iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82698

www.ijraset.com

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

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

NEIL: Design of an AI-Driven Multi-Agent System for Accounts Payable Automation

Amaan A. Khan[1], D. P. Gaikwad[2]
[1, 2]*Department of Computer Engineering, AISSMS College of Engineering, Pune, Maharashtra, India*

Abstract: ***Accounts payable (AP) departments continue to rely on template-based optical character recognition tools, manual data entry with five-to-ten-per-cent error rates, and rule-based engines that scale poorly across vendors, languages, and tax jurisdictions. This paper presents the design of NEIL (Next-Generation Enterprise Invoice Learning), an AI-driven multi-agent platform built on a microservices architecture for end-to-end AP automation. The system separates concerns into two cooperating services: an Agent Service (Eddie) built on FastAPI and LangGraph that hosts specialised state-machine agents for document preprocessing, template-agnostic extraction using multimodal Large Language Models, validation with semantic matching, and tax-code mapping; and a Workflow Service (Neil) built on Django REST Framework, PostgreSQL, and Celery that handles multi-channel ingestion, invoice lifecycle, configurable approval routing, audit trails, and ERP integration. We describe the service decomposition, the inter-service REST contract, the Django app structure, the Celery task-queue topology, and the Kubernetes deployment posture that together enable straight-through processing. The architecture is positioned against the limitations of monolithic legacy AP platforms and is shown to support independent scalability, swappable AI providers, configuration-driven workflows, and observable agent state.***

Index Terms: ***Accounts payable automation, multi-agent systems, microservices architecture, multimodal large language models, LangGraph, FastAPI, Django, Celery, Kubernetes, intelligent document processing.***

I. INTRODUCTION

Invoice processing remains one of the most labour-intensive functions in modern enterprise finance. Despite three decades of investment in workflow software, manual AP operations still exhibit error rates in the five-to-ten-per-cent range, approval cycles of twelve to fifteen days, and a duplicate-payment incidence of approximately one per cent. Conventional deployments depend on template-based optical character recognition engines that require sustained per-vendor configuration and fail on unseen layouts; multi-jurisdictional tax handling — GST in India, VAT in the European Union, and WHT across Asian and African markets — adds further rule-engine complexity that rarely scales without significant manual maintenance.

In this paper we describe the design of the NEIL platform: a multi-agent system for AI-native AP automation built on a deliberately decomposed microservices architecture. The principal design decision is the separation of the platform into two cooperating services. An AI Agent Service hosts swappable, observable LangGraph state-machine agents. A Workflow Service hosts the enterprise-facing concerns: ingestion, approval, audit, and downstream ERP integration. The two services communicate through a typed REST contract, share a containerised platform, and can be scaled and evolved independently. The remainder of the paper reviews related work in document understanding and agentic workflows, describes the proposed system architecture, details the workflow-service implementation, and concludes with deployment considerations and future work.

II. RELATED WORK

Traditional AP platforms inherit a monolithic architecture in which extraction, validation, approval workflow, and ERP integration are tightly coupled into a single deployable. This coupling forces a single technology stack across the entire pipeline, limits the adoption of modern AI components, and complicates independent scaling under varying workloads. Recent work in document understanding has produced specialised models — Donut [1] for OCR-free extraction, LayoutLMv3 [2] for layout-aware encoding, and TrOCR [3] for transformer-based OCR — that each excel in narrow regimes but, taken alone, do not handle the diversity of real-world invoices. Multimodal foundation models such as the Qwen2-VL family [4] and the Llama 3 multimodal variants [5] substantially close this gap; the architecture described here is designed so that the underlying model can be swapped without redeploying business logic.

