele[5]
[1, 2, 3, 4]*Department of Computer Science and Engineering, St. Vincent Pallotti College of Engineering and Technology, Nagpur, India*
[5]*Professor and Head, Department of Computer Science and Engineering*

Abstract: ***Everyday technical failures, ranging from leaking pipes and cracked casings to obscure software exceptions, are difficult for non-specialists to diagnose and even more difficult to resolve without paying for an on-site expert visit. This paper presents Sentinel AI, a multi-role, AI-powered problem-diagnosis and expert-marketplace platform that unifies computer-vision-based hardware fault detection with large-language-model-based software log analysis under a single "Universal Analysis" pipeline. The system ingests either a photograph of a damaged physical asset or a snippet of code, logs, or an error trace, classifies the artifact type, estimates severity and confidence, produces a plain-language technical explanation with a recommended remediation, and, where the issue exceeds what a user can safely resolve alone, routes a structured request to the nearest available domain expert through a real-time, Socket.io-based messaging layer with SMS escalation via Twilio. Three cooperating web portals — Customer, Expert, and Admin — are backed by a Node.js/Express service layer and a MongoDB document store. This paper describes the system's architecture, the diagnostic and expert-matching pipeline, the technology stack, and the implemented user interfaces, and it discusses the platform's advantages, current limitations, and directions for future experimental validation.***

Keywords: Computer Vision, Large Language Models, Object Detection, YOLOv8, Fault Diagnosis, Expert Marketplace, Real-Time Communication, MERN Stack, Human-AI Collaboration, Retrieval-Augmented Generation

I. INTRODUCTION

A. *Background*

Modern life is mediated by an increasing density of electronic devices, mechanical fixtures, and software systems, and with this density comes a corresponding rise in the frequency with which ordinary users encounter faults they cannot interpret. A cracked phone screen, a corroded pipe joint, or an uncaught software exception all present the same underlying difficulty: the user can see or read the symptom but lacks the domain knowledge to translate that symptom into a diagnosis and a safe course of action. The conventional recourse — searching manuals, browsing forums, or waiting for a paid technician — is slow, inconsistent, and frequently disproportionate to the actual severity of the problem. Parallel advances in computer vision and large language models (LLMs) have made it practical to automate the first and most time-consuming step in this workflow: interpreting the symptom. Convolutional and transformer-based vision models can localize and classify visual defects with high reliability, while LLMs, particularly when combined with retrieval-augmented generation (RAG), can produce grounded, context-aware explanations for both visual and textual technical artifacts. However, most existing research systems stop at detection or explanation; they rarely close the loop by connecting a diagnosed problem to a qualified human expert when automated guidance is insufficient.

B. *Problems with Existing Approaches*

1) Domain specificity: prior systems are typically trained for a single vertical, such as infrastructure monitoring or automotive diagnostics, and do not generalize across hardware and software fault types.
2) Detection without resolution: many pipelines output a bounding box or a class label but stop short of an actionable, step-by-step remediation.
3) No human fallback: purely automated systems provide no structured mechanism for escalating a diagnosis to a human specialist

©IJRASET: All Rights are Reserved

when confidence is low or the required action is beyond a layperson's competence.

4) High computational and data overhead: multi-model ensembles (YOLO, Faster R-CNN, SSD in combination) increase inference cost and deployment complexity without a proportionate gain in usability.
5) Limited explainability and trust: black-box classifications reduce user confidence, particularly for safety-relevant faults such as water leaks or electrical damage.

C. Motivation

The motivation for this work follows directly from the gap identified above: users need a system that not only recognizes a problem from a photograph or a log excerpt but also tells them, in plain language, what the problem is, how severe it is, what it will likely cost to fix, and — when appropriate — connects them immediately with a verified expert who can take over the resolution. Sentinel AI was designed to close this loop within a single, coherent product rather than as a chain of independently maintained services.

D. Contributions

Our major contributions are:

1) A unified "Universal Analysis" pipeline that accepts either an image of a physical fault or a block of source code, logs, or error text, and routes each input type to an appropriate detection and reasoning path within a single interface.
2) A severity- and confidence-scored diagnostic output that pairs a technical root-cause explanation with a structured, step-by-step remediation and an approximate cost estimate, making the output directly actionable for non-technical users.
3) A real-time expert-marketplace layer, implemented with Socket.io and MongoDB-backed matching, that routes unresolved or high-severity cases to the nearest available domain expert and notifies that expert via Twilio SMS in addition to the in-app feed.
4) A three-role portal architecture (Customer, Expert, Admin) that separates end-user diagnostics, expert case management, and platform-wide administrative oversight while sharing a common data layer.
5) A working, end-to-end reference implementation on a Node.js/Express/MongoDB stack with a documented setup procedure, demonstrating feasibility of the proposed architecture rather than a purely conceptual design.

