Cascading automata to improve efficiency of large language models agents with GraphRAG for error analysis

Hrishikesh K. Haritas[1], Vineet H. Sadarangani[1], Ganeshayya Ishwarayya Shidaganti[2], Darshan Bankapure[1], Rahul K. Vishal[1], Shreya Vijayasimha[3]
[1]Department of Artificial Intelligence and Machine Learning, Ramaiah Institute of Technology, Bengaluru, India
[2]Department of Computer Science and Engineering, Ramaiah Institute of Technology, Bengaluru, India
[3]Department of Electronics and Communication Engineering, Ramaiah Institute of Technology, Bengaluru, India

Article Info

Article history:

Received Apr 30, 2025
Revised Nov 12, 2025
Accepted Dec 15, 2025

Keywords:

Agentic large language models
AI enhanced automation
Energy efficient LLM agents
Error correction and control
Hyper automation
Low latency agents

ABSTRACT

Robotic process automation (RPA) has been deployed in a plethora of industries, including the banking and insurance sectors. However, the key challenge of handling unexpected situations manifests either as an inadequacy of programming (since all situations cannot possibly be foreseen) or incongruous inputs. In parallel, deep learning models, including large language models (LLMs) and visual language models (VLMs), have shown human-like cognitive capabilities in real-world tasks, germinating the field of agentic LLMs. However, their computational expense, slow inference times, and massive energy consumption impede large-scale usage. We propose a framework that combines the two approaches to enable expedient invocation of LLMs for handling exceptions and supervising RPA bots. It aims to minimize the need for human supervision by "meta" automation, while also reducing energy usage and processing time. The automation workflow is presented as a graph, and our pipeline uses the GraphRAG framework to analyze and fix errors. We demonstrate the potential of our pipeline through two real-world examples in the banking and insurance sectors, provide our GitHub repository for reproducibility, and conclude with future research directions.

Corresponding Author:

Ganeshayya Ishwarayya Shidaganti
Department of Computer Science and Engineering, Ramaiah Institute of Technology
Bengaluru, Karnataka, India
Email: ganeshayyashidaganti@msrit.edu

1. INTRODUCTION

Robotic process automation (RPA) employs software agents, or "bots", to simulate user activities and interact with software systems, thus reducing the workload of human employees. RPA has already been widely adopted in practice, with solution technologies offered by numerous vendors. Organizations that have effectively implemented RPA and streamlined their business processes have seen positive impacts on their strategic objectives, employee productivity, and customer service [1]. RPA is especially attractive to industries that have traditionally been early adopters of new technologies, particularly process-aware information systems (e.g. banking, insurance) [2], [3]. Examples from these industries are used to demonstrate the proposed approach. Syed *et al.* [4] posit the Seamless Handling of Exceptions as one of the main technological research directions of RPA. They state that the cause of exceptions may range from a change to a user interface (e.g., different screen resolution or layout), a change to system interaction (e.g. the use of a different system required in handling an atypical scenario), to a change in business rules (e.g. special considerations regarding certain types of customer). As a result, bots may stop working or progress to an incorrect path, leading to the need for human intervention [5]–[7]. This necessity leads to increased operational costs, reduced speed, and hinders the ubiquity of RPA while also increasing the hesitancy in adoption.

Agentic AI, and agentic large language models (LLMs) in particular, have gained immense popularity for their near-human abilities in performing real world tasks, including interacting with UI. Leveraging robust zero-shot learning capabilities, these models—exemplified by IBM Granite and Microsoft's Autogen—effectively generalize from minimal input to execute complex operations across diverse domains [8]–[10]. Built on state-of-the-art transformer architectures and unsupervised learning paradigms, they enable autonomous decision-making and streamlined user interface interactions. However, these performance gains come at the expense of significant computational overhead, leading to high energy consumption [11], [12], increased latency [13], and elevated operational costs. Ongoing research continues to focus on mitigating these drawbacks while further refining their functional capabilities.

This paper proposes the use of LLMs through the GraphRAG [14] framework as an intermediary supervisor. This work strides towards hyper automation [15] and unassisted RPA. Previously, Jain *et al.* [16] proposed the use of LLMs for RPA for form filling.

GraphRAG is a novel framework built to present large amounts of contextual information to an LLM. In the GraphRAG global search process [14], [16], the LLM confers a knowledge graph to produce an output to any user query. We propose representing the RPA workflow process, embedded with operational information, as a graph to be queried by the LLM in a process similar to the aforementioned global query, enabling a more detailed analysis of the bot's workflow.

We provide a generalized pipeline with heuristic-driven expedient LLM calls applicable across domains and tasks. The idea of expediently calling LLMs—i.e., only when an exception occurs and the RPA is progressing along an incorrect path—was inspired by other software engineering research solutions including [17]. The expedient calling of LLMs offers various benefits including minimizing computational costs and hence, energy usage, with the environmental benefits being significantly pronounced at scale; increased resilience to changes in the underlying process as needed; and reduction in processing time compared to replacing the RPA bot with an LLM.

2. LITERATURE SURVEY

RPA has evolved rapidly over the past decade, becoming a cornerstone technology in industries such as banking, insurance, and manufacturing [1]–[3]. Early studies primarily focused on the efficiency gains and cost reductions achievable through RPA. However, as practical deployments expanded, researchers began to highlight a critical shortcoming: traditional RPA systems struggle with exception handling when confronted with dynamic and unforeseen scenarios [4]–[6].

A significant portion of the literature underscores the challenges posed by exceptions—ranging from unexpected user interface changes to variations in business logic—that can halt automated workflows and necessitate costly human intervention. These challenges have spurred efforts to integrate intelligent supervisory mechanisms into RPA systems. Notably, recent research has proposed leveraging advanced natural language processing techniques to bridge this gap. Jain *et al.* [16] and Chen *et al.* [17] have demonstrated that LLMs can be effectively employed to both detect and resolve exceptions, thereby minimizing the need for human oversight. Complementing this approach, the GraphRAG framework introduced by Edge *et al.* [14] employs graph-based representations of workflows to enhance the contextual understanding of LLMs, enabling more precise error localization and resolution.

While LLMs offer significant advantages in terms of cognitive capabilities, their integration into RPA workflows raises concerns related to computational overhead and energy efficiency [18], [19]. Consequently, the literature emphasizes the importance of developing strategies for the expedient invocation of LLMs—ensuring that these models are activated only when necessary—to balance performance with resource utilization. Additionally, the concept of hyper automation, as discussed in industry reports [15], advocates for the unification of RPA with advanced AI techniques. This integrated approach not only enhances the resilience of automation workflows but also addresses emerging challenges such as LLM hallucination and the need for robust guardrails to maintain output reliability. Figure 1 shows a comparison between various models and Table 1 showcases popular LLMs and relevant parameters for their use as agents.