On the workflow side, enterprise AP systems are characterised by long-running asynchronous processes: ingestion from email, SFTP, and REST channels; staged validation; multi-step approval; and synchronous ERP posting. Such workloads are naturally event-driven and benefit from task-queue decoupling, durable retries, and per-queue scaling — a profile well served by Django and Celery. Validation and matching draw on a separate research tradition: Sentence-BERT [6] embeddings allow semantic matching of vendor names and purchase-order line items, FAISS [7] supports efficient large-scale similarity search, and declarative data-quality frameworks complement model-based validation. Agentic workflow research has matured rapidly: the LangGraph framework [8] models agent execution as an explicit state graph with named nodes and conditional edges, providing controllability and observability that earlier prompt-chaining frameworks lacked.

III. PROPOSED SYSTEM ARCHITECTURE

NEIL is composed of two cooperating microservices, deployed behind a shared API ingress and supported by a common data and infrastructure layer. The high-level decomposition is depicted in Fig. 1.

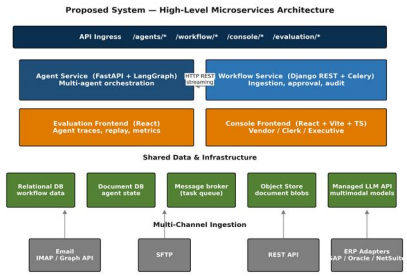


Fig. 1. NEIL high-level microservices architecture.

A. Agent Service (Eddie)

The Agent Service is implemented in FastAPI and exposes a suite of specialised LangGraph state-machine agents. A Document Preprocessing Agent performs format conversion, orientation correction, quality gating, and resolution normalisation. An Invoice Extraction Agent combines configurable OCR backends — cloud (Azure Document Intelligence, Google Vision) or local — with a multimodal LLM that operates under a schema-bound prompt covering more than one hundred structured invoice fields. A Validation Agent performs structural checks, N-way matching against purchase orders and goods receipt notes, and sentence-embedding-based semantic reconciliation. A Tax-Code Mapping Agent assigns VAT, WHT, and GST codes at both invoice and line-item levels. A Multi-Agent Supervisor routes between retrieval-augmented Q&A and natural-language-to-SQL data dialogue. Each agent exposes its state graph through the Eddie API, and inter-agent state is persisted to a document store that supports replay and audit.

B. Workflow Service (Neil)

The Workflow Service is implemented with Django REST Framework on PostgreSQL, with Celery as the asynchronous task substrate and a message broker for task distribution. Neil ingests documents from email (IMAP and Microsoft Graph), SFTP, and REST channels; manages the invoice lifecycle through extraction, validation, approval, and posting states; enforces role-based access control across Vendor and Workbench user types; and integrates with downstream ERP systems through client-specific adapter modules. Neil never calls a multimodal model directly — every AI step is delegated to Eddie over an authenticated REST contract, and all AI artefacts are persisted as Neil-side records so that the workflow remains auditable independent of Eddie's internal state.

C. Inter-service Communication

The two services communicate through a small set of well-defined endpoints. Neil calls Eddie at /eddie/neil/tax_code_assignment for VAT/WHT mapping, /eddie/neil/invoice_summary for role-aware summarisation, and the agent-specific endpoints for extraction, preprocessing, N-way matching, and semantic matching. Each endpoint is versioned, returns a typed response object, and is wrapped on the Neil side by a Celery task that supplies retry, timeout, and back-pressure semantics. This contract keeps Eddie stateless from Neil's perspective and lets either side be redeployed without coupling.

IV. IMPLEMENTATION DETAILS

A. Django App Structure

Neil is organised into modular Django applications, each owning a bounded responsibility. The Documents app manages the core invoice lifecycle, exposing models for Document, Invoice, Extraction, and the four status streams. The Accounts app handles Vendor and Workbench user types with role-based access control. The Channels app integrates external ingestion sources and routes documents based on header metadata. The History app, built on django-simple-history, provides a per-field audit trail. The Admin-Settings app exposes configuration for custom field definitions, validation rules, and approval workflows, allowing operational changes without code deployment. The Chat app implements an AI-powered, session-based conversational interface by streaming responses from Eddie's agent API.

