IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

TechResolve: An AI-Assisted System for Evidence-Based IT Issue Diagnosis and Resolution Using Case-Based Reasoning and Retrieval-Augmented Generation

Deebuck A[1], Mahalakshmi P[2], Ravindar M[3], K. Akila*[4]

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[1-4]

Abstract: IT support incidents are often reported through multiple sources, including users' descriptions, screenshots of error messages, and entries from the Windows Event Log. Although these sources may contain complementary diagnostic information, they are not always analysed together. Consequently, troubleshooting often relies on technician's prior experience, their recollection of similar incidents, or searches for potential solutions online. To address this challenge, this paper presents TechResolve, an AI-assisted platform designed to integrate multiple sources of evidence into a unified diagnostic workflow. Technicians can submit a textual description of an issue along with a screenshot and a relevant Windows Event Log excerpt, where available. The system classifies the reported issue, identifies similar resolved incidents using case-based reasoning (CBR), and retrieves relevant troubleshooting information through retrieval-augmented generation (RAG). A large language model (LLM) then uses the submitted evidence and retrieved information to generate a likely root cause, recommended remediation steps, and preventive guidance. The current prototype supports authenticated user accounts, case management, evidence submission, and an initial issue-classification layer. The CBR-based retrieval and LLM-based reasoning components have been implemented and are currently undergoing integration testing. However, a comprehensive evaluation using a labelled test dataset has not yet been conducted.compares the proposed system's intended capabilities with those of ten related systems identified in the literature and proposes an evaluation protocol for future validation. The paper further describes the system architecture, its seven constituent modules, the technologies used for implementation, and the evaluation metrics planned for subsequent testing.

Keywords: IT issue diagnosis; root-cause analysis; retrieval-augmented generation; case-based reasoning; large language models; Vision AI; Windows Event Log analysis; IT service management

1. INTRODUCTION

IT support incidents are often reported through **multiple sources of information**, including **user-provided descriptions**, screenshots of error messages, and entries from the Windows Event Log. **Although these sources may contain complementary diagnostic information, they are often fragmented across different formats, making it difficult to establish a comprehensive understanding of an incident.** A user may describe a symptom imprecisely, a screenshot may show only part of an error condition, and the Windows Event Log may contain entries that help explain the underlying problem. **Technicians must therefore identify relevant evidence, interpret the available information, and determine an appropriate troubleshooting strategy.** In many support environments, this process relies on personal experience and **ad hoc online searches**, even when similar incidents have previously been resolved within the organisation.

Recent studies indicate that large language models (LLMs) can generate useful root-cause hypotheses and remediation suggestions when supplied with a clear incident summary However, **preparing a reliable incident summary from incomplete and heterogeneous evidence remains a significant challenge.** Furthermore, **diagnostic workflows that do not systematically reuse historical incident records may overlook valuable knowledge gained from previously resolved cases.** Consequently, there is a need for AI-assisted systems that can integrate evidence from multiple sources, retrieve relevant troubleshooting information, and reuse historical cases to support **more consistent and evidence-based IT issue diagnosis**.

To address these challenges, this paper presents TechResolve, an AI-assisted platform for evidence-based IT issue diagnosis and resolution. The proposed system combines case-based reasoning (CBR) with retrieval-augmented generation (RAG) to support incident analysis using both previously resolved cases and a troubleshooting knowledge

 This work is licensed under a Creative Commons Attribution 4.0 International License

base. **The system accepts a textual issue description, an optional screenshot, and relevant Windows Event Log evidence, where available.** It classifies the reported issue, retrieves similar resolved incidents through CBR, and retrieves relevant troubleshooting information through RAG. A large language model then uses the available evidence and retrieved information to suggest a likely root cause, **prioritised remediation steps, preventive recommendations, and a concise resolution summary for communication with the end user**. This integrated workflow is intended to connect evidence collection, knowledge retrieval, diagnostic reasoning, and incident management within a single platform.

The principal objectives of this work are to develop a structured processing pipeline for heterogeneous incident evidence; integrate case-based reasoning with retrieval-augmented generation so that historical resolutions and technical documentation can inform LLM-based diagnosis; generate actionable remediation and preventive guidance; and establish an evaluation framework for assessing the proposed system against a rule-based classification baseline and existing diagnosis systems. The system is designed to support not only the identification of a probable cause but also the communication and documentation of the recommended resolution. The current prototype supports authenticated user accounts, case management, evidence submission, and an initial issue-classification layer. The CBR-based retrieval and LLM-based reasoning components have been implemented and are undergoing integration testing. **A comprehensive evaluation using a labelled test dataset has not yet been completed; therefore, quantitative performance results are not reported in this paper.** The comparative analysis instead examines the intended capabilities of the proposed system in relation to existing approaches, while the evaluation protocol defines the metrics and experiments planned for subsequent validation.

