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

DOI: 10.17148/IJARCCE.2026.151121

AI Based Real-Time Data Analysis and Decision Generation System for Gokul Text Print

Dr. Pankaj V. Baviskar[1], Hemanshu Umesh Suryawanshi[2], Priyanshu Dipak Patil[3], Lakshvedhi Anil Patil[4], Gaurav Manohar Patil[5]

Department of Artificial Intelligence & Machine Learning, R. C. Patel Institute of Technology Shirpur, India[1-5]

Abstract: Manufacturers rely on enterprise resource planning (ERP) systems to track sales, inventory, production, and order fulfilment, yet the records often sit in separate modules and need heavy manual reading before they can guide decisions. This paper presents an AI decision-support platform for Gokul Text Print that links ERPNext data with machine-learning analytics, explainability, retrieval-augmented generation (RAG), and specialized AI agents. A Frappe REST API client and a validation-focused extraction, transformation, and loading (ETL) pipeline pull and check the records. The sales module classifies decision direction and derives probability-based projections, the inventory module estimates reorder risk, SHapley Additive exPlanations (SHAP) support model interpretation, and a Sentence Transformers and Facebook AI Similarity Search (FAISS) retriever supplies organizational knowledge for grounded answers. A manager agent combines the outputs of sales, inventory, and knowledge agents before a language model writes the final response, and a Flask dashboard shows indicators, recommendations, alerts, and reports. Preliminary development runs gave 68.45% accuracy and a receiver operating characteristic area under the curve (ROC-AUC) of 0.7447 for sales, and 100% accuracy with a ROC-AUC of 1.0000 for inventory, a result that still needs temporal and leakage checks. The contribution is an auditable integrated architecture rather than a new algorithm, together with an evaluation framework for prediction, retrieval, agent quality, and reliability.

Index Terms: Enterprise resource planning, explainable artificial intelligence, inventory management, machine learning, manufacturing analytics, multi-agent systems, retrieval-augmented generation, sales intelligence.

I. INTRODUCTION

A. Background and Motivation

manufacturing firms operate in settings where timely decisions on sales, inventory, production, and fulfilment directly affect efficiency. ERP systems give organizations one place to record this activity [1], yet having the data does not mean it turns into good decisions by itself. In practice, information is scattered across transaction records, inventory reports, production schedules, and policy documents, so managers must open several reports, spot patterns by eye, and check procedures before acting, and two people can read the same figures differently.

Recent AI advances offer help. Machine-learning models can find patterns in historical records [7]–[9], explainability methods show why a model produced an output [4], retrieval-augmented generation (RAG) ties language-model answers to organizational knowledge [3], and large language models are drawing growing attention in manufacturing [2]. Multi-agent designs split a task among specialized components and merge their results [5]. This work combines these capabilities into a decision-support platform for Gokul Text Print, joining ERPNext data, sales and inventory intelligence, explainable AI, knowledge retrieval, specialized agents, and an executive dashboard in one workflow.

B. Problem Statement and Objectives

The core problem is the distance between the data held in an ERP system and the actionable insight managers need. A useful platform has to pull current data reliably, catch incomplete or inconsistent records, build suitable predictive models, explain their outputs, retrieve relevant knowledge, and merge the evidence into clear recommendations, while coping with missing data, API failures, and uncertain or conflicting information without presenting unsupported conclusions as fact. The platform is therefore modular and is meant to assist human decisions, not to execute high-impact production or procurement actions automatically.

Our objectives are to integrate ERPNext records through the Frappe REST API [6]; validate and prepare sales, inventory, and production data; build services for sales decision-direction classification and inventory reorder-risk prediction; explain predictions with SHAP [4]; add a RAG pipeline for business questions [3]; coordinate specialized agents through a manager agent [5]; provide an executive dashboard; and define an evaluation framework for predictive performance, retrieval relevance, agent response quality, and reliability. We ask how well ERP data supports sales and inventory intelligence, how far explainability improves transparency, whether RAG supplies relevant organization-specific context, how multi-agent orchestration compares with direct language-model answering, and what reliability and latency trade-

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



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151121

offs arise. The contribution is a unified workflow, not a new algorithm, and its effectiveness still has to be shown through controlled experiments.