B. Celery Task Topology

Celery provides event-driven, asynchronous processing across several dedicated queues. The document-preprocessing queue handles initial format conversion and quality gating via Eddie. The invoice-validation queue orchestrates the full structural-and-N-way-matching sequence. The tax-computing queue manages asynchronous VAT/WHT calls to Eddie. Separate queues serve invoice translation for multilingual support and outbound notification dispatch through Outlook and Gmail. Each queue can be scaled horizontally and independently, isolating slow LLM calls from time-critical user-facing tasks.

C. Document Processing Pipeline

The complete processing pipeline proceeds in eight stages: document upload, document preprocessing, quality check, invoice extraction, tax-code assignment, validation, approval workflow, and ERP posting. Validated invoices auto-approve under straight-through processing; failed invoices route to a clerk via the Workbench console. ERP posting is performed by a client-specific adapter against the downstream system of record.

D. Deployment Architecture

Deployment is on Kubernetes, with separate services for the Eddie backend, the Neil backend, the React-based Workbench frontend, the Celery worker pools, and the PostgreSQL and document-store databases. An Nginx Ingress routes /eddie/* to the Agent Service, /neil/* to the Workflow Service, and /workbench/* to the Workbench frontend. Persistent volumes back the relational and document databases; object storage hosts the original document blobs and intermediate artefacts. Horizontal pod autoscaling responds to queue depth on the Celery worker pools, allowing the AI-heavy preprocessing and extraction stages to scale independently of the workflow-heavy validation and approval stages.

V. CONCLUSION AND FUTURE WORK

This paper has described the design of NEIL, a microservices-based platform for AI-driven accounts payable automation. The key architectural choice is the deliberate separation between an Agent Service that hosts swappable LangGraph-based AI components and a Workflow Service that owns enterprise-facing lifecycle, audit, and ERP-integration responsibilities. The two services communicate through a small typed REST contract, are scaled independently on Kubernetes, and are designed so that either side can be evolved without disrupting the other.

