SupportIQ: A Multi-Level Intelligent Query Resolution System

Jayashree Rajesh Prasad[1], Ruhi Pawar[2], Soham Kulkarni[3], Varun Damal[4], Shashank Naik[5]
[1,2,3,4,5] *Computer Science and Engineering, School of Computing, MIT Art Design and Technology University, Pune, Maharashtra, 412201, India.*

***Abstract*—In today's world where the IT infrastructure has become the backbone of the industry. Even the smallest of the tasks are performed using electronic devices like laptops with high computational capacities, large data centers consisting of tons of servers. These systems control the overall functioning of the company across the different technical and non – technical departments. Due to which a lot of manpower is spent on maintaining these systems. This paper looks at the proposal of automation of these systems by introducing a system which uses a ticketing system which classifies these tickets which are simply the queries of the employees into 3 types of tickets based on its recovery time objective (RTO), L1, L2, L3 are the categories. After categorization, this system uses TF-IDF (Term Frequency-Inverse Document Frequency), RAG (Retrieval-Augmentation Generation), and API (Application Programming Interface) each for one category and they then assist the employee with their problems and if human intervention is necessary then they also alert the specific technical personnel. This paper proposes the automation of IT management systems so that humans can perform high end and more complicated tasks.**

***Index Terms*—Ticket Categorization, TF-IDF (Term Frequency-Inverse Document Frequency), RAG (Retrieval-Augmentation Generation), API (Application Programming Interface), Automation of IT Maintenance Systems.**

I. INTRODUCTION

Information Technology Infrastructure is the backbone of IT business across all origins, for operational efficacy. Since help desks currently are the main interface between users and the technology they depend on, IT service Management plays a very important role. Although, the traditional help desks are often buried with the work of support tickets, many of which are repetitive, and deal with common issues like software access requests or passwords. resets. With its heavy reliance on manual ticket handling and creating these components, incorporating the applicable criteria that follow.

Nowadays, almost there is too much usage of the electronic and automation of the jobs across all spectrums of all the different industries. Industries nowadays are trying to incorporate various technologies at various levels catering to different usage which indirectly help them to curb the deadlines and produce more benefitting the industry.

This usage of technology in industries even the non-technological aspects have to incorporate the use of computers and complex software systems in their daily work usage . This makes it harder for the IT departments to manage the queries which are continuously popping up . Such tasks when done manually consume a lot of time and effort . While with the use of AI and various systems we can minimize this effort and make sure that the human intervention only comes when there is no other option left like when it comes to server crash etc.

This project aims to cater this aspect of the Industries and aid them with a software system which helps them to automize their queries so that they have the humans to perform more R&D (research and development) Tasks by which they can create more efficient products while the management tasks of managing the smaller queries of the employees which are comparatively easier can be performed by AI and the industry can work more efficiently .

II. PROBLEM STATEMENT

This research paper aims to provide systems which prove an IT service management framework used for classifying as well as address issues for a multi – tiered

ticketing systems using TF-IDF (Term Frequency-Inverse Document Frequency), RAG (Retrieval-Augmentation Generation), API (Application Programming Interface) with three levels of ticketing systems L1 , L2 , L3 (where L1 and L2 are performed completely by the system) as part of its main architecture. The following are some aspects of our project:

A. Optimization of the resolution systems:

It aims to minimise the time required to resolve the queries till the L2 levels and also for the L3 levels we can have a faster detection of the problem and notifying of flagging the problems for the humans to intervene for the L3 levels .

B. Automization of the manual work and Analysis for the Support Systems:

We can reduce the manual workload and instead invest them for more complex tasks while such smaller tasks which can be done by our system and these can be done quicker so it saves a lot of time .At the same time we can efficiently mark which are the points where we are finding the maximum amount of problems by flagging the repetitive queries .