II. BACKGROUND AND RELATED WORK

A. AI-Enabled ERP Systems

ERP systems hold structured records of transactions, inventory, production, and orders, so their centralized data can feed predictive analytics and decision assistance when suitable analytical components exist. Recent reviews name predictive analytics, automation, and generative AI as key directions, while noting concerns about governance, trust, integration complexity, and the reliability of generated recommendations [1]; reviews of large language models in manufacturing raise similar points [2]. An effective solution must therefore do more than display history: it should turn records into indicators and give managers evidence to judge each recommendation.

B. Machine Learning for Manufacturing Decisions

Random Forests [8] combine many decision trees to model nonlinear relationships, and gradient-boosting methods such as XGBoost [7] and LightGBM [9] build an ensemble step by step, each tree correcting the errors of the last. They suit structured operational data. For sales, lagged values, moving averages, trend, and momentum describe changing demand; for inventory, available quantity, safety stock, turnover, production gap, and fulfilment rate inform reorder risk. Their usefulness depends on data quality and on being available at prediction time. The sales module here is a classifier whose projections come from class probabilities and a baseline feature vector, so it is not a direct regression model unless one is separately built and evaluated.

C. Explainable Artificial Intelligence

SHapley Additive exPlanations (SHAP) uses a game-theoretic framework to estimate how much each feature contributes to a prediction [4]; global explanations summarize influential features across a dataset, and local ones explain a single prediction. In the platform, SHAP lets managers see, for example, whether recent sales patterns or stock variables drove an output. An explanation does not prove the model is correct or that a feature causes an outcome, so we treat it as a transparency aid, not a substitute for validation.

D. Retrieval-Augmented Generation.

RAG pairs retrieval with generation: instead of relying only on what is stored in the model's parameters, the system retrieves passages from an external collection and gives them to a language model as context [3]. This helps when answers must reflect organization-specific procedures or documents. We use Sentence Transformers with the all-MiniLM-L6-v2 model, a transformer-based encoder [10] that outputs 384-dimensional vectors, and FAISS for similarity search; after L2 normalization, inner-product search approximates cosine similarity. Answer quality still depends on document coverage and freshness, retrieval relevance, and faithful use of the evidence, so retrieval and grounding must be evaluated separately.

E. Multi-Agent Systems and Research Gap

Multi-agent systems divide work among specialized language-model agents and coordinate their outputs; frameworks such as AutoGen show how agent conversations can support applications needing several reasoning or tool-use steps [5]. Here, separate agents handle sales, inventory, and organizational knowledge. Specialization improves modularity but can add routing errors, duplicated work, inconsistent intermediate results, and latency, so its value should be measured against simpler baselines.

Earlier work has treated ERP analytics [1], predictive modelling [7]–[9], explainable AI [4], RAG [3], and multi-agent reasoning [5] largely separately (Table I). What is missing is a practical pipeline that reads ERP data directly, produces sales and inventory intelligence, retrieves organizational knowledge, and presents evidence-aware recommendations. This study fills that gap with a modular platform for Gokul Text Print and an evaluation framework for predictive performance, retrieval relevance, agent quality, and reliability.

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

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

DOI: 10.17148/IJARCCE.2026.151121

Research area	Primary capability	Remaining challenge addressed by this study
AI-enabled ERP	Operational data and enterprise automation	Connecting ERP records to a complete decision workflow
Machine learning	Predictive patterns and classification	Validating outputs against business decision requirements
Explainable AI	Feature-contribution explanations	Connecting explanations to executive-facing decisions
RAG	Retrieval-grounded language generation	Evaluating organizational relevance and source correctness
Multi-agent AI	Specialized task coordination	Measuring the value of orchestration over simpler baselines
Proposed platform	Integrated decision support	End-to-end validation and operational reliability

Table I · Comparison of related research areas

III. PROPOSED SYSTEM ARCHITECTURE

A. System Overview

The platform is modular (Fig. 2). ERPNext supplies records, the Frappe REST API client retrieves them [6], and the ETL pipeline validates and prepares them for the sales and inventory models. SHAP interprets applicable outputs, the RAG service retrieves organizational knowledge, specialized agents handle domain requests, and a manager agent combines their outputs before language-model synthesis. A Flask dashboard presents indicators, recommendations, alerts, and reports. Keeping these stages separate makes each easier to test and helps locate whether a fault lies in ingestion, inference, retrieval, or generation.