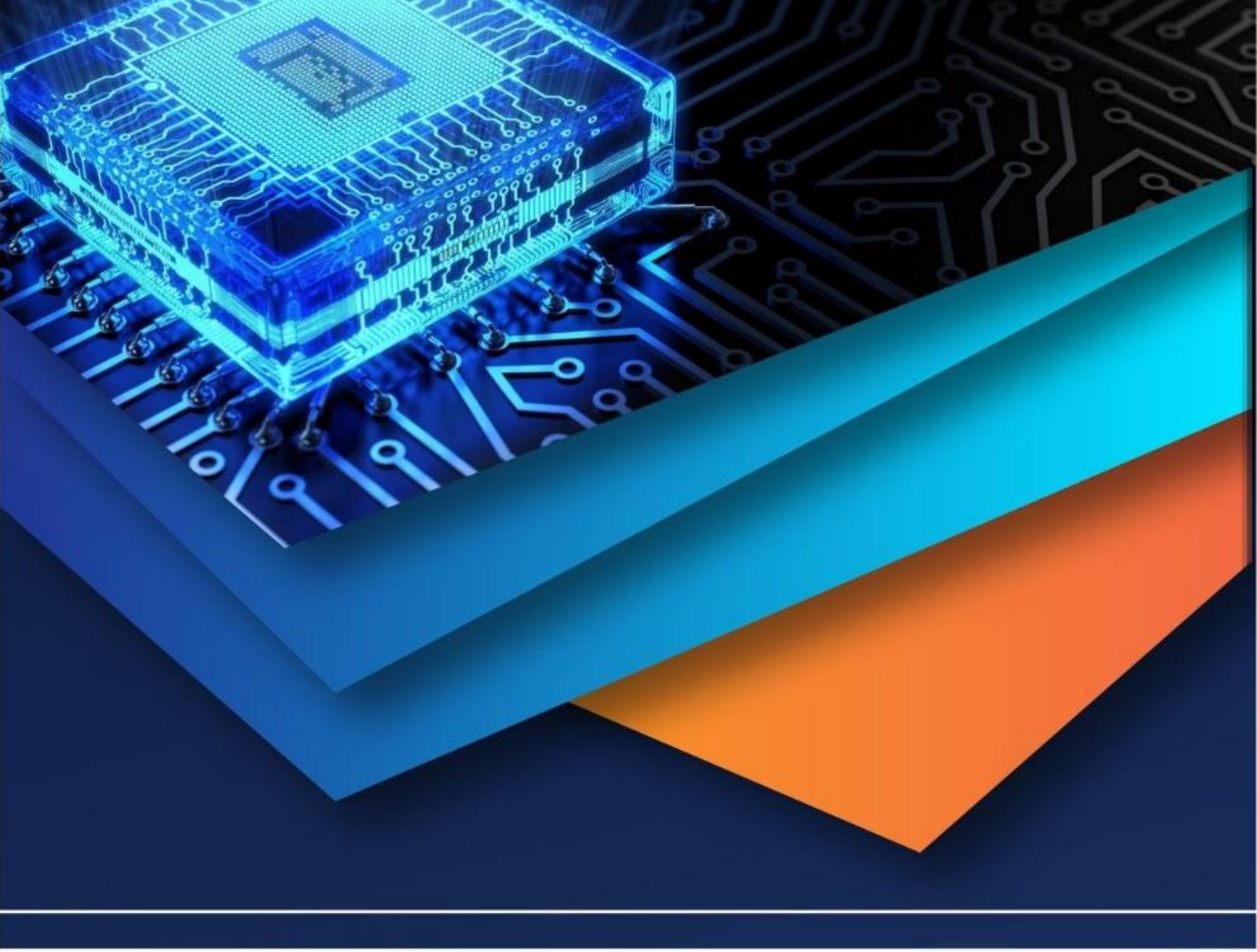
Future work proceeds in three directions. First, the integration of parameter-efficient fine-tuning workflows (LoRA [9], QLoRA [10]) into the Eddie deployment lifecycle for per-tenant adaptation of the extraction model. Second, the addition of an ensemble document-classifier (ResNet-50 [11], MobileNetV2 [12]) at the ingestion layer to triage incoming documents and route supporting documents away from the full extraction pipeline. Third, the construction of an externally-curated invoice-extraction benchmark drawn from anonymised production data, allowing direct empirical comparison between the proposed architecture and academic baselines under per-vendor and per-language reporting.

VI. ACKNOWLEDGEMENT

The first author thanks Dr. S. V. Athawale, Head of the Department of Computer Engineering, AISSMS College of Engineering, Pune, for institutional support throughout the M.E. programme.

REFERENCES

[1] G. Kim et al., "OCR-free Document Understanding Transformer (Donut)," in Proc. ECCV, 2022, pp. 498–517.
[2] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, "LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking," in Proc. ACM MM, 2022, pp. 4083–4091.
[3] M. Li et al., "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," in Proc. AAAI, 2023, pp. 13094–13102.
[4] Qwen Team, "Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution," arXiv:2409.12191, 2024.
[5] Meta AI, "The Llama 3 Herd of Models," arXiv:2407.21783, 2024.
[6] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. EMNLP, 2019, pp. 3982–3992.
[7] J. Johnson, M. Douze, and H. Jégou, "Billion-scale Similarity Search with GPUs," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2021.
[8] LangChain, "LangGraph: Building Stateful, Multi-Actor Applications with LLMs," 2024. [Online]. Available: https://langchain-ai.github.io/langgraph/
[9] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," in Proc. ICLR, 2022.
[10] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," in Proc. NeurIPS, 2023.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE CVPR, 2016, pp. 770–778.
[12] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. IEEE CVPR, 2018, pp. 4510–4520.
[13] Institute of Finance & Management (IOFM), "AP Department Benchmarks and Analysis," annual industry survey, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY