International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 3, May-June 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

FREELANCER PRODUCTIVITY AGENT: A Modular Multi-Agent Framework for Freelancer Workflow Support

Dr. Umesh D R
Meghana M, Kushali K M, Lavanya K, Prajwal M N
Professor | umesh.dr.pesce@gmail.com
meghanaop360@gmail.com | kushkm333@gmail.com | 23lavanyak@gmail.com | mnprajwal004@gmail.com
Department of Computer Science and Engineering (AI & ML)
P.E.S College of Engineering, Mandya, Karnataka, India

Abstract:
Freelancers often lose time to the small administrative tasks around the actual job. This paper presents the Freelancer Productivity Agent (FPA), a modular multi-agent framework designed to reduce that overhead through task-specific agents coordinated by a LangGraph orchestrator. The system separates proposal drafting, communication handling, scheduling, knowledge retrieval, invoice processing, and risk screening into independent services exposed through FastAPI.

Keywords: multi-agent systems, freelancer support, retrieval-augmented generation, OCR, document understanding, semantic search, workflow orchestration, FastAPI

I. INTRODUCTION

Independent freelancers often work with a fragmented toolchain. A single project may require one application for client messages, another for proposal drafting, a calendar for deadlines, and a separate system for invoicing. None of these tools usually share context in a meaningful way, so time is spent rebuilding information that already exists somewhere else.

This overhead is easy to ignore because each task appears minor on its own. In practice, however, the repeated interruptions accumulate. A freelancer may read a job post, evaluate whether it is worth pursuing, prepare a tailored proposal, answer follow-up emails, record the timeline, and later process payment reminders. The work is not difficult in isolation, but it is repetitive and context-heavy.

Recent progress in large language models, retrieval systems, and orchestration frameworks has made it practical to automate parts of this workflow. A multi-agent design is especially suitable because different tasks require different forms of reasoning. Proposal writing relies on prior examples, email handling needs intent recognition, invoice processing depends on document structure, and risk screening depends on heuristics plus language understanding.

This paper presents FPA, a modular system designed around these distinctions. Each agent handles one task family, and the orchestrator decides when one agent's output should become another agent's input. The goal is not to replace the freelancer's judgment, but to reduce the routine work that surrounds the actual project.

The main contributions of this paper are a workflow-oriented multi-agent architecture for freelancer support, a proposal generation pipeline grounded in the freelancer's own document history, an invoice extraction module using OCR and document-layout modeling, a hybrid risk screening module for suspicious job postings, and an evaluation of the individual agents and the integrated workflow.

II. RELATED WORK

Tool-augmented language models have shown that LLMs can learn to call external utilities during

generation. Toolformer is one of the clearest examples of this direction, and it helps motivate systems where the model is not isolated from tools but actively uses them.

In parallel, CAMEL explored communication among multiple agents through explicit roles, which is useful when a system needs different components to cooperate rather than act independently. Survey work on autonomous LLM agents also highlights memory, planning, and tool use as central themes.

Document understanding has advanced quickly as well. LayoutLMv3 showed that text and page structure should be treated together when extracting information from forms and invoices. That matters because a document is not only a list of words; placement, spacing, and alignment often carry meaning.

Work on marketplace fraud detection is relevant too, especially for gig platforms where a worker must judge whether a posting is credible before spending time on it. Most existing approaches stay at the classification level and do not connect that result to the rest of the workflow.

Commercial tools such as Gmail Smart Reply, Notion AI, and scheduling assistants are useful, but each one solves only a slice of the workflow. They are helpers, not a coordinated system. That gap is what the proposed framework tries to address.

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

III. SYSTEM DESIGN

The system is divided into four layers.

A. Agent Layer: The agent layer contains six services, each responsible for a narrow task family. Narrow scope was deliberate, because when a module does too much, debugging becomes harder and improvements become riskier.

B. Orchestration Layer: The orchestrator decides when an input should be routed, paused, or stopped. It also manages the order in which outputs are shared across agents. This layer turns separate services into one workflow.

C. Storage Layer: The storage layer uses three databases for different types of data. PostgreSQL stores structured records such as projects, invoice entries, and audit logs. MongoDB stores transient agent state and execution traces. ChromaDB stores embeddings for semantic retrieval.

D. Interface Layer: The dashboard gives the user one place to see the result of the workflow. Proposal drafts, email summaries, reminders, extracted invoice fields, and risk warnings are all shown in the same interface.