The main contributions of this work are **fourfold**. First, **it presents a system architecture and working prototype that integrate textual descriptions, screenshots, and Windows Event Log evidence for general end-user IT support**, complementing existing research that focuses on cloud and microservice environments. Second, **it introduces a retrieval workflow that searches a repository of resolved cases and a separate troubleshooting knowledge base**, providing the LLM with contextual information from both sources to support its diagnostic suggestions. Third, **it proposes an integrated incident-resolution workflow that connects diagnosis with remediation, preventive guidance, end-user communication, and the storage of resolved cases for potential reuse**. Fourth, **it provides a capability-based comparison with ten related systems identified in the literature and outlines a protocol for future quantitative evaluation**.

2. LITERATURE SURVEY

This section reviews recent studies on LLM-assisted incident diagnosis, root-cause analysis, and log-based fault diagnosis. **Ten relevant studies published between 2023 and 2025 were selected for analysis**, as summarised in Table 1. The studies were chosen based on their relevance to automated diagnosis, fault localisation, and incident resolution. **Their complete bibliographic details are provided in the reference list, and any entries requiring further verification are explicitly identified.**

S.no	Work	Method	Reported Limitation
[1]	RAG for IT Incident Resolution	RAG, encoder-based classification, LLM query generation	Mainly solution recommendation; limited multimodal evidence and root-cause analysis
[2]	GPT-4 for Cloud RCA	GPT-4 with in-context learning for root-cause analysis	Cloud-focused; mainly root-cause analysis, not complete troubleshooting
[3]	LLM-Based Cloud Incident Management	LLMs with multi-stage contextual information augmentation	Cloud/SDLC focused; does not target end-user IT troubleshooting
[4]	LLM Agents for RCA	ReAct-style agent with retrieval tools and LLM reasoning	Primarily root-cause analysis; multimodal user evidence and resolution workflow are limited
[5]	LogExpert: LLM Log Resolution	Log recognizer with LLM and technical-forum knowledge	Relies mainly on anomalous logs and external forum knowledge
[6]	MonitorAssistant: LLM Monitoring	LLM-based anomaly detection, model recommendation, alarm interpretation	Focuses on monitoring and anomaly detection rather than full issue resolution

 This work is licensed under a Creative Commons Attribution 4.0 International License

S.no	Work	Method	Reported Limitation
[7]	Multi-Source Microservice Diagnosis	Multi-source data fusion and embeddings for failure diagnosis	Designed for microservices; not intended for general IT-support issues
[8]	LLM-Based Log Fault Diagnosis	Fine-tuned LLM with fault-oriented log summarisation	Primarily log-based; lacks screenshots, case reuse and complete troubleshooting
[9]	COCA: Generative Root Cause Analysis	Code-knowledge extraction and execution-path reconstruction with LLM-based RCA	Requires code knowledge; focused on distributed systems
[10]	T2MFDF: Multimodal Fault Diagnosis	LLM-enhanced multimodal diagnosis with data augmentation	Industrial domain-specific; modalities differ from IT troubleshooting

Table 1. Summary of related work in LLM-based incident and fault diagnosis.

2.1 Research gap

The studies **summarised** in Table 1 reveal several differences in scope. **Studies focusing on** cloud and microservice operations generally assume access to telemetry data or system logs and **do not explicitly address the integration of user-written descriptions and screenshots typically encountered in desktop IT support**. Log-oriented approaches place greater emphasis on log evidence, while **the present comparison identifies screenshot analysis and the systematic reuse of previously resolved cases as capabilities not explicitly established in the reported scope of these studies**. Retrieval-based systems, such as [1] and [4], **use documents or retrieved examples to support diagnosis; however, the reviewed descriptions do not establish a unified workflow that combines document-based retrieval with the systematic reuse of resolved incidents**. TechResolve is designed to integrate textual, visual, and log-based evidence with case-based reasoning (CBR), retrieval-augmented generation (RAG), preventive guidance, and a user-facing resolution summary. **These observations reflect a comparison of the capabilities reported in the literature and do not demonstrate superior performance by TechResolve.**

2.2 Capability Comparison

As TechResolve has not yet undergone evaluation using a labelled test dataset, the comparison focuses on documented system capabilities rather than measured diagnostic accuracy or effectiveness. Each of the ten studies listed in Table 1 was assessed against six features: free-text input, screenshot or visual evidence, event-log evidence, reuse of previously resolved cases through CBR, document-based knowledge retrieval through RAG, and **an end-to-end workflow extending from root-cause analysis to documented resolution**. A feature was counted only when **explicit supporting evidence was available in the corresponding study's description of its methods, data, or limitations**. This conservative assessment **provides a minimum estimate of the capabilities supported by the reviewed literature rather than a comprehensive representation of each system's functionality**. Under this assessment, the reviewed studies **were found to explicitly support between zero and two of the six assessed features**, whereas TechResolve is designed to address all six. **Accordingly, the results presented in Figure. 1 represent a comparison of documented and intended capabilities, not a quantitative evaluation of diagnostic performance.**