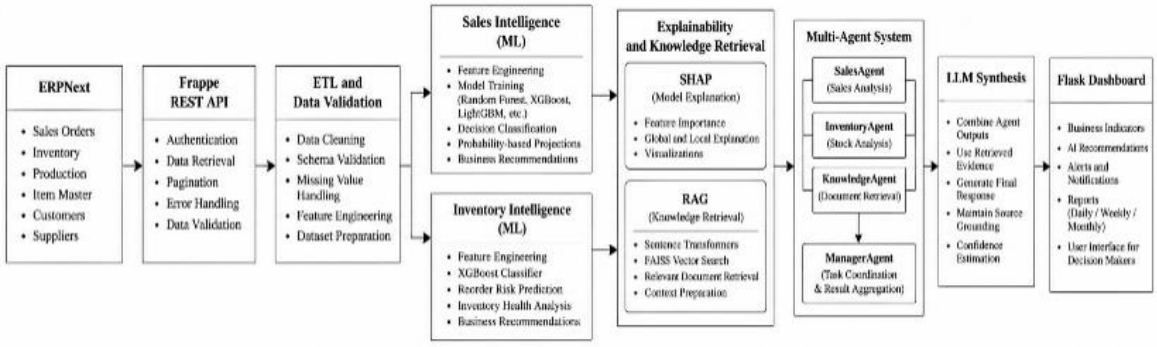


Fig. 2: Overall Architecture of the Proposed AI Decision-Support Platform

B. ERPNext and Frappe API Integration

Records are retrieved from ERPNext through the Frappe REST API, which exposes resources at /api/resource/{doctype} and accepts an API key and secret in the HTTP Authorization header using the token scheme [6]. The client supports pagination, retry handling, a record cap, and deduplication by record name, and covers sales orders, stock records, and work-order data depending on account permissions. It is kept separate from dashboard and machine-learning code so it can be tested alone, and credentials belong in environment configuration, never in public code or logs.

C. Data Extraction, Transformation, and Validation

The ETL layer checks expected columns and types, row counts, missing-value fractions, plausible ranges, and duplicates, and logs failures so incomplete data is never silently treated as current. Validated records become task-specific datasets, and the feature schema must stay identical between training and inference, since differences in column names, units, or missing-value handling can make predictions unreliable. If a request fails, a field is missing, or the dataset is unexpectedly small, the platform should tell the user rather than present stale results as up to date.

D. Sales Intelligence Module

The sales module classifies decision direction (target_decision) from current sales, lagged sales, moving averages over three, seven, and fourteen periods, seven-period variability, momentum, trend, and stock ratio. Candidates are Random Forest [8], XGBoost [7], LightGBM [9], and Gradient Boosting, tuned with RandomizedSearchCV (eight iterations, three-fold cross-validation) and scored by negative root mean squared error, then compared on RMSE and mean absolute error. The model outputs a class and probability, which, with a baseline feature vector, yield 7-, 30-, and 90-day projections; this is probability-derived decision intelligence, not a regression forecast.

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



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151121

A deterministic layer maps projected growth above 15% to increasing production, growth below −10% to reducing inventory, and anything between to maintaining production. These thresholds are rule-based and should be checked against business requirements.