However, the newly emergent field of hallucination [20], [21] in LLMs which are pronounced in GraphRAG [22] even with the use of SOTA models like GPT-5 [23]. Beyond exception handling, recent advances in agentic AI have expanded the capabilities of RPA systems. Agentic LLMs, such as those discussed by McIntosh *et al.* [24] and Brohi *et al.* [25], emphasize autonomous decision-making, dynamic adaptation, and multi-step reasoning—critical aspects for ensuring robust and self-healing automation. Additionally, the OSWorld benchmark introduced by Xie *et al.* [26] evaluates LLM agents in real-world task execution, highlighting key challenges in ensuring reliability and efficiency. These developments align with the broader trend of multi-agent collaboration, where LLM-powered agents operate in synergy to handle exceptions, optimize workflows, and improve overall task completion rates [27], [28].

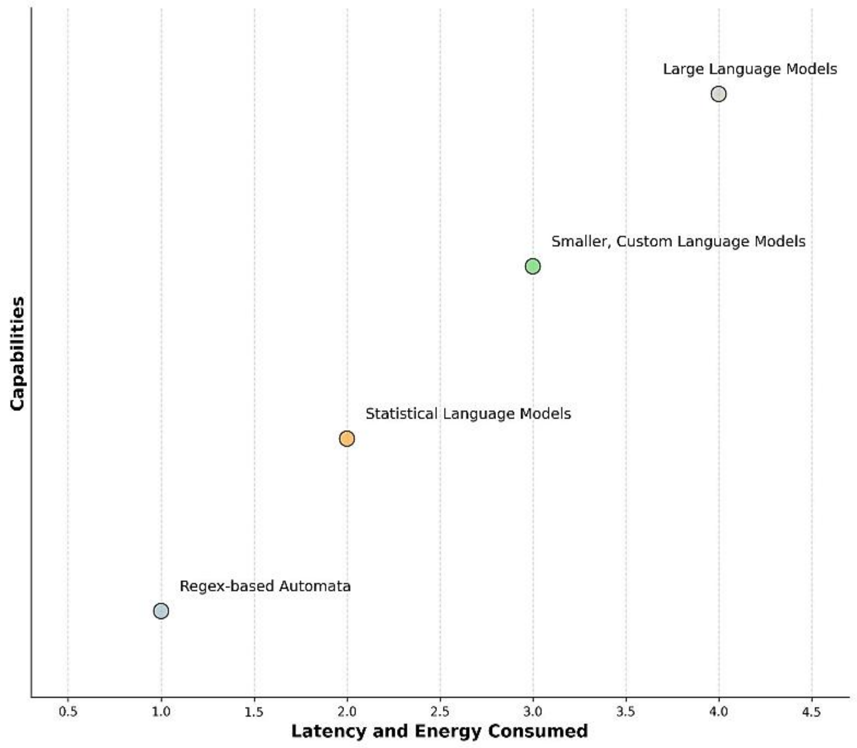


Figure 1. Spectrum of automata proposed with respect to their associated language comprehension system

Table 1. Comparing the popular 0-shot LLMs for agentic and functional AI

Model	Context length	Strengths for agentic applications	Hallucination rate [31]	Cost of input token (1K tokens)	Cost of output token (1K tokens)
Google Gemini	32,000	Complex reasoning tasks	7.7	$0.00025	$0.0005
Anthropic Claude 3.5 Sonnet	100,000	General, knowledge based and decompositional reasoning	4.6	$0.003	$0.009
OpenAI GPT-4 Turbo	128,000	Multi-modal capabilities	1.7	$0.03	$0.06
Meta Llama 3 (70B)	128,000	Instruction flow and medical reasoning	4.1	$0.002	$0.004
Mistral 7B	32,000	Knowledge based, general intelligence and physical knowledge	7.5	$0.0003	$0.0003
DeepSeek-V3	32,000	Mathematical reasoning and competitive programming	3.9	$0.0007	$0.0011

A particularly relevant paradigm in this context is zero-shot learning (ZSL), which enables LLMs to generalize new automation scenarios without task-specific retraining [29]. Brown *et al*. [8] demonstrated how large-scale pre-trained models can extrapolate patterns from limited context, significantly improving exception handling in dynamic environments. Furthermore, Wang *et al*. [30] proposed EDCEW-LLM, a large language-based approach for effective error detection and correction, showing its potential in enhancing GraphRAG-based exception-handling mechanisms.

Collectively, these studies illustrate a clear trajectory toward merging traditional RPA systems with modern AI methodologies. Despite the progress made, a comprehensive framework that seamlessly integrates the strengths of both approaches while mitigating their individual limitations remains an open research challenge. The work presented in this paper aims to fill this gap by introducing a GraphRAG-based exception handling mechanism that expedites LLM invocation, thereby improving both the robustness and efficiency of RPA workflows.

3. METHOD

Our work finds the balance between the temporal and energy efficiency offered by RPA and the awareness and knowledge that modern large-scale deep learning models such as LLMs offer. We achieve this by calling upon LLMs/visual language models (VLMs) expediently. Here, we note that the LLMs proposed are not specially trained/fine-tuned for the task of exception handling. Instead, we experiment with off-the-shelf models and prompt them appropriately to specify their position in the pipeline and regulate the format of their output. This assumption is substantiated by LLMs being widely used as zero-shot agents [8].

Small language models (SLMs) can serve as more efficient supervisors or can be cascaded with LLMs when trained appropriately. Obstructions to the RPA workflow have been broadly categorized into i) syntactic exceptions by inadequately programmed bots (since exceptions are unpredictable and often unprecedented), ii) syntactic exceptions from unexpected and incongruous inputs provided, and iii) incorrect outputs provided as a result of ambiguity in input formats. In the 1st and 2nd cases, LLMs are called upon if and only if the workflow throws an error, while for case 3, we use a gating mechanism to determine whether the situation warrants an LLM call.

Our workflow provides an advantage in energy and temporal efficiency, since serving LLMs has been a topic of active research [18] and inference times are incomparably worse compared to RPA (RPA bots are a hard computing task). Additionally, LLMs consume gargantuan amounts of energy, making their usage for RPA workflows at large scales environmentally irresponsible [19].

However, LLMs present human-like intelligence and awareness for specific cases (like the ones dealt with in RPA domains). LLMs prove exceptional at resolving real-world exceptions encountered by RPA as demonstrated in our results. Our workflow provides a reduction in LLM calls and a proportionate reduction in energy usage and inference times. The reduction in usage is inversely dependent on the complexity of the bot's workflow amongst other factors.