TABLE I
COMPARISON OF EXISTING SYSTEMS WITH FPA

System	Proposal Support	Email Support	Finance / Risk
Notion AI	Partial	No	No
Gmail Smart Reply	No	Partial	No
Upwork UMA	Yes	No	No
Motion / Reclaim AI	No	No	No
FPA (Proposed)	Yes	Yes	Yes

One issue that appeared during integration was inconsistent state after asynchronous requests. Some calls returned correctly but reached the dashboard too late or in the wrong order. The fix was centralized state updates, explicit retries, and stricter response validation.

IV. METHOD

A. Workflow Overview: The workflow is event-driven. A request enters the system, the orchestrator classifies it, and the relevant agent is called. The output of one step may become the input of the next step.

For a new job post, the risk agent runs first. If the post looks unsafe, the workflow stops early and the user is warned. If the post is acceptable, the proposal agent retrieves earlier examples and drafts a proposal. The scheduling agent then extracts due dates and places work blocks on the calendar. The communication agent prepares a reply draft. The dashboard is refreshed after the chain is complete.

B. Proposal Generation: The ProposalAgent uses retrieval-augmented generation to support proposal writing. Past proposals, notes, and reusable templates are broken into chunks and embedded using text-embedding-3-small. When a new opportunity arrives, the system retrieves similar chunks from ChromaDB and inserts them into the prompt.

This matters because proposal writing is not only about wording. It is also about remembering how the freelancer has handled similar work in the past. During testing, the retrieval stage sometimes

returned repeated passages. A diversity filter was added so that the final prompt did not contain too many near-identical chunks.

C. Email Handling: The CommunicationAgent reads incoming messages and assigns them to one of six groups: new inquiry, revision request, payment follow-up, deadline extension, clarification request, or out-of-scope message. After classification, it prepares a short summary and a draft response.

The most frequent confusion occurred with short, informal messages. Revision requests and clarification requests often looked similar, so the prompt had to be tightened. Adding explicit priority labels improved the drafts because the model received a clearer sense of what mattered most.

D. Scheduling: The SchedulingAgent works with deadlines and milestone dates. It extracts date expressions from messages, checks the calendar, and looks for overlap before adding new blocks.

Date parsing was one of the more practical problems in the system. Human-written messages often use relative phrases such as 'next Friday' or 'by the weekend', which are easy for people but not always consistent for software. Validation logic and fallback parsing were added to make the schedule output more stable.

E. Knowledge Retrieval: The KnowledgeAgent acts as the shared memory of the framework. It does not produce the final content itself. Instead, it supplies relevant context to the agents that need it.

Keeping retrieval separate from generation made the system easier to test and easier to trace when something went wrong.

F. Invoice Processing: The FinanceAgent handles invoice PDFs in two stages. Tesseract OCR first extracts text and basic layout cues. LayoutLMv3 then classifies tokens into fields such as invoice number, vendor name, amount due, date, and line items.

The main weak point was document quality. Low-resolution scans, skew, and compression artifacts reduced OCR accuracy. Simple preprocessing steps such as sharpening and thresholding improved the output enough for the extraction pipeline to remain usable.

G. Risk Screening: The RiskDetectionAgent checks job posts for suspicious patterns. The first stage uses simple rules such as vague deliverables, unrealistic payment claims, and requests for advance purchases.

The second stage uses language-model reasoning to produce a score and explanation. The system was tuned to stay conservative, so some legitimate posts with rough wording were scored higher than expected. A configurable threshold lets the user choose how strict the filter should be.

V. IMPLEMENTATION

The backend services were implemented in Python 3.11 using FastAPI. LangGraph was used for orchestration, and LangChain supported prompt assembly and tool calls. The frontend was built in React.js with Bootstrap 5.

TABLE II
Technology Stack

Component	Technology	Purpose
Orchestration	LangGraph / LangChain	Workflow control
Backend	FastAPI	Agent APIs
LLMs	GPT-4o, Gemini Pro, LLaMA-3	Generation and classification
Retrieval	ChromaDB + text-embedding-3-small	Semantic memory
Invoice Processing	LayoutLMv3 + Tesseract OCR	Field extraction
APIs	Gmail API, Google Calendar API	Email and scheduling
Databases	PostgreSQL, MongoDB, ChromaDB	Persistent storage
Frontend	React.js + Bootstrap 5	Dashboard

VI. EXPERIMENTAL SETUP

The agents were evaluated separately before the full workflow was tested. The ProposalAgent was tested on 120 job descriptions from web development, technical writing, graphic design, data analysis, and digital marketing.