E. Inventory Intelligence Module

The inventory module estimates reorder risk (reorder_flag) with XGBoost [7] from total stock, warehouse count and distribution, stock concentration, planned and completed production, production gap, fulfilment rate, open-order status, available quantity, safety stock, turnover, days in inventory, fast- and slow-moving indicators, dead-stock status, and reorder level. False reorders raise carrying costs and missed ones cause shortages, so the model needs precision, recall, F1-score, ROC-AUC, precision-recall AUC, and a confusion matrix, plus a check that no feature leaks the target or is unavailable at prediction time.

F. Explainability and Knowledge Retrieval

SHAP [4] feature contributions are shown beside supported predictions, but they are read in the context of the model and data and never as causal evidence. The retrieval service embeds passages with all-MiniLM-L6-v2, searches them with FAISS, and returns a bounded set of passages with source information, using a short-lived cache. Retrieved passages go to the language-model service as evidence [3]; when evidence is missing, the system should say so rather than make an unsupported claim.

G. Multi-Agent Orchestration

SalesAgent handles sales questions and predictive outputs, InventoryAgent handles stock and reorder requests, and KnowledgeAgent retrieves organizational information through the RAG service. ManagerAgent selects eligible agents, runs them concurrently where appropriate, merges their structured outputs, builds a synthesis prompt, and calls the language model (for multi-agent background see [5]). Its confidence value is the mean of the active agents' confidence, an internal aggregate and not a calibrated probability of correctness.

H. Executive Dashboard

The Flask dashboard shows total sales, revenue, sales growth, inventory value and health, low-stock products, AI-generated analysis, a business-health score, and aggregated recommendations. Alerts cover sales spikes or declines, low-stock and overstock conditions, ETL and ML-service failures, low AI confidence, and missing data, and daily, weekly, or monthly reports can be exported as PDF or CSV. Recommendations support managers; production and procurement decisions remain subject to business rules and human approval.

IV. METHODOLOGY

A. Data Collection and Preparation

Authenticated ERPNext records are retrieved with pagination and retry [6] and then checked by the ETL layer (Fig. 3). The sales run uses 23,740 rows and the inventory run 62,366 rows; these are separate task-specific datasets. A final report should document the date range, number of entities, target distribution, preprocessing, and exact partitions, and no test-set information may influence feature selection, tuning, or preprocessing, which matters most for temporal sales and inventory data where future information can leak into historical predictions.

B. Feature Engineering

For sales, lagged values capture recent history, moving averages smooth fluctuations, variability, momentum, and trend describe change, and stock ratio relates sales to stock. For inventory, stock and warehouse features describe quantity and concentration, production gap and fulfilment rate link planned output to delivery, and safety stock, available quantity, turnover, age, movement category, and reorder level add further status. Every feature must be computable at the prediction timestamp, features based on future transactions or target-derived calculations must be excluded or justified, and inference code must match the training definitions.

C. Model Training and Selection

Sales and inventory are trained as separate classification problems, using the candidates and tuning described above for sales and XGBoost [7] for inventory. The inventory model should be compared with simple baselines such as a reorder-level rule, since a complex model is worthwhile only if it clearly beats simpler procedures. Time-dependent records should be split chronologically, for example 70% training, 15% validation, and 15% test, with tuning limited to the first two and the test set used once.

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

DOI: 10.17148/IJARCCE.2026.151121

D. Evaluation Metrics

Sales classification is evaluated with accuracy, precision, recall, F1-score, ROC-AUC, and a confusion matrix; projections, if evaluated, need mean absolute error, root mean squared error, and a percentage metric such as symmetric mean absolute percentage error, used carefully near zero sales. Inventory uses precision, recall, F1-score, ROC-AUC, precision-recall AUC, and a confusion matrix, with attention to class imbalance and the different costs of false and missed reorders. RAG is assessed with Recall@k, Precision@k, mean reciprocal rank, and latency on curated questions, and answers for correctness, groundedness, and unsupported claims; multi-agent output is compared with direct language-model and general-agent baselines on the same questions.

E. Ablation and Robustness Evaluation

An ablation study compares four configurations (A: machine-learning outputs only; B: plus SHAP; C: plus RAG; D: the full specialized-agent system) to see whether each layer improves correctness, traceability, or answer quality. Robustness tests should cover incomplete ERP records, stale documents, API failures, ambiguous questions, low-confidence outputs, retrieval failures, and conflicting agent responses, with latency measured on a documented configuration. These tests belong to the evaluation plan and should be reported as completed only when measured.