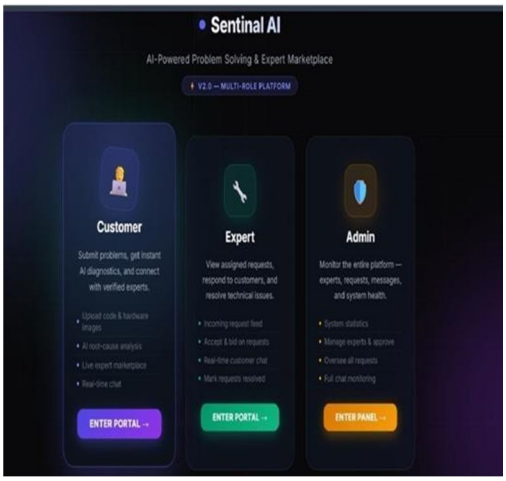


Fig. 1. Sentinel AI landing page showing the Customer, Expert, and Admin entry portals.

II. RELATED WORK

A growing body of research applies computer vision and, more recently, vision-language and large language models to the detection and explanation of physical and digital faults. This section reviews five representative systems spanning infrastructure monitoring, automotive diagnostics, hybrid detection architectures, environmental monitoring, and explainable defect analysis, and positions Sentinel AI relative to each.

M. I. Sheikh et al. propose InfraGPT Smart Infrastructure, an end-to-end pipeline that applies YOLO-family object detectors to street-level CCTV footage to identify and segment multiple categories of urban infrastructure defects, after which a vision-language model produces a natural-language summary of the detected issues. The system demonstrates the value of pairing detection with language-based summarization but is architecturally tied to fixed CCTV infrastructure, which limits its use outside monitored urban corridors and makes it unsuitable for on-demand, user-submitted diagnostics.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

I. Nanthini and K. Manivannan present Diagnostics-LLaVA, an adaptation of the LLaVA vision-language architecture fine-tuned for automotive fault diagnosis. The model accepts an image of a vehicle component and generates a contextual diagnostic explanation, improving on purely classification-based baselines by producing free-text reasoning. The approach, however, requires domain-specific fine-tuning and curated automotive datasets, and its generalization to non-automotive domains such as plumbing, electronics, or software has not been demonstrated.

K. Verma and G. Agarwal integrate multiple detection backbones — YOLO, Faster R-CNN, and SSD — with CNN-based classifiers and connect the resulting predictions to rule-based or robotic repair actions. Combining several detectors improves robustness but at the cost of architectural complexity, higher inference latency, and increased maintenance burden, which the authors themselves note as an open concern for real-time deployment.

A. Ayanzadeh and P. Dixit describe Wildfire VLM, which applies YOLOv12 to satellite imagery for early wildfire and smoke detection and couples the detector with a multimodal LLM for contextual risk assessment. The system is effective for large-scale environmental monitoring but depends on satellite data feeds such as Landsat and GOES and is not designed for close-range, consumer-submitted imagery of everyday technical faults.

K. Indrajit et al. incorporate explainable AI techniques, including Grad-CAM and SHAP, alongside CNN and transformer-based image models to generate interpretable defect reports. This work directly addresses the trust and transparency gap identified in Section 1.2, but the added explanation layers increase computational overhead and can slow real-time response, and the paper does not report empirical performance benchmarks that would allow direct comparison with detection-only baselines.

Across these five systems, a consistent pattern emerges: each improves one dimension of the pipeline — detection accuracy, explanation quality, domain breadth, or interpretability — but none combines hardware and software fault triage in a single interface, none includes a live human-expert fallback, and none reports a deployable, multi-role application architecture. Sentinel AI is designed to address these three gaps concurrently.

Table I. Comparison Of Related Work

Ref.	System / Technique	Domain	Advantages	Limitations	Research Gap Addressed by Sentinel AI
[1]	InfraGPT — YOLO + VLM summarization	Urban infrastructure (CCTV)	End-to-end detection-to-summary pipeline; real-time street monitoring	Tied to fixed CCTV networks; not usable for on-demand user uploads	Accepts ad-hoc, user-submitted images rather than fixed camera feeds
[2]	Diagnostics-LLaVA — fine-tuned vision-language model	Automotive diagnostics	Contextual, free-text diagnostic explanations	Requires domain-specific fine-tuning and curated datasets	Uses a general reasoning layer applicable across hardware and software domains
[3]	Multi-detector ensemble (YOLO, Faster R-CNN, SSD) + CNN	General object/defect detection	Improved detection robustness via ensembling	High architectural complexity and inference cost	Single lightweight detection path with clearly scoped severity output
[4]	Wildfire VLM — YOLOv12 + multimodal LLM	Environmental / satellite monitoring	Effective large-area early-warning detection	Dependent on satellite imagery; not suited to close-range faults	Targets close-range, everyday hardware and software problems
[5]	XAI-augmented defect detection (Grad-CAM, SHAP)	General defect analysis	Improved interpretability and user trust	Added overhead; no reported empirical benchmarks	Pairs plain-language explanation with a live expert fallback for low-confidence cases