The CommunicationAgent was tested on 150 synthetic emails. The FinanceAgent was tested on 80 SROIE invoices and 40 custom invoices. The RiskDetectionAgent was tested on 200 labeled job posts, split evenly between legitimate and fraudulent examples.

TABLE III
KPI Comparison

KPI	Manual Workflow	FPA
Proposal drafting time	35-60 min	2-4 min
Invoice entry time	8-12 min	< 1 min
Email intent classification	Manual	91.3% accuracy
Fraud detection	Subjective	0.893 precision, 0.871 recall
Scheduling conflicts	Frequent	Reduced

VII. RESULTS AND DISCUSSION

The system reduced repetitive administrative work across the tested task areas. Proposal generation performed better when retrieval context was available. The RAG-based outputs received mean scores of 4.3 for relevance, 4.5 for professionalism, and 4.2 for completeness.

Zero-shot generation was still functional, but it was less specific and less tied to the freelancer's own history. The FinanceAgent reached an F1 score of 0.914 on SROIE and 0.871 on the custom invoice set. Most of the remaining errors came from OCR quality rather than field classification.

The RiskDetectionAgent reached precision of 0.893, recall of 0.871, and F1 of 0.882. Fraud cases were missed mainly when warning signs were subtle, while false positives were more common when legitimate clients used vague or casual language.

The CommunicationAgent reached 91.3% accuracy in intent classification. The biggest confusion remained between revision requests and clarification requests, which is expected because real client messages often blur that boundary.

TABLE IV
Agent-Level Performance Summary

Agent	Metric	Result
ProposalAgent	Relevance with RAG	4.3 / 5.0
CommunicationAgent	Intent accuracy	91.3%
FinanceAgent	SROIE F1	0.914
FinanceAgent	Custom invoice F1	0.871
RiskDetectionAgent	Precision / Recall / F1	0.893 / 0.871 / 0.882
Full System	End-to-end runtime	< 8 s/job

VIII. CONCLUSION

FPA was proposed as a modular multi-agent framework for the work that surrounds freelancer projects. The system combines proposal drafting, email handling, scheduling, invoice extraction, semantic retrieval, and risk screening in one connected workflow.

The most important result is not that the system automates everything, but that it reduces the constant small interruptions that make freelancing harder than it needs to be. The evaluation showed that the agents can handle their own tasks and still cooperate as one pipeline.

Several limitations remain. Invoice extraction still depends on scan quality, and risk detection is sensitive to the style of the posting. Future work will focus on stronger date parsing, live platform integration, payment reminders, and better adaptation to individual user behavior.

IX. ACKNOWLEDGMENT

The authors thank the contributors who developed and maintained the IEEE LaTeX style files that inspired this template.

X. REFERENCES

[1] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," Proc. NeurIPS, vol. 36, 2023.

[2] G. Li et al., "CAMEL: Communicative agents for mind exploration of large language model society," Proc. NeurIPS, vol. 36, 2023.

[3] Z. Xi et al., "The rise and potential of large language model based agents: A survey," arXiv preprint arXiv:2309.07864, 2023.

[4] Y. Xu et al., "LayoutLMv3: Pre-training for document AI with unified text and image masking," Proc. ACM Multimedia, 2022.

[5] M. Gupta et al., "LLM-aided scam detection in gig economy platforms," arXiv preprint arXiv:2406.03178, 2024.

[6] Upwork, "Introducing UMA, Upwork's AI-powered assistant," 2024.

[7] H. Zhang et al., "LLM-based job recommendation in online freelancing marketplaces," arXiv preprint arXiv:2402.01240, 2024.

[8] X. Wu et al., "GraphDocNet: Graph neural networks for structured document information extraction," Pattern Recognition Letters, vol. 172, pp. 59-66, 2024.

[9] Y. Chen et al., "GraphLLM-Fraud: Hybrid graph neural network and language model for fraud detection," arXiv preprint arXiv:2501.00888, 2025.

[10] LangChain Documentation, "LangGraph: Stateful workflow framework for LLM applications."

[11] FastAPI Documentation, "FastAPI framework for Python APIs."

[12] A. Kannan et al., "Smart Reply: Automated response suggestion for email," Proc. ACM SIGKDD, 2016.

[13] L. Wang et al., "Survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, 2024.