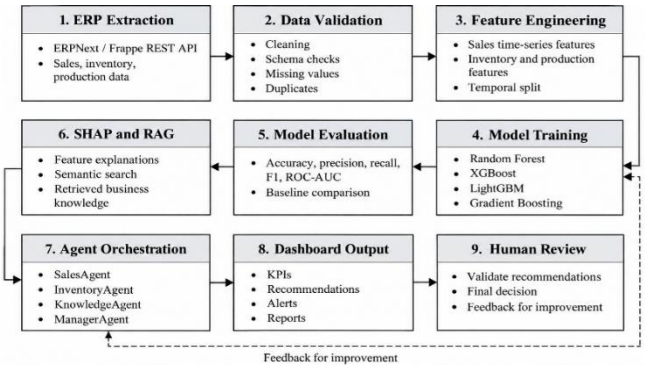


Fig. 3. Methodology and experimental workflow.

V. SYSTEM IMPLEMENTATION

A. Software Architecture

Data access, orchestration, machine-learning services, retrieval, and presentation are kept separate. The ERP client is in services/frappe_client.py, the orchestration layer in app/business_ai_layer.py routes dashboard requests to domain services, and the sales and inventory code sits under app/ml/sales/ and app/ml/inventory/. This lowers coupling and lets each part be tested or changed independently as long as service interfaces stay consistent.

B. Model and Recommendation Services

The sales service combines classification, probability-based projection, deterministic recommendation, and SHAP explanation, and the inventory service uses XGBoost [7] for reorder risk. Because outputs differ in origin, we distinguish a trained prediction from a calculation or a rule: for example, a remaining-inventory-days value from a formula is not a machine-learning prediction. This lets users tell whether a result comes from a model, a business rule, retrieval, or language-model synthesis.

C. Knowledge Retrieval and Language-Model Services

The knowledge service embeds text with all-MiniLM-L6-v2 and searches a normalized FAISS IndexFlatIP index, returning passages and sources to the calling agent. A transformer-based language model [10], configured as Gemini 2.5 Flash with an OpenAI-compatible alternative, combines this evidence with structured agent outputs to write the response [2], [3]. The configuration behind any reported experiment should be documented, and caches must be managed when documents change, with source identifiers kept so answers can be checked.

D. Monitoring, Security, and Reporting

Ingestion errors, missing data, and ML or AI service failures appear as explicit system states, not as recommendations that merely look valid, and alerts separate operational issues such as low stock from technical failures. Secure deployment needs least-privilege ERP credentials, environment-based secrets, dashboard authentication, and logs that avoid credentials; reports should record their interval and timestamp, and production use should log model versions, data timestamps, and recommendation sources.

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

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

DOI: 10.17148/IJARCCE.2026.151121

VI. EXPERIMENTAL RESULTS AND EVALUATION

A. Preliminary Classification Results

Fig. 4 shows the classification results recorded during development; they are development-stage figures not yet confirmed on a held-out, chronologically separated test set.

Task	Training rows reported	Accuracy	ROC-AUC
Sales decision classifier	23,740	68.45%	0.7447
Inventory reorder classifier	62,366	100%	1.0000

Fig. 4. Reported Development- Stage Classification Result

B. Sales Model Results

The sales classifier reached 68.45% accuracy and a ROC-AUC of 0.7447, an initial sign that it separates the configured decision classes, though not proof the recommendations are useful. Further work should report class distribution, precision, recall, F1-score, a confusion matrix, and the cost of wrong increase or reduce decisions, and evaluate the probability-derived projections separately against baseline forecasts.