III. PROBLEM STATEMENT

Real-world technical issues — device damage, mechanical faults such as pipe leakage, and software errors reported as stack traces or logs — present a two-part challenge: (i) accurately identifying the nature and severity of the problem from unstructured visual or textual input, and (ii) translating that identification into a course of action the user can either execute independently or hand off to a qualified expert with minimal friction. Existing systems, as surveyed in Section 2, typically address only the identification stage and are further restricted to a single fault domain (either vision-based hardware defects or text-based software errors, but not both).

Sentinel AI is designed to address the following problem: given a user-submitted image of physical damage or a block of code, error output, or log text, produce (a) a classification of the problem type, (b) a confidence score and severity rating, (c) a structured technical explanation and step-by-step remediation, (d) an approximate cost estimate where applicable, and (e) when the user requests further assistance, an automated match to the nearest available human expert with the relevant skill, communicated in real time through in-app messaging and SMS notification.

IV. PROPOSED METHODOLOGY

The proposed methodology follows a seven-stage pipeline: data ingestion, pre-processing, universal problem detection, classification, solution recommendation, expert-matching escalation, and output presentation. Each stage is described below.

A. Data Ingestion

The Customer Portal exposes a single Diagnostic Input Terminal (Fig. 2) that accepts two input modalities through one interface: a text area for software code, error logs, or stack traces, and afile-upload control for a photograph of physical/hardware damage. Both modalities can be submitted together, allowing the system to consider mixed evidence — for example, an error log accompanied by a photograph of the affected equipment.

B. Pre-processing

Uploaded images are validated, resized, and normalized prior to inference to ensure consistent input dimensions for the vision pipeline; submitted text is cleaned of non-essential whitespace and tokenized for downstream language processing. Metadata such as declared device type and user-provided location is extracted at this stage and passed forward to the contextual aggregator.

C. Universal Problem Detection

A single "Run Universal Analysis" action triggers a routing decision: image inputs are passed to the object-detection and classification path, while text inputs are passed to the language-reasoning path. The object-detection path is designed around a YOLOv8 backbone for problem-area localization, and a CNN/EfficientNet-based classifier assigns a fault category to the localized region (Fig. 5).

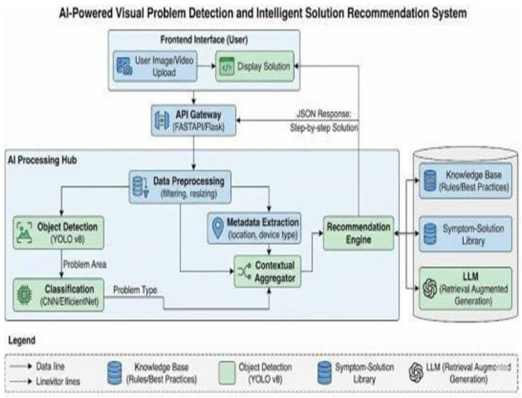


Fig. 2. System / block diagram of the AI-Powered Visual Problem Detection and Intelligent Solution Recommendation pipeline.

D. Contextual Aggregation and Classification

Outputs from the vision path, the metadata extractor, and, where present, the text-reasoning path are combined by a Contextual Aggregator, which reconciles the detected problem type with any declared device type or location before passing a unified problem representation to the Recommendation Engine.

E. Solution Recommendation

The Recommendation Engine queries a Knowledge Base of rules and best practices and a Symptom-Solution Library, and, for cases requiring free-text reasoning beyond fixed rules, consults a retrieval-augmented-generation (RAG) large language model. The engine returns a confidence score, a severity rating (e.g., Low, Medium, High), a technical-analysis summary, a step-by-step recommended solution, an associated risk statement, and an approximate cost estimate, as illustrated by the "Pipe Leakage Detected" result in Fig. 3, which reports 98% confidence, High severity, and a cost estimate of \$50–\$150 for professional remediation.