 This work is licensed under a Creative Commons Attribution 4.0 International License

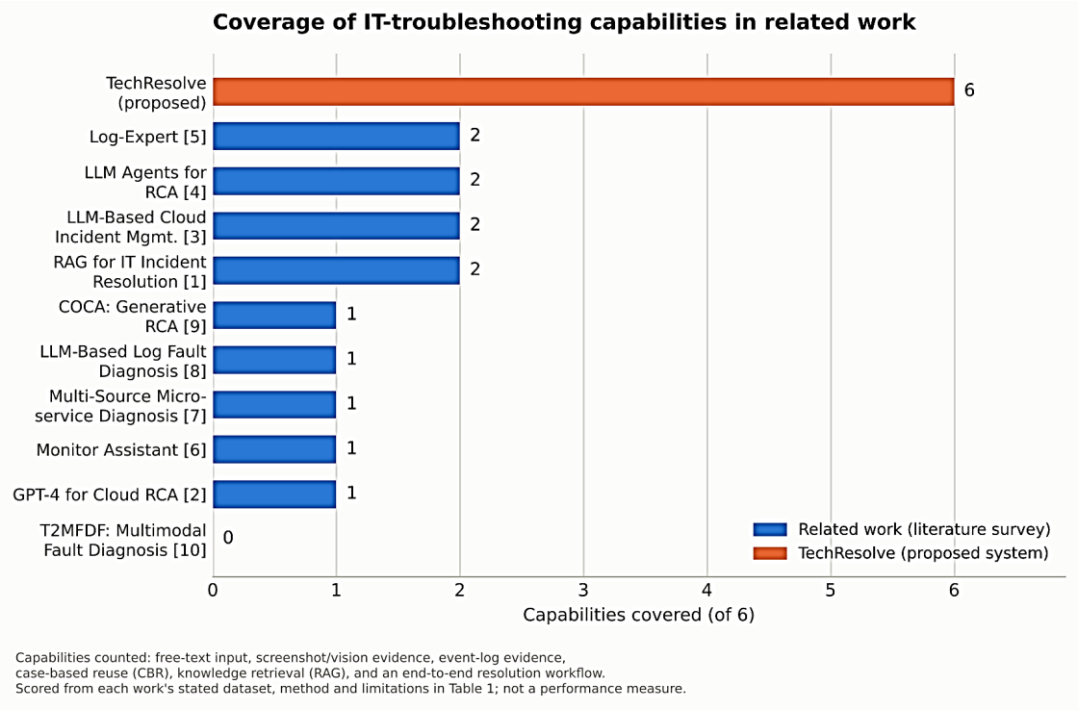


Fig. 1. Capability coverage of TechResolve and the ten related systems listed in Table 1. The comparison considers text input, screenshot evidence, log evidence, CBR, RAG, and an end-to-end resolution workflow. Scores reflect reported scope, not measured system performance.

3. WORKING MECHANISM

3.1 System Overview

A technician begins by entering a **textual description of the reported issue and, where available, attaching a screenshot and a relevant Windows Event Log excerpt**. TechResolve processes the submitted evidence, assigns an issue category, and searches for relevant resolved cases and troubleshooting documents. **Based on the available evidence and retrieved information, the system generates** a likely root cause, **prioritised troubleshooting steps**, preventive recommendations, and a summary suitable for communication with the end user. When a technician marks an incident as resolved, **the resolution is stored in the case repository for potential reuse in future incidents**. Consequently, similar incidents can **benefit from previously documented resolutions**, depending on the relevance, quality, and coverage of the stored cases.

3.2 System Architecture

Figure 2 **System Architecture**

System Architecture

The architecture of **TechResolve illustrates the architecture of TechResolve, comprising eight interconnected layers**, each performing a specific function in the IT troubleshooting process. The user interface collects issue descriptions and supporting evidence from users and presents diagnostic recommendations and resolution summaries. **The evidence-processing layer analyses the submitted inputs**: a vision model extracts relevant text and interface details from uploaded screenshots, while a Windows Event Log analyser filters and normalises log entries to identify useful diagnostic information. The issue-analysis layer **interprets the available evidence, assigns an issue category, and formulates a query for information retrieval**, preparing the input for the subsequent retrieval process.