C. Inventory Model Results

The inventory classifier recorded 100% accuracy and a ROC-AUC of 1.0000. Such perfect separation may reflect highly separable data, but also target leakage, duplicates across partitions, target-derived features, or an unrepresentative evaluation. It should be retested on a leakage-safe, chronologically ordered partition, with each feature confirmed as available at prediction time, duplicates checked across partitions, label generation reviewed, and full metrics and error costs reported. Until then it is a preliminary result, not proof of flawless performance.

D. End-to-End System Evaluation

Verified results for RAG retrieval quality, agent answer correctness, source grounding, and end-to-end latency are not yet available, so these remain pending experiments. The final evaluation should send the same documented business questions, with expected answers and known source documents, to a direct language-model baseline, a general-agent baseline, and the specialized-agent system, recording correctness, routing accuracy, source usage, unsupported claims, failure rate, and latency, with review by domain-informed evaluators.

VII. DISCUSSION

The platform joins data integration, ETL validation, machine-learning services, SHAP [4], RAG [3], and specialist agents [5] into one workflow. Explanations and retrieval give complementary evidence, but neither guarantees correctness: an explanation can faithfully describe a flawed model, and a retrieved document can be outdated or irrelevant, so source information and uncertainty should stay visible.

Orchestration improves modularity but adds computation, latency, and room for inconsistent outputs, so it should be compared with simpler alternatives, and the manager's confidence should not be read as calibrated correctness until validated. Whether the design shortens decision time, improves stock availability, or raises performance remains an empirical question needing real users and outcomes.

VIII. LIMITATIONS AND THREATS TO VALIDITY

The study has several limitations. The platform covers one organization, so transfer to other ERP configurations and inventory policies is unproven. Reported metrics are development-stage and need temporal validation and leakage checks; the perfect inventory score in particular needs review of features, labels, duplicates, and splits. The sales module classifies direction and should not be treated as a regression forecaster without separate validation, and quantitative benchmarks for RAG and multi-agent quality are missing. Finally, no causal business improvement has been shown, and data quality, ERP availability, stale documents, model drift, and language-model variability may all affect reliability.

IX. FUTURE WORK

Next steps are a reproducible dataset description and leakage-safe chronological evaluation, an audit of inventory features and labels, and baseline comparisons for sales, with projections assessed separately. RAG should be tested on labelled questions with retrieval and answer-level metrics [3], and the multi-agent pipeline compared with direct and simpler

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

DOI: 10.17148/IJARCCE.2026.151121

agent baselines under identical conditions [5], with latency, failures, and caching measured. Later work may add human feedback, confidence calibration, drift monitoring, and audit trails, and a pilot with domain experts at Gokul Text Print would test usability and real gains.

X. CONCLUSION

We presented an integrated AI decision-support platform for Gokul Text Print that links ERPNext ingestion, validation-oriented ETL, sales and inventory machine learning, SHAP explanations [4], RAG retrieval [3], specialized agents [5], and a Flask dashboard in a modular, traceable design. The preliminary sales result is only an initial indication, and the perfect inventory score needs validation before it can be read as generalizable. Further empirical work must establish leakage-safe performance, grounded answer quality, robustness, and business impact; the system is meant to support human decisions while keeping uncertainty and the need for verification visible.

ACKNOWLEDGMENT

The authors acknowledge the institutional guidance and the operational context provided for the Gokul Text Print project. Specific acknowledgments for project supervision, organizational support, or funding should be included after the relevant details have been confirmed.

REFERENCES

[1]. S. Hurbean et al., "Unveiling AI-powered ERP systems: A systematic review of trends and future directions," *Information Systems and e-Business Management*, 2026, doi: 10.1007/s10257-026-00728-7.

[2]. M. Maghanaki, M. Shahin, and C. Chen, "Large language models in manufacturing: A comprehensive review," *The International Journal of Advanced Manufacturing Technology*, vol. 145, pp. 7601–7627, 2026, doi: 10.1007/s00170-026-18711-4.

[3]. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[4]. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5]. Q. Wu et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023.

[6]. Frappe Technologies, "REST API," *Frappe Framework Documentation*. [Online]. Available: https://docs.frappe.io/framework/user/en/api/rest

[7]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[8]. L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[9]. G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10]. A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

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