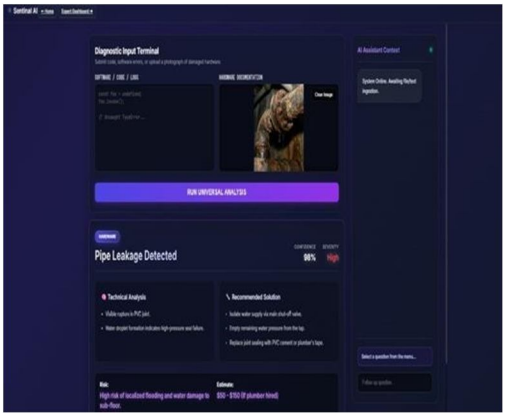


Fig. 3. Customer Portal Diagnostic Input Terminal with Universal Analysis result — Pipe Leakage Detected (98% confidence, High severity).

F. Expert-Matching Escalation

If the user elects to request expert help, a structured ticket containing the problem context and location metadata is written to MongoDB. The server queries registered experts filtered by declared skill category and current online status, and Twilio SDK dispatches an SMS notification to matching experts in addition to surfacing the request in the Expert Portal's Incoming Help Requests feed (Fig. 4). The first available expert to accept the ticket is bound to the requesting customer through a dedicated Socket.io room for real-time chat.

G. Output Generation

The final output is rendered in the Customer Portal as a structured diagnostic card (Fig. 2, Fig. 3) combining the classification label, confidence and severity indicators, the technical analysis, the recommended solution, the risk assessment, the cost estimate, and, where applicable, a Request Expert Help action that initiates the escalation pipeline described in Section 4.6.

V. SYSTEM ARCHITECTURE

Fig. 1 presents the system architecture, organized into a frontend interface, an API gateway, an AI Processing Hub, and a persistent data layer. The frontend interface exposes an image/video and text upload control and a solution-display panel. Submitted content is sent to an API Gateway layer (conceptually implemented with Express.js-style routing in the deployed system) which forwards requests to the AI Processing Hub and returns a JSON response containing the step-by-step solution.

Within the AI Processing Hub, a Data Preprocessing module performs filtering and resizing; an Object Detection module (YOLOv8) localizes the problem area within an image; a Classification module (CNN/EfficientNet) assigns a problem type to the localized region; a Metadata Extraction module captures location and device-type context; and a Contextual Aggregator merges these signals before handing off to the Recommendation Engine. The Recommendation Engine draws on three persistent resources: a Knowledge Base of rules and best practices, a Symptom-Solution Library, and a retrieval-augmented-generation LLM for cases requiring open-ended reasoning. At the application layer, three role-specific portals sit above this pipeline: the Customer Portal (Fig. 2, Fig. 3) for problem submission and diagnostic review, the Expert Portal (Fig. 4) for reviewing and accepting incoming requests, and an Admin Panel for platform-wide monitoring of experts, requests, and system health (Fig. 6 in the original interface set). All three portals share the same MongoDB-backed data layer, ensuring a consistent view of requests, expert availability, and chat history across roles.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

VI. IMPLEMENTATION

A. Frontend

The client interface is implemented in vanilla HTML, CSS, and JavaScript with a glassmorphic visual design, chosen to keep the deployment lightweight and framework-independent. The frontend integrates a Socket.io client for live chat and status updates, and uses FormData payloads for combined text-and-image submissions to the backend.

B. Backend

The backend runs on Node.js with the Express.js framework, exposing REST endpoints for request creation, expert registration, and administrative queries. Multer middleware handles multipart file uploads for hardware-damage photographs, and the Twilio SDK is integrated server-side to dispatch SMS alerts to matched experts.

C. Database

MongoDB, accessed through the Mongoose ODM, stores three principal document collections: Expert, Request, and Message, in addition to supporting collections for authentication and system statistics. The schema-flexible nature of MongoDB accommodates the heterogeneous fields produced by the diagnostic pipeline (confidence score, severity, cost range, risk text) without requiring a fixed relational schema.

D. Authentication and Registration

Experts register through a structured profile form capturing full name, email, mobile number, city/location, years of experience, and skill category (Fig. 4), which is the basis for the skill- and location-aware matching described in Section 4.6.

E. Real-Time Communication

Socket.io binds a customer and an assigned expert to a unique socket room immediately upon ticket acceptance, enabling typing indicators, live message delivery, and status updates without page refresh, while Twilio SMS provides an out-of-band notification channel for experts who are not actively viewing the portal.

F. Admin Controls

The Admin Panel provides platform-wide visibility: system statistics, expert management and approval, oversight of all open and resolved requests, and full chat monitoring for quality assurance and dispute resolution.