III. LITERATURE SURVEY

Sr.	Publication / Ref No.	Method	Features	Limitations
1	[1] Baqar (2025)	Retrieval-Augmented Generation (RAG)	It suggests the framework RAG4Tickets, which applies FAISS the vector search and Sentence Transformers. This also generates the recommendations for resolutions which help in integrating data from sources like GitHub (pull requests, comments) and JIRA tickets. Also, it helps in the retrieval of comparable previous cases by embedding code metadata and tickets.	FAISS index type shows the implicit challenges in terms of the input user balancing query speed, accuracy, and memory usage.
2	[2] Liu et al. (2025)	LLM-Based Framework (TickIt)	It comprises of an online ticket escalation framework ("TickIt") which use the LLMs.	An LLM-based online ticket escalation system. Helping facilitating the relationship-driven, dynamic, as well as topic-aware escalation.
3	[3] Lewis et al. (2020)	Retrieval-Augmented Generation (RAG)	The base foundational RAG paper. It also combines pre-trained parametric memory with non-parametric memory.	It involves the integration of the non-parametric memory along side of pre-trained parametric memory which reaches SOTA for NLP tasks.

4	[4] Zhang et al. (2025)	Multi-Agent RAG	Uses an offline-first multi-agent system for the execution process unstructured data (tickets, emails, social media data) before it's used by RAG.	Before being used in RAG, unstructured data is processed using an "offline-first" system. Categorization, Knowledge Synthesis, and Category Discovery are all done by agents.
5	[5] Engelbrechtsmüller (2024)	Fine-Tuning LLMs	Evaluates the improved GPT and BERT models for classifying the IT tickets with respect of their domains. Utilizing the data augmentation methods (AEDA and EDA) to enhance BERT's functionality.	All-purpose LLMs are not pre trained for fine tuning, which frequently struggle to comprehend domain specific text, such as that found in IT tickets.
6	[6] Wulf & Meierhofer (2024)	LLM Prototyping (incl. RAG)	It analyses LLM in customer support and finds the ones which are effective at different levels (low levels and high levels each level has its own requirements for its performance).	It research's in the customer service which are good enough for low- and high-level tasks.
7	[7] Zhang et al. (2024)	Supervised Fine-Tuneable RAG	It proposes the RAG4ITOps, which is a comprehensive framework for IT operations.	Suggests the inclusive IT operations framework which are known as RAG4ITOps. These improve performance over tasks by directly integration of supervised fine tuning with RAGframe.
8	[8] Henriksson & Grattan (2025)	Quantized LLM with RAG	It explores and research's the models which are quantized and are equalling and these models can surpass GPT-4 in the terms of its technical performance also helps in highlighting the RAG pipeline	The Qwen2.5 14B, a smaller, quantized, self-hosted LLM, as an affordable substitute for SOTA APIs for

			alongside the infamous knowledge base.	SMEs. It is discovered that the quantized model can perform low complexity, even better than GPT-4. Therefore, the importance of the RAG pipeline and the quality of the knowledge base is emphasized.
9	[9] Virpiö (2025)	LLM Agents (Design Science Research)	It caters to the main issue of the required accurate blend of probabilistic and deterministic generation and code respectively. It also forms the very basis of prompt engineering.	It examines trust in human-LLM agent interactions.
10	[10] Della Sala (2025)	RAG with Fine-Tuned LLaMA 3	The RAG systems using LLaMA3 model which is used for fine tuning on custom datasets for a specific product (Bogen E7000) it has significantly outperformed the base model with the custom domain understanding.	Applies RAG using a LLaMA 3 model that was refined on a special dataset of technical manuals for a specific product (Bogen E7000). The improved RAG model outperformed the base mode in terms of usefulness and domain comprehension.
11	[11] Nikolova & Toleva-Stoimenova (2025)	LLM Application (Literature Review)	It analyses the potential of LLMs to enhance bug reporting. LLMs can automate triaging, generate summaries, assist in test case creation, and predict severity are the LLMs which are identified.	It looks at how bug reporting could be enhanced by LLMs. shows how LLMs can be used to forecast severity, generate summaries, automate triaging, and create test cases.

12	[12] Zhang & Chen (2025)	Fine-Tuned Transformers vs. LLMs	The best performance was obtained with fine-tuned transformers (RoBERTa) (F1=0.90). Zero-shot LLMs (LLaMA3-70B) were incredibly slow (139 minutes vs. seconds) but had similar accuracy (F1=0.86).	Large language models have high computational costs as well as long inference times make them unsuitable for handling humongous volumes of data.

IV. DATA PREPARATION

In the project we have used two kinds of data Structured datasets (from Kaggle) and Unstructured data which is Retrieval-Augmentation Generation (RAG) corpus.Therefore, we have two tiers of data which are used for training purposes.