The retrieval layer searches two knowledge sources simultaneously: **a case repository containing previously resolved incidents for case-based reasoning (CBR), and a document repository containing troubleshooting guides, manuals, and knowledge-based articles for retrieval-augmented generation (RAG)**. Both repositories use vector representations to identify information that closely matches the reported problem. The root-cause analysis layer **combines the submitted evidence with the retrieved information to generate a probable cause and an associated confidence estimate**, helping technicians understand the likely source of the issue. Based on this analysis, the solution-generation layer produces ordered **troubleshooting procedures and preventive recommendations**, guiding users through appropriate resolution steps while helping reduce the likelihood of similar issues recurring.

The communication layer converts the diagnostic results into an email-style **summary for technicians and end users, with an optional text-to-speech output where supported**, making the findings easier to understand and communicate. Finally, the storage layer maintains **user records, incident details, submitted files, log data, application data, and vector indexes** required for the system's operation. This layered design **supports the management of operational records and retrieval indexes as distinct data components**, improving the organisation, maintainability, and clarity of the overall architecture.

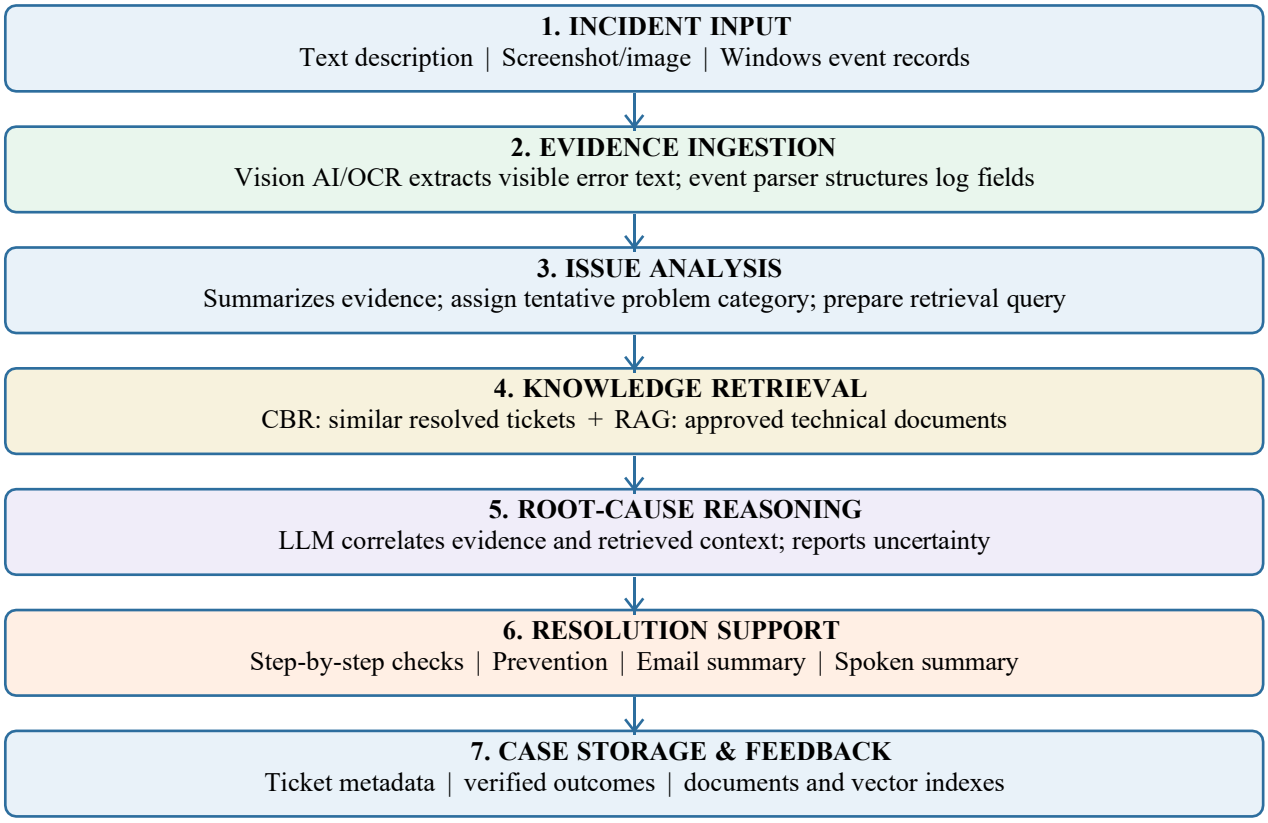


Fig. 2. TechResolve architecture and data flow across its seven processing layers

3.3 Processing Pipeline

Figure 3 **illustrates the sequence of operations involved in processing an IT support incident through the proposed TechResolve workflow**. Modules 1–3 handle **evidence submission and preliminary analysis**, including textual and image input, screenshot interpretation, Windows Event Log processing, and issue classification. Modules 4–7 **focus on information retrieval, diagnostic reasoning, and response generation**. The CBR and RAG components operate in parallel, retrieving relevant information from previously resolved cases and the troubleshooting knowledge base, respectively. **The retrieved information is then provided to the LLM reasoning stage alongside the available incident evidence to support root-cause analysis.** The resulting output comprises a likely root cause, **prioritised troubleshooting steps, preventive recommendations, and a written summary or optional spoken response**. Once the incident is resolved, **the case record is updated to preserve the resolution popotential reuse in future incidents**.