TABLE V. TECHNOLOGY STACK SUMMARY

Layer	Technology	Purpose
Frontend	HTML, CSS, JavaScript (vanilla)	Responsive, glassmorphic dashboards for all three portals
Real-time layer	Socket.io	Live chat, typing indicators, status updates
Backend runtime	Node.js	Server-side execution environment
Backend framework	Express.js	REST API routing and middleware
File handling	Multer	Multipart form-data / image upload middleware
Notifications	Twilio SDK	SMS dispatch to matched experts
Database	MongoDB with Mongoose ODM	Document storage for Expert, Request, and Message entities
AI reasoning (proposed API layer)	FastAPI / Flask, YOLOv8, CNN/EfficientNet, RAG-LLM	Object detection, classification, and language-based recommendation

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

TABLE VI. SYSTEM MODULES AND RESPONSIBILITIES

Module	Responsibility
Data Preprocessing	Filtering and resizing of uploaded images prior to inference
Object Detection (YOLOv8)	Localization of the problem area within an image
Classification (CNN/EfficientNet)	Assignment of a fault category to the localized region
Metadata Extraction	Capture of location and device-type context
Contextual Aggregator	Reconciliation of vision, text, and metadata signals
Recommendation Engine	Generation of severity, confidence, solution steps, risk, and cost output
Expert Matching Service	Skill- and availability-filtered query against the Expert collection
Notification Service (Twilio)	SMS dispatch to matched experts
Real-Time Messaging (Socket.io)	Live chat between customer and assigned expert

VII. EXPERIMENTAL SETUP

The reference implementation was exercised locally using Node.js with MongoDB bound to 127.0.0.1:27017 (or a MongoDB Atlas cloud URI), with environment variables for the Twilio SDK configured in a backend/.env file. Dependencies — Express, Socket.io, Mongoose, and the Twilio SDK — were installed via npm inside the backend directory, and the server was started with node server.js, confirming a successful boot on port 3000 and a successful MongoDB connection. Functional testing followed a fixed protocol: an expert account was registered with a specific skill tag (e.g., Plumber) and set to an active online status; a matching problem image was then submitted from the Customer Portal at http://localhost:3000, and an expert-view session was opened in a separate (incognito) browser window at http://localhost:3000/expert.html to verify that the request appeared in the Incoming Help Requests feed and that the resulting chat session functioned correctly end to end. This setup constitutes a functional and integration test of the application layer rather than a statistical evaluation of the underlying detection and language models; the distinction is discussed further in Section 8.1.

VIII. RESULTS AND DISCUSSION

This section discusses the implemented user interfaces and the qualitative behavior of the diagnostic pipeline, illustrated with figures captured from the working prototype.

1) Fig. 2 — Landing / Role Selection Page

The landing page (Fig. 2, top) presents the three entry points of the platform: Customer ("Submit problems, get instant AI diagnostics, and connect with verified experts"), Expert ("View assigned requests, respond to customers, and resolve technical issues"), and Admin ("Monitor the entire platform"). Each card enumerates its primary capabilities, giving new users an immediate sense of the platform's multi-role structure before they commit to a portal.

2) Fig. 3 — Diagnostic Input Terminal and Universal Analysis Result

Fig. 3 shows the Customer Portal's Diagnostic Input Terminal with two parallel input fields — a code/log text area and a hardware-photograph upload — beneath a single "Run Universal Analysis" action. Following analysis, the interface displays a structured result card labeled "Pipe Leakage Detected," tagged as a Hardware issue with 98% confidence and High severity. The card separates Technical Analysis ("Visible rupture in PVC joint," "Water droplet formation indicates high-pressure seal failure") from Recommended Solution (isolating the water supply, relieving residual pressure, and resealing the joint), and reports a Risk statement and a cost Estimate of \$50–\$150 if a plumber is hired, demonstrating the system's ability to pair a diagnosis with an economically meaningful recommendation.

3) Fig. 4 — Escalation to Expert Help

When the recommended self-service steps are insufficient, the same result card exposes a "Request Expert Help" action (Fig. 4), which initiates the escalation pipeline described in Section 4.6 and hands the ticket off to the Expert Portal's request feed.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

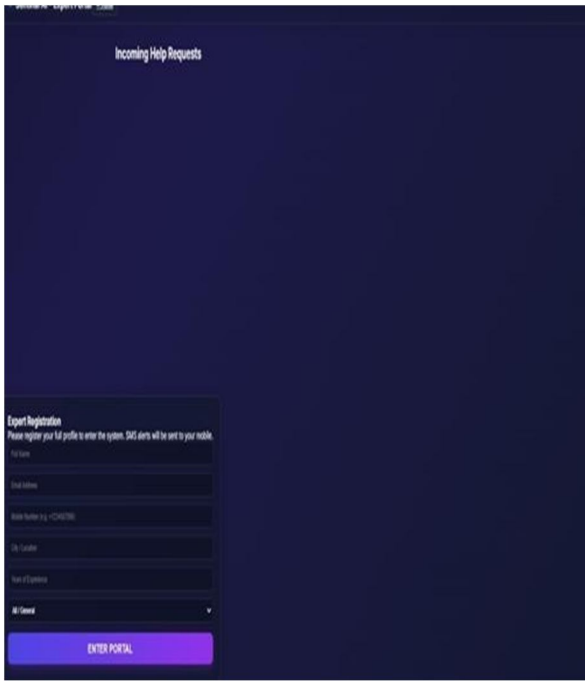


Fig. 4. Expert Portal showing the Incoming Help Requests feed and the Expert Registration form.