A. Structured Dataset (FAQ , Tier-1 Data)

In this we have a supervised dataset with a key-value pair structures which are very simple to implement and have high precision while matching.

- Question based query: It is the query that a normal user will input into the system.

i) It is the input feature, TF IDF Vectorizer used in the generation of sparse vector representations.

- Resolution (the answering system): These is static data which has the solution and this script is pre verified and is the resolution to the users' query.

i) Input query solution is obtained by the similarity matching.

B. Unstructured Dataset (Text corpus, Tier-2 Data)

Retrieval Augmentation Generation corpus is the Tier-2 dataset for our project. It has the collection of the unstructured text which has the technical documentation .

i) Relevance for project : Distinct semantic blocks containing data extracted from the relevant technical and architectural guides.

ii) The Data Entities in the project :

In the data which we have used that is the corpus consists of a wide range of domains for handling complex and diverse data which contains specific entities from these domains which are critical for the IT Service Management systems. The following are the categories :

(1) API Integration and Optimization :

(a) Limiting Rate : It puts constraints dictating a maximum throughput of the API throughput of 1000 requests per hour per account .

(b) SDK Initialization : It contains the procedure for importing as well as initializing mobile SDK modules using unique API keys .

(c) Tuning : It helps in tuning performance as well as for the implementation of asynchronous API calls for the optimization of the latency of our application.

iii) Security Protocols:

(1) Access Management : It uses JSON Web Tokens , Oauth2protocols and backend session validation for the flow of authentication.

(2) Cryptographic Standards: It uses HTTPS and Advanced Encryption Standard (AES-256) for implementation of the data encryption policies.

iv) Service Level Agreements (SLA):

(1) It contains the operational parameters which are available at any time of the day also by this the help desks which run 24/7 can also be connected as it has the capability of connecting the help desks via an email or phone.

C. Vectorization Techniques as well as the Structural features:

- Transformer Model: Every textual chunk is going through the all MiniLM-L6-v2 sentence transformer model.
- Dimensions of the model: The models convert human readable texts into the high dimensional numerical arrays (numpy arrays are being mapped to float32).

- Indexing System: The embeddings are ingested to FAISS (Facebook AI Similarity Search) index (IndexFlatL2) which is responsible for calculating the distance of L2 (Euclidean distance) to get the most proximate (accurate) solution and mapping for its ticket classification and likewise provide the solution by the system.

V. SYSTEM ARCHITECTURE

Once we had our data planned and structured we see the structure of the whole system , it has a hierarchical structure having an AI pipeline designed to assess , classify and based on the classify provide a solution for the IT queries. The proposed system having a three tier architecture(L1,L2,L3).All of this is managed by a Flask based RESTful API which takes responsibility for user input and asynchronous routing.[2][4][6].

A. Tier 1(L1 level tickets):
It is a FAQ retrieval system designed for the most basic queries , these generally are the queries which are asked mostly and have a common solution .

1) Vectorization technique : When the system receives an unstructured query data from the user , the input is converted or rather transforms the user input into numerical representation by Term Frequency Inverse Document Frequency (TF-IDF).

2) The main logic : The system uses cosine similarity between the input vector and the L1 query and then match with the TF-IDF vectors for the FAQ structured data of the dataset.

B. Tier 2(L2 level tickets):
These are the tickets which do not match L1 module these are a little advanced user queries . These are the queries which are solved using the Retrieval Augmented Generation (RAG) approach.
1) The Transformer Model of the project : The queries are transformed by a pre-trained model **all-MiniLM-**L6-v2 .2) Vector Similarity: This tier uses vector similarity it uses FAISS (Facebook AI Similarity Search) IndexFlat2 database. This helps the system store and evaluate unstructured data.
3) Euclidean Distance: In this Tier we use Euclidean distance for calculating the distance between the user query and the L2 tier data. This is different from Tier 1 as in that system used cosine similarity instead of Euclidean distance.[1][2]

C. Tier 3(L3 level tickets):
This is the tier which handles the most complex queries which have not matched Tier 1 or Tier 2.
a) Integration Of API:This module has direct use of LLM (Large Language Model) using the API key of Gemini API .
b) Dynamic Characteristic of the module: In the previous tiers we had a pretrained data with which we compared the user query and then the output was given. However, in this tier the system uses Gemini API by which there is a LLM model which dynamically generates the required solution for the user query.