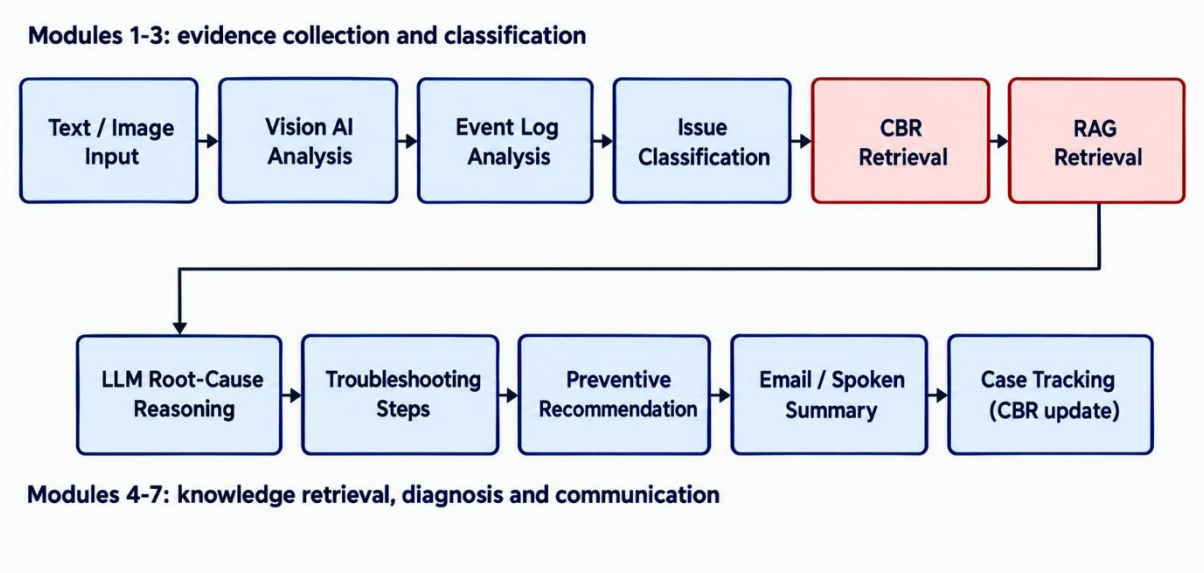


Fig. 3. Flow from initial issue submission to resolution tracking and case retention.

3.4 Methodology

TechResolve **adopts a modular methodology for evidence-based IT issue diagnosis and resolution**. It integrates user-reported information, available technical evidence, artificial intelligence (AI), and knowledge derived from previously resolved incidents to support IT professionals during troubleshooting. **The proposed workflow comprises six stages, from evidence collection and issue classification to diagnostic reasoning, communication, and knowledge reuse.**

3.4.1. Issue Reporting and Evidence Collection

The process begins when an end user reports a technical problem through the application. The user can provide a textual description of the issue and, where available, attach a screenshot and relevant Windows Event Log information. **These inputs provide complementary contextual and technical evidence for subsequent analysis.**

3.4.2. Issue Classification Using a Master Taxonomy

The collected information is analysed to identify the relevant technical issue. A centralised master taxonomy organises reported incidents into predefined categories, including network, operating system, software, and driver-related problems. **The AI-based classification component identifies a suitable issue category, which is then validated against the predefined taxonomy.** This approach is intended to promote consistent classification across users and incidents.

3.4.3. Knowledge Retrieval Using CBR and RAG

Following classification, the system retrieves information relevant to the reported issue. Case-based reasoning (CBR) searches for previously resolved incidents with similar characteristics, whereas retrieval-augmented generation (RAG) retrieves relevant technical information from the available troubleshooting knowledge base. **The retrieved cases and documents provide complementary sources of contextual information for the subsequent diagnostic stage.**

3.4.4. Root-Cause Analysis and Resolution Guidance

The available incident evidence and retrieved information are provided to the reasoning component to generate a likely root cause. The system then produces an explanation of the issue, an ordered sequence of troubleshooting steps, and preventive recommendations intended to reduce the likelihood of recurrence. **When the available evidence is insufficient to support a reliable diagnosis, the suggested cause should be treated as provisional and verified by an IT professional before corrective action is taken.**

3.4.5. Communication and Case Resolution

To facilitate communication between technicians and end users, the system prepares an email-style summary describing the reported issue, the proposed diagnosis, and the recommended resolution. **An optional spoken summary may also be provided where text-to-speech functionality is available.** The IT professional reviews the recommendations, performs the necessary checks, and updates the case status following verification of the resolution.