RPA workflow can be represented as a feature-rich directed graph. Many RPA design software, such as UiPath, offer built-in tools to convert the workflow to a JSON file. As a part of the pre-processing, the JSON file is converted into a graph which serves as the input to GraphRAG. GraphRAG enables the handling of complex workflows and allows for structured analysis when an error occurs by constructing a tree of information upon the graph of the workflow.

Figure 2 depicts the proposed pipeline for exception handling. When an RPA bot encounters an error, an LLM is called upon prior to human intervention. First, the intermediate outputs of the various blocks of the workflow (such as conditional statements and iterables) are imputed into the previously prepared graph of the workflow. Then, we prepare a prompt by combining: the error message + the block during whose execution the error occurred + a predefined prompt for our pipeline. The prompts were tailored to ensure precise and contextual responses rather than generic troubleshooting steps. Below are the prompts used for the respective use cases.

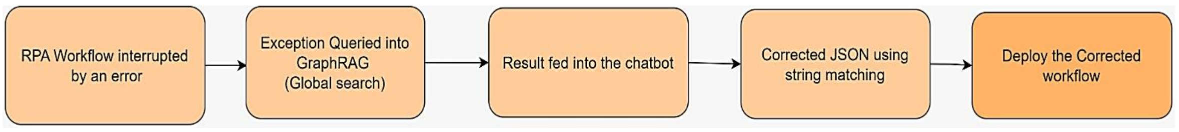

RPA Workflow interrupted by an error
Exception Queried into GraphRAG (Global search)
Result fed into the chatbot
Corrected JSON using string matching
Deploy the Corrected workflow

Figure 2. Exception handling pipeline

3.1. Enhancing document validation in KYC

The prompt in this case is, "You are a robot supervisor tasked with fixing workflow errors during deployment. When given a query describing an error at a specific step and its input schema, provide a clear, specific solution, not a generic one. The issue is not due to poor programming but a minor discrepancy between the programmed logic and ground truth. Analyze the differences between the documents in the workflow and suggest how to adapt the current workflow to fix the issue where the system incorrectly identifies two people as different. Explain why and how the error occurred."

3.2. Document rejection analysis

The prompt in this case is, "As a supervisor, your task is to explain deployment errors based on the input I provide. Your response should be specific to the error description and input schema, avoiding generic fixes. The issue is likely due to a mismatch between the programmed logic and the actual data. In this case, the error 'Add Data Row: Object reference not set to an instance of an object' may occur because the details in the PDF are not being correctly extracted by the regex, even though the PDF contains the required information. The workflow and extracted PDF text are given for review."

These prompts were instrumental in ensuring that GraphRAG provided targeted insights into workflow errors, allowing us to refine our KYC document validation and rejection analysis processes. By leveraging GraphRAG's reasoning capabilities, we were able to bridge discrepancies between programmed logic and ground truth, improving the overall accuracy and reliability of the system.

Regex is applied in the initial data validation stage, where names and addresses in KYC verification are standardized using predefined patterns (e.g., ensuring "John S" and "John Smith" are matched. Similarly, document rejection cases in the insurance workflow use regex-based parsing to detect missing fields, incorrect formats, or blurry scans before escalating to an LLM for intelligent reasoning. Additionally, the workflow graph traversal step can incorporate finite-state automata (DFA/NFA) rules to identify known failure points deterministically. By integrating regex-based rule engines as a preprocessing layer before LLM calls, we create a multi-tier automata cascade, where deterministic automata (DFAs) handle simple errors, lightweight statistical models determine fuzzy matches, and LLMs serve as the final fallback for ambiguous exceptions. This would improve efficiency by ensuring LLMs are invoked only when truly necessary, optimizing both energy consumption and processing time.

The prompt is fed into the global search system of GraphRAG [14] designed to retrieve relevant information and context from the LLM's intrinsic knowledge of the domain and the state of the process itself. The output describes the location and nature of the error and offers fixes. Upon a failure to locate/ resolve the error, the workflow signals the need for human intervention.

The output of GraphRAG (i.e., error resolution information) combined with the JSON file of the workflow is fed into an LLM (not via the GraphRAG framework). Once the LLM has processed the input, it generates the corrected section of the JSON file. Then, utilizing traditional string-matching techniques, the corrected sections are placed into the JSON file. The LLM is asked to output only the corrected sections of the framework in the interest of reducing the number of tokens generated. After the corrections are made, the updated workflow is automatically deployed, allowing the process to resume without the need for human intervention.

While the aforementioned pipeline covers the 1st and 2nd category of obstructions, the 3rd category warrants a more case specific approach. In general, when the workflow or sub process within the workflow provides a negative output (in the case of false negatives being prominent), a gating mechanism (such as a lighter machine learning model or a handwritten algorithm) can call upon the LLM if a false negative is suspected. Example #1 in section 4 details our approach to this category of obstruction.

Certain cases of the third category (as exemplified below) can be detected purely by measuring the extent of deviation from the mean of values produced intermediately in the automation pipeline. The mean and standard deviation can either be observed experimentally or configured manually based on domain knowledge.

If the intermediate outputs are strings, we may use non-data driven embedding techniques such as TF-IDF vectorization. Once the TF-IDF vectors have been created, Cosine Similarity is used to compare the vectors for different fields. Cosine Similarity calculates the angle between two vectors and assigns a score ranging from 0 to 1, with 1 indicating perfect similarity and 0 indicating no similarity. This score quantifies the similarity of two fields, even if they are not identical, by focusing on the relationship between the terms in each. The threshold is established through experimentation and analysis of various documents; we fine-tune and set the threshold to align with the workflow's practical needs. This approach enables the system to efficiently automate document verification, flagging only cases that fall below the threshold for further analysis, improving the process's speed and scalability.

3.3. Mathematical formulation

To formalize our pipeline, consider the RPA workflow as a directed graph $G = (V, E)$, where:
$V = \{v1, v2, \ldots, vn\}$ are the logical steps (decision blocks, actions)
$E \subseteq V \times V$ are the execution edges capturing flow.

When an exception occurs at node vk, we build a subgraph context

$$G_k = (V_k, E_k),$$
$$V_k = \{v_i : d(v_i, v_k) \leq \ell\},$$

where $d(\cdot,\cdot)$ is graph distance and ℓ a context window size.

Define a prompt construction function

$$P = \varphi(error, v_k, G_k)$$

where ϕ concatenates the error message, node metadata and the serialized graph context.