VI. RESULTS AND EVALUATION

In this section of the paper, we evaluate our proposed IT Support System which is a multi-tiered system based on empirical threshold, local knowledge base with its structural boundaries as well as the routing efficiency of the based on similarity constraints by the Application Programming Interface (API).

A. Empirical Threshold: This is the accuracy of the multi-tiered routing mechanism which measures the ability to prevent the false positive resolution these are strictly measured by the similarity thresholds. The following are used in each of the three tiers of the system:

1. Tier 1: This uses Cosine Similarity for the incoming user queries which first are converted using TF_IDF into a numerical vector then the cosine similarity score is checked the threshold is greater than 0.65 against the FAQ vector (this is a known dataset).

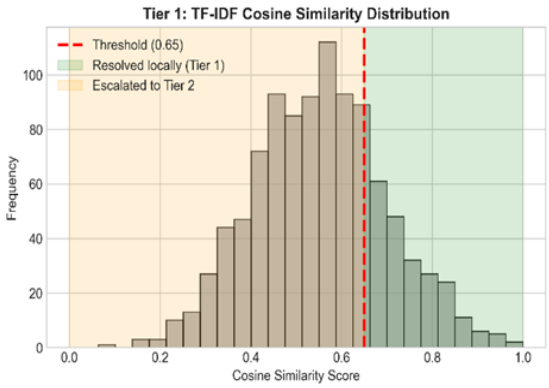

Fig 1. Cosine Similarity Threshold (Tier 1)

2. Tier 2: This tier of our system uses, a RAG (Retrieval Augmentation generation) it uses Euclidean distances which helps in calculating the distance between the vectors which are complex queries which use FAISS indexFlatL2 , the system has to match L2 distance is less than 1.0 if it is greater than 1.0 then the query is passed on L3 query detection.

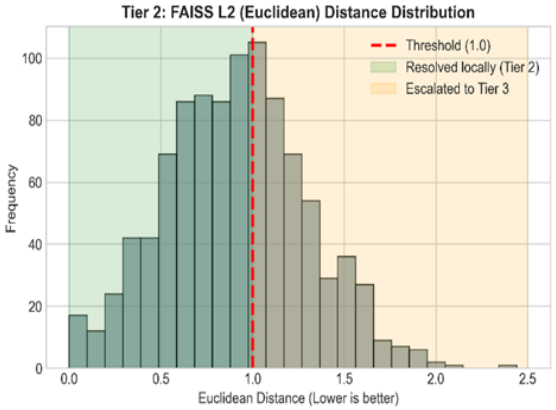

Fig 2. FAISS L2 Uses Euclidean Distance

3. Tier 3: This tier uses Gemini API this module does not calculate any distance metric instead it invokes LLM (Large Language Model) we have used Gemini 1.5 Flash LLM model which is accessed using Gemini API. This tier uses dynamically generated answers by the model unlike the previous models. If the external API requests fails then this system bypasses the application crash and returns a degradation message. This step helps with the stability of application.[8]

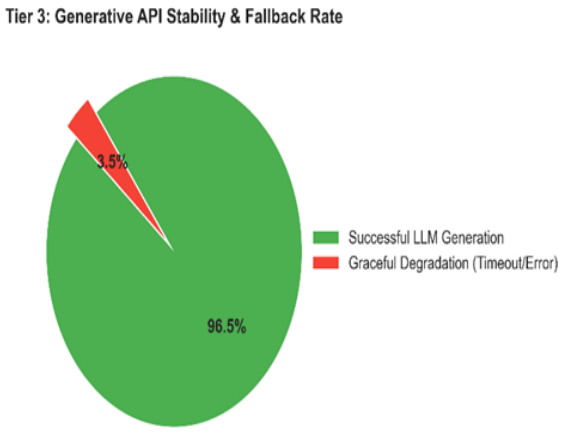

Fig 3. The Generative API (GEMINI API)

These are the above evaluation metrics that we have used to evaluate the system we have used multi tiered system therefore each tier has its own evaluation method which is according to the methods and techniques used in their respective tier.