3.4.6. Knowledge Reuse and Continuous Improvement

When an incident is marked as resolved, its problem description, identified cause, and verified resolution can be stored in the case repository for potential reuse. Subsequent incidents with similar characteristics can retrieve these records

through the CBR component. **This process supports the accumulation and reuse of organisational troubleshooting knowledge, provided that stored cases are sufficiently accurate, relevant, and appropriately maintained.**

Overall, the proposed methodology integrates evidence collection, issue classification, knowledge retrieval, root-cause analysis, resolution guidance, and case reuse into a unified workflow. **TechResolve is intended to assist IT professionals by structuring troubleshooting activities and facilitating access to relevant historical knowledge, while retaining human oversight of diagnosis verification and final resolution.**

3.5 Module Description

The TechResolve prototype is organised into seven modules, each responsible for a specific function within the overall workflow.

3.5.1. User Authentication and Case Management

Technicians authenticate using a username and password. Passwords are **stored as cryptographic hashes**, and JSON Web Tokens (JWTs) are used for session management. Each incident is associated with the technician who created it, and its status can be updated through the states *Open*, *In Progress*, and *Resolved*.

3.5.2. Issue Input and Evidence Collection

The module enables technicians to create an incident by entering a title and textual description. **Supporting evidence, including screenshots and raw Windows Event Log text, can be attached where available.** These inputs provide the information required for subsequent classification and diagnosis.

3.5.3. AI-Based Issue Analysis and Classification

This module analyses the submitted textual description and available screenshot and log evidence to determine the issue type and category, such as *Network*, *Operating System*, or *Security/Identity*. It also generates a severity estimate and a concise summary of the available evidence **to support subsequent information retrieval and diagnostic reasoning**.

3.5.4. CBR and RAG Knowledge Retrieval

The evidence summary is converted into a vector embedding and used to search two separate vector stores. The CBR repository contains previously resolved incidents, whereas the RAG repository contains troubleshooting guides and relevant knowledge-base documents. **Retrieved results are passed to the reasoning component when they satisfy the configured similarity threshold. If no sufficiently relevant results are identified, the system should avoid relying on weak matches and instead base its response on the available evidence or indicate the need for further investigation.**

3.5.5. LLM-Based Root-Cause Analysis and Troubleshooting

The large language model (LLM) analyses the current incident evidence alongside the retrieved cases and documents to generate a likely root cause and recommended troubleshooting steps. **The reasoning instructions are designed to constrain the response to the available evidence, avoid unsupported claims about log entries or error codes, and indicate when additional information or verification is required.** The generated diagnosis is intended to support the technician's investigation rather than replace professional verification.

3.5.6. Preventive Recommendations and Communication

This module generates recommendations intended to reduce the likelihood of recurring incidents. A template-based component converts the stored diagnostic information into an email-style summary and, where supported, a spoken version. **Generating the summary from the recorded case information is intended to maintain consistency between the documented diagnosis and the information communicated to the end user.**

3.5.7. Integration, Testing, and Evaluation

The modules are integrated through a common application programming interface (API) and web-based dashboard. **Integration testing examines the interaction between the modules and the flow of information across the workflow.** Section 5 presents the proposed evaluation metrics and experimental protocol for assessing the integrated system. **A comprehensive quantitative evaluation using a labelled test dataset remains future work.**

4. IMPLEMENTATION

Layer	Technology
Frontend	React.js (Vite), React Router, Axios, custom CSS
Backend	Python 3.10+, FastAPI, SQLAlchemy, Alembic migrations
Authentication	passlib (bcrypt password hashing), python-jose (JWT sessions)
Relational database	PostgreSQL — user, case and application data
Vector database	Chroma — case-store and knowledge-store embeddings for CBR and RAG
Log collection	pywin32 / win32evtlog for Windows Event Log access
LLM and embedding services	Google Gemini API (text, vision and embedding endpoints)
Communication	SMTP for email summaries; gTTS for optional spoken summaries

Table 2. Technology stack.

5. PERFORMANCE METRICS AND COMPARATIVE EVALUATION

This section describes how TechResolve is intended to be evaluated after integration testing is complete. No numerical performance results are reported because the integrated pipeline has not yet been tested against a labelled dataset. Presenting placeholder or illustrative values as measured results would be misleading. Tables 4 and 5 therefore define the metrics, baseline, and experimental procedure for the next phase, allowing later results to be interpreted against a stated method.