The GraphRAG retrieval module selects top-m relevant nodes by a scoring function:

$$s(v_i, P) = sim(E_i, P) + \lambda \cdot deg(v_i)$$

where

E_i is the embedding of node vi,

sim(·,·) is cosine similarity in embedding space,

deg (v_i) is the node-degree bias,

λ balances context vs. structural importance.

The LLM response R=LLM(P) outputs a corrected subgraph $\Delta G_k = (\Delta V_k, \Delta E_k)$.We integrate corrections via string patching into the original JSON: $G' = G \oplus \Delta G_k$.

3.4. Pseudocode

Having established mathematical formulation, the pipeline can be translated into a procedure that reflects how the system behaves during runtime. Algorithm 1 defines the full execution loop of the RPA bot. showing how the workflow advances, detects exceptions, constructs context, retrieves relevant information through GraphRAG, and applies LLM generated corrections before continuing execution. Algorithm 2 outlines the retrieval and decision logic used by the agentic LLM supervisor to score workflow nodes, assemble contextual evidence, and generate an action plan through the LLM.

Algorithm 1. CascadingAutomataWithGraphRAG(G, threshold τ)

```
Input: RPA graph G, gating threshold τ
Output: Updated graph G′
        for each execution step v_k in G do
             try
             execute(v_k)
             catch error e at node v_k:
             // Preprocess graph context
           G_k ← extractSubgraph(G, v_k, window=ℓ)
             // Build prompt
             P ← buildPrompt(e, v_k, G_k)
             // Retrieve relevant graph context
           C ← GraphRAG.retrieve(G_k, P, top_m)
             // Query LLM for correction
             ΔG_k ← LLM.resolveExceptions(P, C)
             // Patch workflow
             G ← applyPatch(G, ΔG_k)
             resume execution at v_k
        end for
        return G
```

Algorithm 2. AgenticLLMSupervisor(P, embeddings)

```
Input: prompt P, node embeddings embeddings
Output: action plan A
        // Structured retrieval
        scores ← [sim(e_i, P) + λ deg(i) for i in nodes]
        C ← selectTop(scores, m)
        // LLM planning
        A ← LLM.generatePlan(P, context=C)
        return A
```

4. IMPLEMENTATION AND RESULTS

The proposed pipeline was implemented on two real world scenarios from the banking and insurance sectors where RPA is seeing massive adoption [3]. Our work is available on GitHub for full reproducibility. For identified scenarios, automata were developed using UiPath. Microsoft's original implementation of GraphRAG with GPT-4o through OpenAPI was used.

4.1. Example 1: enhancing document validation in KYC

In industries like banking and finance, automating Know Your Customer (KYC) processes is critical for compliance and fraud prevention. However, discrepancies in customer information across documents cause exceptions and pose challenges for traditional RPA systems. Even minor variations like expansion or contraction of surnames or middle names—in our inputs while implementing a KYC verification process such as "John S" on one document and "John Smith" on another, or differences in address details like "123 Elm Street" versus "123 Elm (formerly Baker Street) Street, Opp. Caffe House" — caused the automation to misinterpret them as separate individuals. This results in manual intervention, diminishing the efficiency of the automation.

To overcome this, integrating large language models (LLMs) into RPA workflows as implemented can dramatically enhance document validation. LLMs, with their sophisticated natural language processing, can intelligently interpret such discrepancies and assess whether the documents refer to the same person. For instance, they determine that "John S" and "John Smith" likely belong to the same individual by analyzing contextual cues, such as address similarities or other corroborating information.

Incorporating LLMs eliminates the need for manual resolution by enabling automated systems to account for these subtle variations, streamlining the KYC process. This results in greater accuracy, faster processing, and reduced operational bottlenecks. Research highlights how discrepancies between official documents frequently lead to false rejections by RPA systems, underscoring the need for AI-enhanced solutions to minimize such errors and ensure compliance.

In the document verification workflow illustrated in Figure 3, it is crucial to determine whether to invoke the LLM, especially in scenarios where no errors are present and the LLM is being used just supervisor. Frequent calls to LLM for oversight can result in significant computational overhead, undermining the efficiency of the automation process. To address this challenge, a similarity score is calculated. In our example, we use a TF-IDF vectorizer to embed the candidate documents, cosine similarity between the vectors produces a score. Cases where the similarity score is above a threshold (determined experimentally) indicate a verification mismatch rather than a substantive discrepancy, the LLM is not invoked, thereby optimizing resource utilization while maintaining the integrity of the workflow.

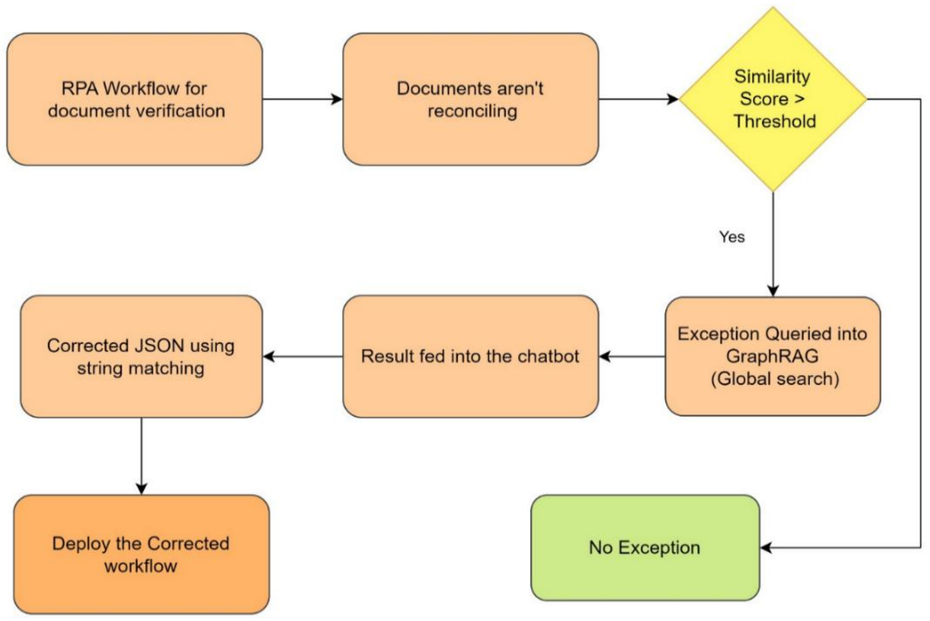

RPA Workflow for document verification
Documents aren't reconciling
Similarity Score > Threshold
Yes
No
Exception Queried into GraphRAG (Global search)
Result fed into the chatbot
Corrected JSON using string matching
Deploy the Corrected workflow
No Exception

Figure 3. Example workflow for efficient LLM supervision to combat unpredictably varying input

4.2. Example 2: document rejection analysis

In automated processes that require document uploads—such as insurance claims, loan applications, or regulatory filings—document rejection is a common hurdle. These rejections can occur for various reasons, including missing information, incorrect formats, or poor document quality. Traditional RPA systems often reject documents without providing detailed reasons, leaving users or administrators to investigate the issue manually, which diminishes the effectiveness of automation.