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion:

In this paper we have proposed an IT Support System which has a multi-tier system architecture. This proposed architecture has Tier 1 (FAQ) , Tier 2 (RAG), Tier 3 (LLM) each Tier has its own techniques which cater to three different types of queries ranging from simplest to the complex queries , this system has incorporated the techniques like TF-IDF , Cosine Similarity , FAISS , Euclidean distance and integration of API we have used Google's Gemini API .This integration helps our system or a highly cost effective , fast as well as scalable framework which can optimize the working of IT Support Systems , this proposed framework has been designed with the sole purpose of assisting the support desks for the IT Industry which handles heavy traffic of queries which are manually handled therefore our system automates this process and helps the Support desk and allows the human resource of the IT Industry for more complex tasks.

B. Future Scope:

Our current prototype works seamlessly in automation of IT Automation Systems although there are some key areas to focus on which include:

1. Relational Database Migration: Currently FAISS index is being used which needs to be migrated to a relational data base with vector extensions allowing dynamic, real time CRUD (Create, Read, Update, Delete) operations by this we will not a full recompilation of vector indexing.
2. Frequent Updates of the LLM models: We have used Gemini 1.5 Flash LLM model which provides a cost-effective architecture. Although if we want to achieve excellent results with this fast changing world, we will need to upgrade LLM model which helps with complete data privacy for sensitive data.
3. Enhancement of NER Capabilities: When custom NER models for the L1 ticketing tier by which we can include domain specific key words as well as

we can incorporate different languages and local language slang words.

REFERENCES

[1] Baqar, M. (2025). RAG4Tickets: AI-Powered Ticket Resolution via Retrieval-Augmented Generation on JIRA and GitHub Data. arXiv preprint arXiv:2510.08667.

[2] Liu, F., He, X., Zhang, T., Chen, J., Li, Y., Yi, L., ... & Shi, R. (2025, June). TickIt: Leveraging Large Language Models for Automated Ticket Escalation. In Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (pp. 343-354).

[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33, 9459-9474.

[4] Zhang, Y., Shang, Z., Patel, S., & Zuniga, M. (2025). From unstructured communication to intelligent RAG: Multi-agent automation for supply chain knowledge bases. arXiv preprint arXiv:2506.17484.

[5] Engelbrechtsmüller, P. (2024). Fine-Tuning Large Language Models for Ticket Classification at Doka GmbH/submitted by Philipp Engelbrechtsmüller.

[6] Wulf, J., & Meierhofer, J. (2024). Utilizing large language models for automating technical customer support. arXiv preprint arXiv:2406.01407.

[7] Zhang, T., Jiang, Z., Bai, S., Zhang, T., Lin, L., Liu, Y., & Ren, J. (2024). RAG4ITOps: A supervised fine-tunable and comprehensive RAG framework for IT operations and maintenance. arXiv preprint arXiv:2410.15805.

[8] Henriksson, S., & Grattan, M. (2025). Towards Efficient LLM Deployment in Customer Service: An Exploratory Evaluation of a Quantized LLM.

[9] Virpiö, M. (2025). LLM Agents & Trust-Role of Deterministic Code: Design Science Research for Trust in Human-AI Interaction.

[10] Della Sala, J. (2025). Technical Customer Service Support with RAG Fine Tuned LLaMA 3.

[11] Nikolova, T., & Toleva-Stoimenova, S. (2025, June). Application of Large Language Models for Enhancing Bug Reporting Activities. In 2025 MIPRO 48th ICT and Electronics Convention (pp. 1928-1932). IEEE.

[12] Zhang, X., & Chen, M. (2025). Improving Crash Data Quality with Large Language Models: Evidence from Secondary Crash Narratives in Kentucky. arXiv preprint arXiv:2508.04399.

[13] Nihal, A. (2025). A comparative study of retrieval-augmented generation (RAG) and fine-tuned large language models in conversational AI.

[14] De Aquino, G. D. A., de Azevedo, N. D. S., Okimoto, L. Y. S., Camelo, L. Y. S., de Souza Bragança, H. L., Fernandes, R., ... & Torné, I. G. (2025). From RAG to Multi-Agent Systems: A Survey of Modern Approaches in LLM Development.