4) Fig. 5 — Expert Portal and Registration

The Expert Portal (Fig. 5) presents an "Incoming Help Requests" feed above an Expert Registration form capturing full name, email, mobile number, city/location, years of experience, and a skill-category selector (defaulting to "All / General"). This registration data is the basis for the skill- and availability-aware matching logic described in Section 4.6.

5) Fig. 6 — System / Block Diagram

Fig. 1 (the system architecture diagram, reproduced from the design documentation) traces the full path from frontend upload through the API gateway, the AI Processing Hub's detection and classification modules, and the Recommendation Engine, to the persistent Knowledge Base, Symptom-Solution Library, and RAG-LLM resources, and back to the frontend as a JSON step-by-step solution response.

A. Future Experimental Validation Needed

The results presented in this paper are qualitative and interface-level, drawn from a working prototype rather than a benchmarked evaluation. No controlled accuracy, precision/recall, latency, or user-study figures are reported here because such measurements have not yet been collected against a held-out, labeled dataset. Future work should evaluate, at minimum: (i) object-detection and classification accuracy (mAP, precision, recall) against a labeled multi-domain fault dataset; (ii) end-to-end latency of the Universal Analysis pipeline under realistic network conditions; (iii) expert-matching time-to-acceptance and SMS delivery latency under Twilio's production API; (iv) inter-rater agreement between the system's severity/confidence scores and assessments from licensed domain experts; and (v) a controlled user study measuring task success rate and time-to-resolution relative to a baseline of unassisted troubleshooting. Any accuracy or performance figures reported prior to such an evaluation should be treated as provisional.

IX. COMPARATIVE ANALYSIS

Table II positions Sentinel AI against the five related systems reviewed in Section 2 along dimensions relevant to a consumer-facing, multi-domain diagnostic product.

TABLE II. COMPARATIVE ANALYSIS AGAINST RELATED SYSTEMS

Capability	InfraGPT [1]	Diagnostics-LLaVA [2]	Multi-Detector Ensemble [3]	Wildfire VLM [4]	XAI Defect Detection [5]	Sentinel AI (Proposed)
Hardware fault detection	Yes (urban)	Yes (automotive)	Yes (general)	No	Yes (general)	Yes (general, user-submitted)
Capability	InfraGPT [1]	Diagnostics-LLaVA [2]	Multi-Detector Ensemble [3]	Wildfire VLM [4]	XAI Defect Detection [5]	Sentinel AI (Proposed)
Software / log analysis	No	No	No	No	No	Yes
Confidence & severity scoring	Partial	No	No	Partial	No	Yes
Cost estimation	No	No	No	No	No	Yes
Human-expert escalation	No	No	No	No	No	Yes (real-time, SMS + chat)
Multi-role application layer	No	No	No	No	No	Yes (Customer/Expert/ Admin)
Reported empirical benchmarks	Partial	Partial	No	Partial	No	Not yet (see Sec. 8.1)

TABLE IV. FEATURE COMPARISON ACROSS PORTALS

Feature	Customer Portal	Expert Portal	Admin Panel
Problem submission	Yes	No	No
AI diagnostic results	Yes	View only (assigned)	View only (all)
Real-time chat	Yes	Yes	Monitoring only
Expert registration	No	Yes	Approval
Request acceptance / bidding	No	Yes	No
Platform statistics	No	No	Yes
Full chat monitoring	No	No	Yes

X. ADVANTAGES

A. Advantages

By combining detection, explanation, cost estimation, and human escalation in one product, Sentinel AI reduces the number of separate tools a user must consult when facing an unfamiliar technical problem, and it gives platform operators a single point of oversight through the Admin Panel.

B. Applications

The architecture generalizes across several domains represented in the prototype's screenshots and design documentation, including household plumbing and hardware damage, general electronics troubleshooting, and software error diagnosis, and could plausibly extend to appliance repair, HVAC systems, and light industrial equipment.

C. Scalability

The use of a schema-flexible document database (MongoDB) and a stateless REST API layer allows new fault categories, expert skill tags, and knowledge-base entries to be added without a schema migration, and the Socket.io messaging layer can be horizontally scaled with a shared adapter (e.g., a Redis-backed Socket.io adapter) as concurrent chat sessions grow.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

D. *Performance and Reliability*

Real-time responsiveness depends on the availability and latency of the underlying detection and LLM services and on Twilio API responsiveness for SMS delivery; these have not yet been benchmarked (Section 8.1) and should be treated as an engineering risk rather than a proven property.