In our study, we generated two knowledge graphs to enhance document validation and rejection analysis within the KYC workflow. These knowledge graphs, shown in Figures 4 and 5, were constructed to capture relationships between document attributes, workflow errors, and corrective actions, enabling more precise issue resolution. By leveraging these structured representations, we improved the system's ability to detect and address discrepancies between programmed logic and real-world data.

An ideal solution is to integrate LLMs into the rejection analysis process, enabling the system to provide an intelligent, clear explanation for why a document was rejected. For instance, in the case of an insurance claim, the input is a blurry photo of a required medical bill, the LLM examines the document and identifies that the rejection is due to its unreadable or blurry nature. The LLM then generates an explanation, “Document rejected due to blurriness, rendering key information (e.g., invoice number, total amount) illegible.”

This capability allows the LLM to not only flag the issue but also offer specific guidance for correcting it, saving both the user and administrators time. It eliminates guesswork for reasons of rejection, making the process more transparent and efficient. Moreover, this approach ensures faster resubmissions and improved user experience, as users are provided with actionable feedback on how to fix the problem and meet the necessary requirements.

Research into automated document processing highlights the frequent issue of rejections due to document quality, which can significantly slow down workflows. The integration of LLMs can bridge this gap by diagnosing issues such as blurriness or missing elements and offering a human-readable explanation, allowing for faster corrections and minimizing delays in processes like insurance claims or loan approvals.

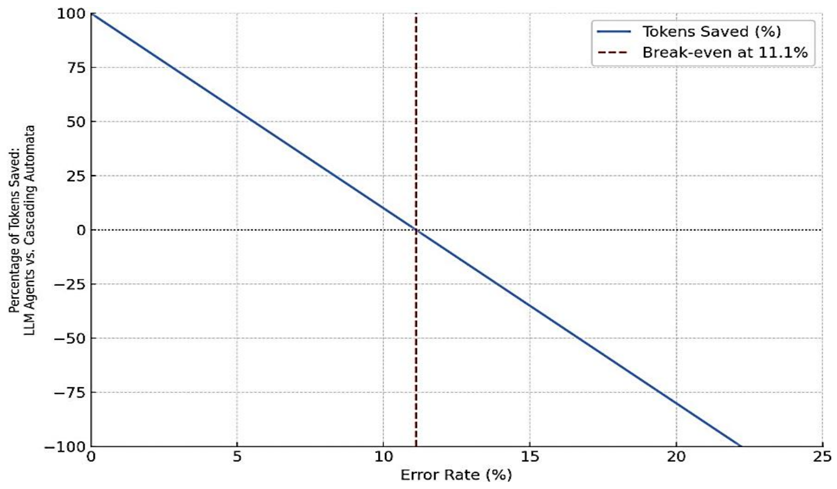
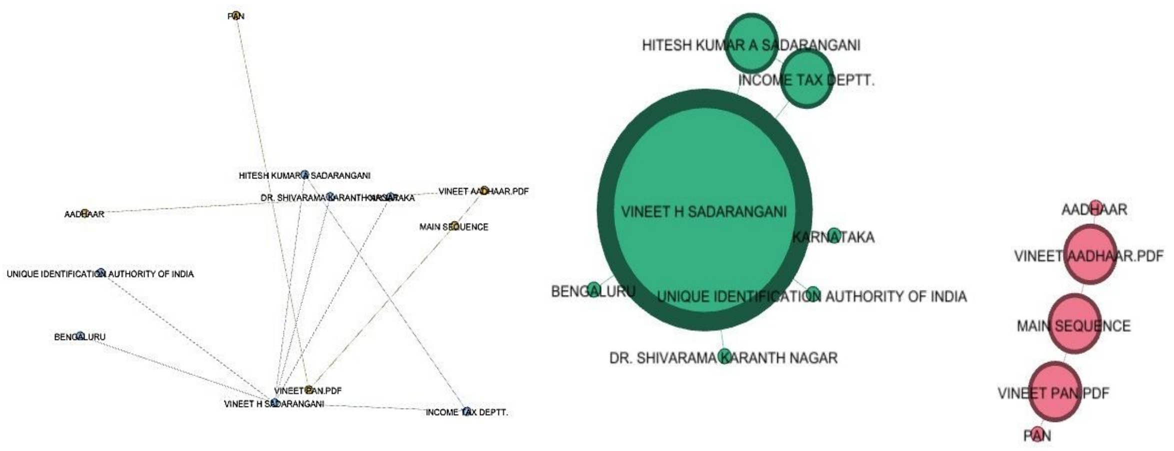


Figure 4. Token usage reduction: cascading automata vs. LLM agents for use-case #1

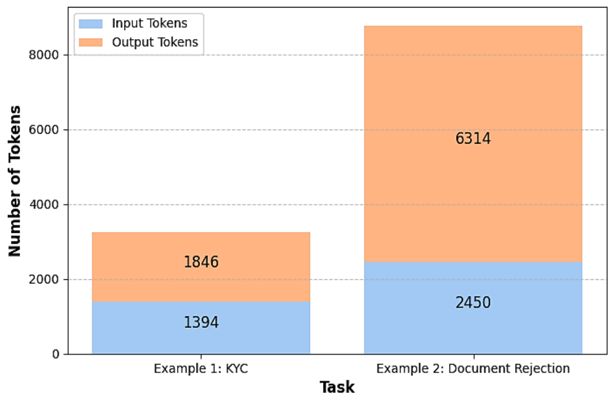


The knowledge graph generated from various entities extracted from the input

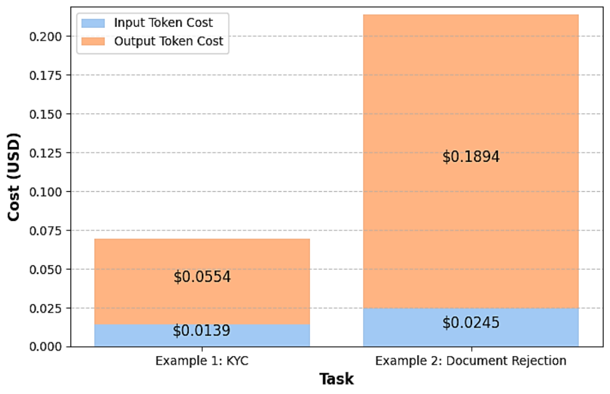
Result after applying the Leiden algorithm for clustering to the graph