5.1 Evaluation Metrics

Metric	Definition	Measurement method
Diagnosis accuracy	Share of cases for which the system's stated root cause matches an expert-assigned ground-truth cause	Blind comparison of system output against technician-labelled root cause on a held-out test set
Troubleshooting success rate	Share of cases resolved by following the generated steps without further escalation	Technician follow-up after applying the suggested steps
Response time	Wall-clock time from case submission to the diagnosis being returned	Server-side timing instrumentation around the Module 3–6 pipeline
Time saved vs. manual process	Reduction in technician time compared to unaided investigation of the same case	Paired comparison on a matched sample of cases, with and without TechResolve
User-rated usefulness	Technician rating of the clarity and usefulness of the generated guidance	1–5 Likert-scale rating collected after each case is closed

Table 4. Planned evaluation metrics and how each will be measured.

5.2 Comparative Baseline

Figure 3 compares the stated scope of TechResolve with that of published systems; it is not a controlled performance comparison because the systems were not tested on a common dataset. For a quantitative comparison within this project, the existing keyword-matching classifier will serve as the rule-based baseline. It will process the same cases as the Gemini-based classification and the full CBR+RAG+LLM workflow. Using the same test cases and metrics for both approaches will help show whether retrieval and LLM reasoning add value beyond structured case records and basic keyword classification.

5.3 Planned Experimental Protocol

- Prepare a test set containing at least 50 IT support incidents across the categories described in Section 3.4, including network, operating system, security/identity, hardware, software, and performance issues. Include text-only cases as well as cases with screenshots, event logs, or both.

- Ask a technician who has not seen the system's output to label each case independently with its expected root cause and category. This is intended to reduce the risk that the system's suggestion influences the reference label.
- Run every case through both the rule-based baseline and the full TechResolve pipeline, then assess the outputs using the metrics in Table 4.

Report the dataset size, category distribution, and confidence intervals alongside aggregate results, so readers can judge how much uncertainty remains.

6.DISCUSSION AND LIMITATIONS

Several limitations remain in the current implementation of TechResolve. First, the system **depends on a cloud-based large language model (LLM) API for classification, reasoning, and embedding generation**, introducing network dependency, response latency, usage costs, and potential variability in generated outputs. Second, the current log-processing workflow is designed for Windows Event Logs. **Extending the system to macOS, Linux, and network-device logs would require platform-specific parsing mechanisms and further validation.**

The effectiveness of the retrieval components also depends on the **availability, relevance, and quality of the stored incident cases and troubleshooting documents**. In newly deployed environments, a limited collection of resolved cases may constrain case-based retrieval, although diagnosis can still draw on the available incident evidence and documentation. Furthermore, **the classification and reasoning workflow using the Gemini-based model has not yet been evaluated against a labelled test dataset**. Consequently, the present work establishes the system architecture and describes its intended capabilities but does not demonstrate quantitative improvements in diagnostic accuracy or resolution effectiveness. **The evaluation protocol presented in Section 5 provides a framework for assessing these aspects in future testing.**

7. CONCLUSION AND FUTURE WORK

This paper presented TechResolve, **an AI-assisted system designed to support evidence-based diagnosis and resolution of end-user IT issues**. The proposed workflow integrates user-provided issue descriptions, screenshots, and Windows Event Log entries with case-based reasoning (CBR) and retrieval-augmented generation (RAG). The system is designed to generate a probable root cause, **ordered troubleshooting recommendations, preventive guidance, and a concise communication summary**.

The current prototype supports authenticated case management and evidence submission, while the retrieval and LLM-based reasoning components have been implemented and remain under integration testing. A comparison with ten related systems indicates that **integrating multiple evidence sources, historical case reuse, document-based retrieval, and end-to-end troubleshooting represents a distinguishing design objective of TechResolve**. However, this comparison reflects the documented scope and intended functionality of the systems considered and **does not establish superior diagnostic performance or operational effectiveness**.

Future work will focus on evaluating the system using a labelled dataset and appropriate quantitative metrics, extending log-processing support to additional operating systems and network devices, and investigating **how retrieval relevance and recommendation quality vary as the case repository expands**. Further testing will also be needed to assess the consistency, reliability, and practical usefulness of the generated troubleshooting guidance.

ACKNOWLEDGMENT

The authors gratefully acknowledge **Prof. Dr. K. Akila for her guidance and support throughout the design and development of this project**. The authors also thank the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, for **providing the facilities and resources that supported this work**.

REFERENCES

[1]. T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, “Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models,” in Proc. IEEE/ACM International Conference on Software Engineering (ICSE), 2023, pp. 1737–1749. DOI: 10.1109/ICSE48619.2023.00149.

[2]. X. Zhang et al., “Automated Root Causing of Cloud Incidents Using In-Context Learning with GPT-4,” in Proc. ACM International Conference on Foundations of Software Engineering (FSE), Industry Track, 2024. arXiv:2401.13810.