E. *Use Cases*

Representative use cases include a homeowner photographing a leaking pipe joint and receiving an immediate risk assessment and cost estimate (Fig. 3), a developer pasting an uncaught exception and receiving a root-cause explanation, and a platform operator using the Admin Panel to monitor expert response times across the marketplace.

TABLE III. SYSTEM ADVANTAGES

Advantage	Description
Domain-agnostic intake	A single Universal Analysis action accepts both hardware images and software text, removing the need for separate tools.
Advantage	Description
Actionable output	Every diagnosis is paired with a concrete, step-by-step remediation and a cost estimate rather than a bare classification label.
Human-in-the-loop fallback	Low-confidence or high-severity cases are escalated to a real, verified expert rather than left to an automated best guess.
Real-time responsiveness	Socket.io-based chat and Twilio SMS notification minimize the delay between a request and expert acceptance.
Role-separated administration	The Admin Panel gives platform operators visibility and control without exposing that functionality to end users.
Lightweight, portable stack	A vanilla JavaScript frontend and a Node.js/Express/MongoDB backend keep the deployment footprint small and framework-independent.

XI. LIMITATIONS

1) Image quality dependency: detection accuracy is expected to degrade under poor lighting, motion blur, or occlusion, none of which have yet been systematically characterized.
2) Internet dependency: both the AI Processing Hub and the real-time messaging layer require a persistent network connection; the system currently has no offline or degraded-connectivity mode.
3) Inference cost: LLM-based reasoning for open-ended cases and RAG retrieval introduce latency and computational cost that scale with request volume and have not yet been profiled at production scale.
4) Residual reliance on experts: cases falling outside the Knowledge Base or Symptom-Solution Library, or carrying low detection confidence, still require a human expert, meaning the system reduces but does not eliminate dependency on specialist availability.
5) Domain expansion required: the current knowledge base and expert skill taxonomy cover a limited set of fault categories (illustrated by plumbing and general hardware/software issues) and must be extended before broader deployment.
6) Unvalidated accuracy claims: as noted in Section 8.1, the confidence and severity scores shown in the prototype (e.g., 98% confidence in Fig. 3) reflect the interface design rather than a benchmarked model and require empirical validation before being presented as a performance guarantee.

XII. FUTURE WORK

1) Video-based diagnosis, allowing users to submit short video clips for faults that are difficult to capture in a single still image (e.g., intermittent mechanical noise).
2) IoT sensor integration, combining visual diagnosis with live sensor telemetry (e.g., pressure, temperature, vibration) for continuous rather than one-shot fault detection.
3) Augmented-reality repair guidance, overlaying step-by-step instructions on a live camera feed of the faulty equipment.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

4) Native mobile applications for iOS and Android to reduce upload friction and enable push notifications in place of, or alongside, SMS.
5) Predictive maintenance, using historical request data to flag equipment likely to fail before a visible symptom appears.
6) Offline or edge-deployed inference for low-connectivity environments.
7) Cloud-native deployment with autoscaling of the AI Processing Hub and a managed MongoDB Atlas cluster for production reliability.
8) Multi-language support for both the diagnostic text interface and the LLM-generated explanations.
9) A voice-assistant interface allowing users to describe a problem verbally rather than through text or image upload alone.