Figure 5. Intermediate results of GraphRAG for document rejection analysis

Figures 5 and 6 provide an analysis of the token usage against error rates for our use cases. Here, we assume that the energy consumption and latency of rule-based automata are insignificant compared to LLMs. Hence, the reduction in token usage is representative of the computational advantages of our method. Furthermore, Figure 7 illustrates the results obtained from our GraphRAG based rejection analysis, which automatically organises root cause identification, proposed solutions, and stepwise implementation strategies derived from workflow error data. This structured reasoning output demonstrates the capability of GraphRAG to contextualize and trace logical dependencies between document errors and corrective measures, thereby enhancing explainability and supporting intelligent automation in error handling.

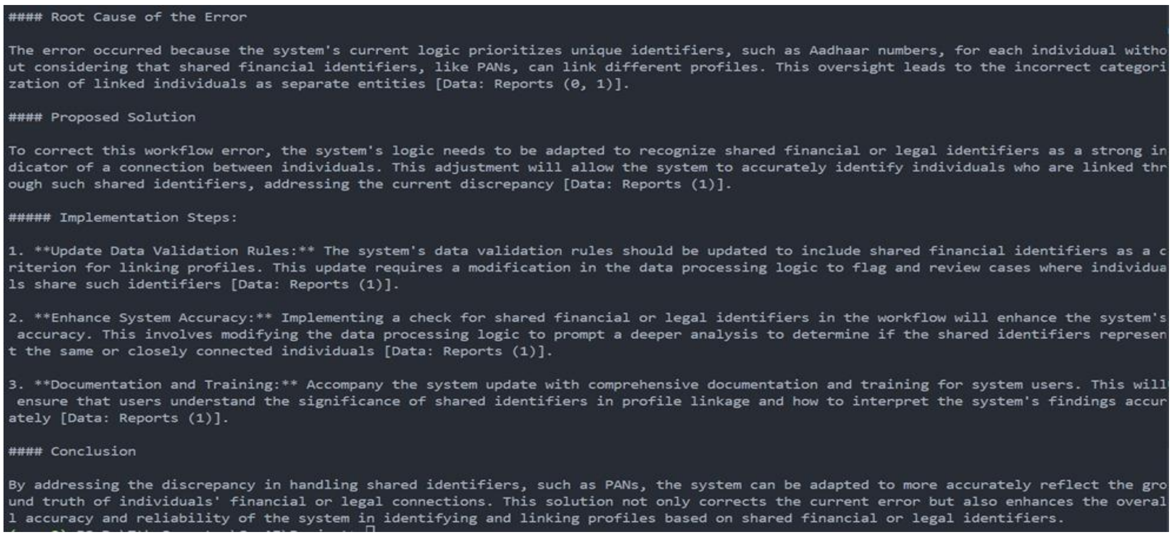


Token usage across different examples



Cost estimation based on GPT-4 Turbo pricing

Figure 6. Average input and output token usage for handling the exceptions encountered in the experimental use cases through GraphRAG



```
#### Root Cause of the Error

The error occurred because the system's current logic prioritizes unique identifiers, such as Aadhaar numbers, for each individual witho
ut considering that shared financial identifiers, like PANs, can link different profiles. This oversight leads to the incorrect categori
zation of linked individuals as separate entities [Data: Reports (0, 1)].

#### Proposed Solution

To correct this workflow error, the system's logic needs to be adapted to recognize shared financial or legal identifiers as a strong in
dicator of a connection between individuals. This adjustment will allow the system to accurately identify individuals who are linked thr
ough such shared identifiers, addressing the current discrepancy [Data: Reports (1)].

##### Implementation Steps:

1. **Update Data Validation Rules:** The system's data validation rules should be updated to include shared financial identifiers as a c
riterion for linking profiles. This update requires a modification in the data processing logic to flag and review cases where individua
ls share such identifiers [Data: Reports (1)].

2. **Enhance System Accuracy:** Implementing a check for shared financial or legal identifiers in the workflow will enhance the system's
 accuracy. This involves modifying the data processing logic to prompt a deeper analysis to determine if the shared identifiers represen
t the same or closely connected individuals [Data: Reports (1)].

3. **Documentation and Training:** Accompany the system update with comprehensive documentation and training for system users. This will
 ensure that users understand the significance of shared identifiers in profile linkage and how to interpret the system's findings accur
ately [Data: Reports (1)].

#### Conclusion

By addressing the discrepancy in handling shared identifiers, such as PANs, the system can be adapted to more accurately reflect the gro
und truth of individuals' financial or legal connections. This solution not only corrects the current error but also enhances the overal
l accuracy and reliability of the system in identifying and linking profiles based on shared financial or legal identifiers.
```

Figure 7. Results obtained by using GraphRAG

5. CONCLUSION

Our pipeline of expediently invoking LLMs for exception handling and supervision demonstrates significant potential in advancing true hyper-automation. By strategically integrating LLMs only when necessary, our approach effectively addresses one of the most persistent challenges in RPA—the handling of exceptions—without the constant reliance on computationally expensive and high-latency models. This selective invocation not only optimizes resource utilization but also ensures that automation workflows remain efficient, responsive, and scalable.

Through our two case studies, we illustrate how this workflow has the potential to redefine the automation landscape by enabling more intelligent, context-aware decision-making within RPA systems. The ability to seamlessly integrate rule-based mechanisms with LLM-powered exception handling contributes to greater system adaptability and reliability, reducing manual intervention while maintaining accuracy. This advancement could significantly enhance the adoption and ubiquity of RPA across various industries, further bridging the gap between traditional automation and AI-driven cognitive automation. Consequently, our approach serves as a foundational step toward realizing fully autonomous and resilient automation pipelines in enterprise environments.

6. FUTURE WORK

There are several works to be expected in the future. The first is preventing LLM hallucination. LLMs frequently generate inaccurate information or introduce security vulnerabilities within RPA systems. Addressing this challenge is an ongoing research area where methodologies such as reinforcement learning with human feedback (RLHF), retrieval-augmented generation (RAG), and structured prompting have demonstrated potential in mitigating hallucination. Additionally, advancements in adversarial testing and fine-tuned domain-specific models further contribute to improving reliability.