 This work is licensed under a Creative Commons Attribution 4.0 International License

[3]. D. Goel et al., "X-lifecycle Learning for Cloud Incident Management Using LLMs," in Proc. ACM International Conference on Foundations of Software Engineering (FSE), Industry Track, 2024. arXiv:2404.03662.

[4]. D. Roy et al., "Exploring LLM-based Agents for Root Cause Analysis," in Proc. ACM International Conference on Foundations of Software Engineering (FSE), Industry Track, 2024. arXiv:2403.04123.

[5]. J. Wang et al., "LogExpert: Log-Based Recommended Resolutions Generation Using Large Language Model," in Proc. IEEE/ACM International Conference on Software Engineering (ICSE), New Ideas and Emerging Results Track, 2024.

[6]. A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry Practices and Event Logging: Assessment of a Critical Software Development Process," in Proc. IEEE/ACM International Conference on Software Engineering (ICSE), 2015, pp. 169–178. DOI: 10.1109/ICSE.2015.145.

[7]. P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," in Proc. IEEE International Conference on Web Services (ICWS), 2017, pp. 33–40.

[8]. S. Huang et al., "HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log," IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2064–2076, 2020.

[9]. W. Meng et al., "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs," in Proc. International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 4739–4745.

[10]. X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: Robust and Unified Deep Learning Based Log Anomaly Detection for Diverse Faults," in Proc. IEEE International Symposium on Software Reliability Engineering (ISSRE), 2020, pp. 92–103. DOI: 10.1109/ISSRE5003.2020.00018.

[11]. J. Qi et al., "LogGPT: Exploring ChatGPT for Log-Based Anomaly Detection," in Proc. IEEE International Conference on High Performance Computing and Communications, Data Science and Systems, Smart City and Dependability in Sensor, Cloud and Big Data Systems and Application, 2023, pp. 273–280. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys60770.2023.00045.

[12]. Z. Ma, A. R. Chen, D. J. Kim, T.-H. P. Chen, and S. Wang, "LLMParser: An Exploratory Study on Using Large Language Models for Log Parsing," in Proc. IEEE/ACM International Conference on Software Engineering (ICSE), 2024, pp. 1209–1221. DOI: 10.1145/3597503.3639150.

[13]. J. Liu et al., "Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression," in Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 863–873. DOI: 10.1109/ASE.2019.00085.

[14]. Y. Sun et al., "Accurate and Interpretable Log-Based Fault Diagnosis Using Large Language Models," IEEE Transactions on Services Computing, vol. 18, no. 5, pp. 2602–2615, 2025.

[15]. S. Zhang et al., "Fault Diagnosis for Test Alarms in Microservices Through Multi-Source Data," in Proc. ACM International Conference on Foundations of Software Engineering (FSE), Industry Track, 2024.

[16]. H. Jiang, X. Li, Z. Yang, and J. Xuan, "What Causes My Test Alarm? Automatic Cause Analysis for Test Alarms in System and Integration Testing," in Proc. IEEE/ACM International Conference on Software Engineering (ICSE), 2017, pp. 712–723.

[17]. J. Kim, V. Savchenko, K. Shin, K. Sorokin, H. Jeon, G. Pankratenko, S. Markov, and C.-J. Kim,"Automatic Abnormal Log Detection by Analyzing Log History for Providing Debugging Insight," in Proc. ACM/IEEE International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP),2020, pp.71–80.

[18]. M. Kim, R. Sumbaly, and S. Shah, "Root Cause Detection in a Service-Oriented Architecture," ACM SIGMETRICS Performance Evaluation Review, vol. 41, no. 1, pp. 93–104, 2013.

[19]. D. Liu et al., "MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems," in Proc. IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2021, pp. 338–347.

[20]. Y. Sui et al., "LogKG: Log Failure Diagnosis Through Knowledge Graph," IEEE Transactions on Services Computing, vol. 16, no. 5, pp. 3493–3507, 2023. DOI: 10.1109/TSC.2023.3293890.

[21]. S. Zhang et al., "No More Data Silos: Unified Microservice Failure Diagnosis With Temporal Knowledge Graph," IEEE Transactions on Services Computing, vol. 17, no. 6, pp. 4013–4026, 2024. DOI: 10.1109/TSC.2024.3489444.

[22]. P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.

[23]. A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," AI Communications, vol. 7, no. 1, pp. 39–59, 1994.

[24]. Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge Graph Embedding: A Survey of Approaches and Applications," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724–2743, 2017.

[25]. M. Ma et al., "AutoMap: Diagnose Your Microservice-Based Web Applications Automatically," in Proc. The Web Conference (WWW), 2020, pp. 246–258.