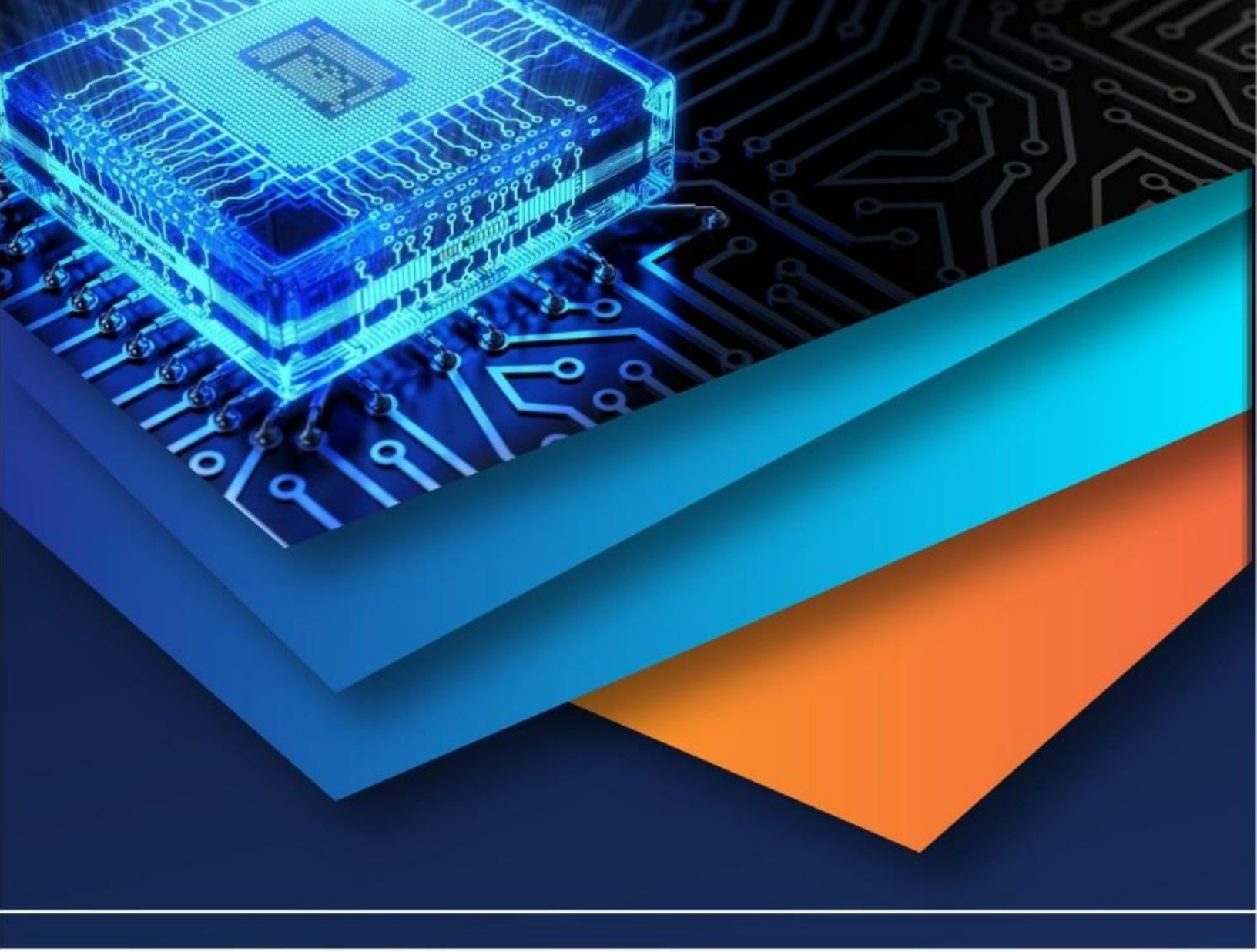
XIII. CONCLUSION

This paper presented Sentinel AI, a multi-role platform that unifies computer-vision-based hardware fault detection, LLM-assisted software diagnosis, and a real-time expert marketplace within a single Customer/Expert/Admin architecture. The system's Universal Analysis pipeline produces confidence-scored, severity-rated diagnoses paired with actionable remediation and cost estimates, and escalates unresolved or high-severity cases to a verified expert through Socket.io chat and Twilio SMS notification. The working prototype, illustrated through its landing page, diagnostic terminal, and expert portal, demonstrates the feasibility of the proposed architecture on a lightweight Node.js/Express/MongoDB stack. The principal open work is empirical: the accuracy, latency, and user-outcome benefits of the platform have not yet been benchmarked against labeled data or a controlled user study, and Section 8.1 and Section 12 outline the evaluation and extension work required before the system can be considered validated for production deployment. Subject to that validation, Sentinel AI offers a practical template for closing the gap between automated problem detection and human-expert resolution across both hardware and software domains.

REFERENCES

[1] M. I. Sheikh, "InfraGPT smart infrastructure: An end-to-end VLM-based framework for detecting and managing urban defects," Oct. 2025.
[2] I. Nanthini, K. Manivannan, T. Vishnu, S. Thirumalaivasam, M. Yugith, and R. Thanuja, "Integrating computer vision and deep learning for automated object diagnosis and repair," pp. 1–6, May 2025.
[3] R. K. Verma, G. Agarwal, N. Pandey, and R. K. Verma, "Explainable AI for automated defect detection," in Advances in Computational Intelligence and Robotics book series, Aug. 2025.
[4] A. Ayanzadeh, P. Dixit, S. Kamal, and M. Halem, "Wildfire VLM: AI-powered analysis for early wildfire detection and risk assessment using satellite imagery," Feb. 2026.
[5] K. Indrajit, S. M. C, V. Ankit, and P. V. D, "Explainable artificial intelligence (AI)-based image analytics for automatic damage detection and estimation systems," June 2025.
[6] [Placeholder — requires verification before submission] Ultralytics, "YOLOv8 documentation," 2023.
[7] [Placeholder — requires verification before submission] MongoDB Inc., "MongoDB and Mongoose ODM documentation," 2024.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)