Second, it is unified techniques for heuristic design. The development of lightweight heuristic models and rule-based systems to determine the necessity of LLM invocation is crucial for optimizing computational efficiency and minimizing costs. Techniques such as feature extraction, statistical anomaly detection, and edge-based AI implementations enable precise heuristic design that ensures LLMs are leveraged only when necessary, thereby improving response times and reducing unnecessary computational overhead.

The third one is exhaustive testing across domains and use cases. While this study illustrates the efficacy of our pipeline in the banking and insurance industries, comprehensive validation across diverse sectors—such as healthcare, supply chain management, and legal automation—would provide a more robust evaluation. Establishing benchmark datasets, real-world stress testing, and defining formal quantitative assessment frameworks incorporating factors like accuracy, latency, and robustness under adversarial conditions are essential to enhance the generalizability of the approach.

The next is multi-tier cascades for scalability. The proposed architecture currently integrates two levels of cascading—rule-based automata and LLM agents—but for large-scale deployments involving millions of transactions, additional layers of processing could be introduced. These may include specialized domain-specific LLMs, hybrid architectures incorporating Fuzzy Automata, and reinforcement learning-based decision systems that dynamically adjust processing flows based on context and historical performance metrics.

We also need adaptive model selection and continuous learning. To enhance adaptability, dynamic model selection mechanisms can be integrated, allowing the system to switch between different LLMs or heuristic methods based on contextual requirements. Additionally, incorporating continuous learning mechanisms—such as federated learning and self-supervised training—can help improve model performance over time by assimilating real-world feedback and domain-specific updates.

The second last is robust explainability and interpretability frameworks, ensuring that the decision-making processes of LLMs remain transparent is a fundamental requirement for enterprise adoption, particularly in regulated industries. Incorporating explainability techniques such as attention-based visualization, causal inference methods, and post-hoc interpretability models would enhance user trust and regulatory compliance.

Finally, it is security and adversarial robustness measures. As LLMs are increasingly integrated into critical automation workflows, ensuring their resilience against adversarial attacks is paramount. Techniques such as adversarial training, robust input validation, and multi-layered authentication mechanisms should be incorporated to safeguard against potential security threats, data poisoning, or unauthorized access.

FUNDING INFORMATION

The authors state that no funding was involved in supporting the research described in this article.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Hrishikesh K. Haritas	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Vineet H. Sadarangani	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		
Ganeshayya Ishwarayya Shidaganti			✓	✓	✓	✓	✓		✓	✓		✓	✓	
Darshan Bankapure	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓		
Rahul K. Vishal	✓	✓		✓	✓	✓	✓		✓	✓				
Shreya Vijayasimha		✓		✓		✓	✓			✓	✓			

C	**C**onceptualization	I	**I**nvestigation	Vi	**Vi**sualization
M	**M**ethodology	R	**R**esources	Su	**Su**pervision
So	**So**ftware	D	**D**ata Curation	P	**P**roject administration
Va	**Va**lidation	O	Writing - **O**riginal Draft	Fu	**Fu**nding acquisition
Fo	**Fo**rmal analysis	E	Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data pertaining to the experiments are available through our GitHub repository at https://github.com/Vineet-Sadarangani/GraphBotics.

REFERENCES

[1] M. Lacity and L. Willcocks, "Robotic process automation at Telefonica O2," *MIS Quarterly Executive*, vol. 15, no. 1, pp. 21–35, 2016.

[2] L. P. Willcocks, M. Lacity, and A. Craig, "Robotizing global financial shared services at Royal DSM," London, UK, 2017.

[3] T. Tarquini, "Practical robotics in insurance – The future is here already," in *The InsurTech Book*, S. Chishti and J. Barberis, Eds. Wiley, 2018, pp. 231–235.

[4] R. Syed *et al.*, "Robotic process automation: Contemporary themes and challenges," *Computers in Industry*, vol. 115, p. 103162, 2020, doi: 10.1016/j.compind.2019.103162.

[5] ACCA Global, "The robots are coming? Implications for financial shared services," *accaglobal.com*, 2015. https://www.accaglobal.com/us/en/technical-activities/ (accessed Mar. 15, 2025).

[6] S. Burnett, M. Aggarwal, A. Modi, and S. Bhadola, "Defining enterprise RPA," 2018. [Online]. Available: https://www.fusionsol.com/wp-content/uploads/sites/22/2019/07/Everest-Group-UiPath-Defining-Enterprise-RPA.pdf.

[7] P. Hallikainen, R. Bekkhus, and S. L. Pan, "How OpusCapita used internal RPA capabilities to offer services to clients," *MIS Quarterly Executive*, vol. 17, no. 1, p. 4, 2018.

[8] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 1877–1901.

[9] J. Lu and Others, "Turn every application into an agent," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024, vol. 1, pp. 7711–7743.

[10] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," in *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023, pp. 68539–68551.

[11] S. Poddar, P. Koley, J. Misra, S. Podder, N. Ganguly, and S. Ghosh, "Towards sustainable NLP: Insights from benchmarking inference energy in large language models," *arXiv preprint arXiv:2502.05610*, 2025, doi: 10.48550/arXiv.2502.05610.

[12] S. Samsi *et al.*, "From words to watts: benchmarking the energy costs of large language model inference," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2023, pp. 1–9, doi: 10.1109/HPEC58863.2023.10363447.

[13] E. Miehling *et al.*, "Agentic AI needs a systems theory," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2503.00237.

[14] D. Edge *et al.*, "From local to global: A graph RAG approach to query-focused summarization," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2404.16130.

[15] IBM Cloud Education Team, "Hyperautomation: The benefits and challenges," *ibm.com*, 2025. https://www.ibm.com/think/insights/hyperautomation-benefits-and-challenges (accessed Mar. 15, 2025).

[16] A. Jain, S. Paliwal, M. Sharma, L. Vig, and G. Shroff, "SmartFlow: Robotic process automation using LLMs," *32nd ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1–9, 2024.

[17] Y. Chen *et al.*, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys '24)*, 2024, pp. 674–688, doi: 10.1145/3627703.3629553.

[18] C. Baquero, "The energy footprint of humans and large language models," *Communications of the ACM*, 2024. https://cacm.acm.org/blogcacm/the-energy-footprint-of-humans-and-large-language-models/ (accessed Mar. 15, 2025).

[19] S. Yin *et al.*, "A survey on multimodal large language models," *National Science Review*, vol. 11, no. 12, Nov. 2024, doi: 10.1093/nsr/nwae403.

[20] G. P. Reddy, Y. V Pavan Kumar, and K. P. Prakash, "Hallucinations in large language models (LLMs)," in *2024 IEEE Open*

Conference of Electrical, Electronic and Information Sciences (eStream), Apr. 2024, pp. 1–6, doi: 10.1109/eStream61684.2024.10542617.
[21] L. Huang *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint*, 2024, doi: 10.48550/arXiv.2311.05232.
[22] X. Zhu, Y. Xie, Y. Liu, Y. Li, and W. Hu, "Knowledge graph-guided retrieval augmented generation," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025, pp. 8912–8924, doi: 10.18653/v1/2025.naacl-long.449.
[23] K.-B. Ooi *et al.*, "The potential of generative artificial intelligence across disciplines: perspectives and future directions," *Journal of Computer Information Systems*, vol. 65, no. 1, pp. 76–107, Jan. 2025, doi: 10.1080/08874417.2023.2261010.
[24] T. R. McIntosh *et al.*, "From Google Gemini to OpenAI Q* (Q-Star): a survey on reshaping the generative artificial intelligence (AI) research landscape," *Technologies*, vol. 13, no. 2, p. 51, Jan. 2025, doi: 10.3390/technologies13020051.
[25] S. Brohi, Q. Mastoi, N. Z. Jhanjhi, and T. R. Pillai, "A research landscape of agentic AI and large language models: applications, challenges and future directions," *Algorithms*, vol. 18, no. 8, p. 499, Aug. 2025, doi: 10.3390/a18080499.
[26] T. Xie *et al.*, "OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments," *arXiv preprint*, 2024, doi: 10.48550/arXiv.2404.07972.
[27] I. Seilonen, T. Pirttioja, P. Appelqvist, K. Koskinen, and A. Halme, "Modelling cooperative control in process automation with multi agent systems," in *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN '04. 2004*, pp. 260–265, doi: 10.1109/INDIN.2004.1417341.
[28] K. Ronanki, "Facilitating trustworthy human-agent collaboration in LLM-based multi-agent system oriented software engineering," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, Jun. 2025, pp. 1333–1337, doi: 10.1145/3696630.3728717.
[29] J. Huang, Z. Li, and Z. Zhou, "A simple framework to generalized zero-shot learning for fault diagnosis of industrial processes," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 6, pp. 1504–1506, Jun. 2023, doi: 10.1109/JAS.2023.123426.
[30] J. Wang, "EDCEW-LLM: Error detection and correction in English writing: A large language model-based approach," *Alexandria Engineering Journal*, vol. 129, pp. 1153–1164, Oct. 2025, doi: 10.1016/j.aej.2025.08.005.

BIOGRAPHIES OF AUTHORS

Hrishikesh K. Haritas is a project associate at the Department of Computational and Data Sciences, Indian Institute of Science, Bangalore. He has previously worked on computer vision research projects with Samsung and explainable deep learning projects with Unisys. He holds a B.E. degree from Ramaiah Institute of Technology, India. He is part of a research collaboration between the National Payments Corporation of India and the Indian Institute of Science. He can be contacted at hrishikeshkh@gmail.com.

Vineet H. Sadarangani is currently working as an apprentice at Trellix (erstwhile McAfee Enterprise and Fireeye) as a Software Development Engineer in Test and has completed his undergraduate studies at Ramaiah Institute of Technology with distinction. He has accumulated over two years of industry experience through various internships at Unisys, Samsung R&D and Larsen & Toubro, gaining practical exposure to real-world challenges. His research interests lie in the field of Artificial Intelligence, with a particular focus on Computer Vision, Deep Learning, and related technologies. Vineet has been recognized with a Certificate of Excellence from Samsung for his contributions and has won an innovation competition organised by Unisys, reflecting his commitment to academic and practical excellence in the field. He is dedicated to advancing AI-driven solutions through both research and hands-on industry experience. He can be contacted at 1ms21ai062@msrit.edu.

Ganeshayya Ishwarayya Shidaganti is currently working as an associate professor in the Computer Science and Engineering Department at M S Ramaiah Institute of Technology. With over a decade of teaching experience, he has published 30+ research papers in international conferences/book chapters and journals, indexed in Scopus. He has received appreciation from UiPath Academic Alliance, for his leadership in enabling Institute Participation at UiPath, DevCon 2020. Apart from robotic process automation, he also teaches other educational technologies like cloud computing, big data and analytics, and computational intelligence. Under his guidance, several students have cleared UiRPA and have secured second place in Automation: Techfest IIT-Bombay, 2022. He has participated in and delivered multiple guest lectures throughout his teaching journey. He has co-authored a couple of chapters in a book, journal paper, and conference paper as well. He is also a member of professional societies: IEEE, ACM and CSI. He can be contacted at ganeshayyashidaganti@msrit.edu.

Darshan Bankapure is an associate software engineer at Unisys, Bangalore. He earned his bachelor's degree in artificial intelligence and machine learning from Ramaiah Institute of Technology. His Internships include developing no-code data transformation solutions at OneNine AI, full-stack development for campus digitization at Unisys, a novel per-pixel deflicker approach at Samsung R&D, and applying predictive analytics to propulsion systems at Larsen & Toubro Defence. Research contributions include publications in Taylor & Francis and IEEE, and leading projects on federated learning for privacy-preserved sleep apnea detection and innovations in parallel sorting algorithms. His notable hackathon victories include the Unisys Innovation Program Y-14 and Y-15. He can be contacted at 1ms21ai016@msrit.edu.

Rahul K. Vishal is a graduate in artificial intelligence and machine learning from Ramaiah Institute of Technology, Bangalore, with a strong focus on research and innovation in AI applications. Research contributions include a liver cancer detection model using CNNs, hybrid metaheuristic clustering optimization, federated learning for privacy-preserving sleep apnea detection, and real-time adaptive AR systems for vision enhancement, with publications in IEEE and other platforms. Notable works include developing per-pixel de-flickering methods at Samsung R&D, automating testing frameworks at Unisys, and advancing segmentation-based AR overlays using Meta and Apple technologies. His notable achievements include winning the Unisys Innovation Program Year 14. He can be contacted at 1ms21ai045@msrit.edu.

Shreya Vijayasimha is an electronics and communication engineer who graduated from Ramaiah Institute of Technology (RIT). She is currently contributing her expertise as a Full-Time Engineer at Centum T&S, where she focuses on embedded systems development for high-profile projects, including those for BHEL and Vande Bharat. Her background includes valuable research experience in metasurface-based technologies gained during an internship at the Indian Institute of Science (IISc) and academic project work focused on wireless power transfer and system optimization for implantable medical devices. She can be contacted at 1ms21ec106@msrit.edu.