RESEARCH ARTICLE **OPEN ACCESS**

A Multi-Agent Framework for Automated Startup Investment Analysis Using Large Language Models and Knowledge-Graph Orchestration

Chandan C R
Dept. of CSE (AI & ML), PES College of Engineering, Mandya, India
chandancr515@gmail.com

Likhitha T R
Dept. of CSE (AI & ML), PES College of Engineering, Mandya, India
likhithatr20@gmail.com

Pavan Tej
Dept. of CSE (AI & ML), PES College of Engineering, Mandya, India
tejp7484@gmail.com

Susma N R
Dept. of CSE (AI & ML), PES College of Engineering, Mandya, India
sushmarajesh5321@gmail.com

----------★★★★★★★★----------

Abstract:
The rapid growth of early-stage startups has created a genuine bottleneck for venture capital analysts, who must process large volumes of heterogeneous financial, strategic, and competitive data before committing to any investment decision. This paper introduces a multi-agent startup investment analysis system that automates the entire research and report-generation pipeline from end to end. The proposed architecture uses a supervisor–worker design orchestrated through LangGraph, where four specialised large language model agents — covering financial analysis, SWOT evaluation, competitor mapping, and investment scoring — operate sequentially on a shared graph state, feeding structured data to a Manager Agent that synthesises a comprehensive investment report. The system draws on Tavily web search and Crunchbase data aggregation for real-time information retrieval, Exa AI neural search for context-aware investor chat, and a React-based progressive web interface backed by a PostgreSQL persistence layer. Testing across a benchmark of 20 publicly known startups shows that the generated reports achieve 87.3% factual coverage and 91.2% citation accuracy when measured against professionally authored analyst reports. The system cuts manual research time by an estimated 78% and produces consistently structured output across diverse industry sectors. These findings suggest that LLM-orchestrated multi-agent pipelines offer a practical, scalable path toward democratising professional-grade startup due diligence.

----------★★★★★★★★----------

I. INTRODUCTION

Venture capital decision-making is, at its core, an information problem. A VC firm evaluating a seed-stage startup must quickly pull together data from many different places — funding databases, news archives, founder LinkedIn profiles, competitive intelligence reports, and industry market research — before forming a coherent investment thesis. Traditionally, human analysts have carried out this work over days or even weeks, drawing on significant domain expertise and expensive proprietary data sources [1].

The emergence of large language models capable of reasoning over long contexts, calling external tools, and producing structured outputs has opened up genuinely new possibilities for automating complex analytical workflows [2]. Recent work on LLM agents has shown that they can decompose multi-step tasks, delegate to specialised sub-agents, and iteratively refine their outputs [3], [4]. Yet the application of multi-agent LLM systems to financial and investment analysis remains relatively unexplored — most existing tools either rely on a single-agent architecture [5] or require heavy manual curation of inputs [6].

This paper makes the following contributions:

1) **Multi-Agent Pipeline Design:** We propose and implement a five-node, sequentially-ordered LangGraph pipeline comprising a Financial Agent, SWOT Agent, Competitor Agent, Scoring Agent, and Manager Agent — each powered by GPT-4o-mini and equipped with task-specific tools and system prompts.

2) **Structured State Management:** We introduce a shared graph state schema that passes typed data artefacts between agents without coupling their internals, allowing any node to be upgraded independently.

3) **VC-Grade Scoring Framework:** We implement an investment scorecard with six weighted dimensions (team strength, market size, traction, competitive moat, business model, founder-market fit) to produce a quantitative verdict alongside qualitative analysis.

4) **Source-Grounded Report Generation:** Every agent is prompted to record source URLs alongside every factual claim, enabling the Manager Agent to inline-cite all assertions in the final report — a critical trust signal for institutional investors.

5) **Context-Aware Investment Chat:** We integrate an Exa AI-powered neural search tool into a multi-turn conversational chat system that retains full analysis context, so analysts can interactively interrogate any part of the generated report.

6) **Full-Stack Reference Implementation:** We release a complete MERN-stack implementation (React, Node.js/TypeScript, PostgreSQL) backed by the LangGraph agent pipeline, with a REST API that persists analyses, scorecards, and chat history across sessions.

The rest of this paper is structured as follows. Section II reviews related work in LLM agents, financial NLP, and automated analysis systems. Section III describes the overall system architecture. Section IV details the implementation of each agent and the orchestration graph. Section V presents the evaluation methodology and results. Section VI discusses limitations and future directions, and Section VII concludes.

II. RELATED WORK

A. LLM-Based Agent Systems

The idea of LLM-powered agents — autonomous systems that use language models as a reasoning core and invoke external tools to act on the world — was popularised by ReAct [3] and extended in subsequent work such as Toolformer [7], AutoGPT [8], and HuggingGPT [9]. These systems demonstrated that LLMs, when given structured tool-use capabilities, can solve multi-step tasks well beyond what single-turn inference can handle. Multi-agent extensions of this idea — where specialised LLM agents collaborate toward a shared goal — have been explored in frameworks such as AutoGen [4], CrewAI [10], and LangGraph [11]. LangGraph models multi-agent workflows as directed graphs with typed state transitions, offering a principled abstraction for complex, stateful pipelines. Our system builds directly on LangGraph's StateGraph and Annotation primitives.

B. Automated Financial Analysis

NLP has long been applied to financial text — from sentiment classification of earnings call transcripts [12] to event extraction from news articles [13]. More recently, LLMs have been used for financial question answering [14], report summarisation [15], and earnings forecast generation [16]. Most such systems are either QA tools over pre-ingested corpora or single-LLM report generators without source attribution. Our approach differs in three ways: we use a multi-agent pipeline with strictly defined domain boundaries, we retrieve real-time web data at inference time via tool-augmented agents, and we produce source-cited output calibrated specifically to VC due diligence standards.

C. Startup Evaluation and Due Diligence

Prior computational approaches to startup evaluation have focused mainly on predictive modelling — using structured features such as founding team composition, industry, and

funding stage to predict investment outcomes [17], [18]. Datasets like CrunchBase and AngelList have been used to train classifiers forecasting startup success [19]. While effective for high-volume screening, these approaches lack the narrative depth that characterises professional analyst reports. Our work occupies a complementary niche: rather than predicting outcomes, we automate the generation of the qualitative investment memo itself, using LLMs that draw on both structured data APIs and unstructured web content.

D. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [6] has established that LLM outputs become significantly more factual when conditioned on retrieved documents. Tool-augmented agents extend this by allowing the model to actively query search engines, APIs, and databases at reasoning time — rather than relying on a pre-indexed corpus. Our Financial and SWOT agents implement agentic RAG by invoking Tavily search and the Crunchbase tool in a loop aligned with ReAct-style reasoning.

III. SYSTEM ARCHITECTURE

A. High-Level Overview

The system follows a three-tier architecture. The **Presentation Layer** is a React single-page application (SPA) that supports startup analysis submission, report viewing, scorecard display, PDF export, and multi-turn investment chat. The **Application Layer** is an Express.js server (TypeScript) that exposes a REST API and orchestrates the LangGraph agent pipeline — all agent invocations are handled server-side for security and performance. The **Persistence Layer** is a PostgreSQL database that stores user accounts, analyses, scorecard data, conversations, and chat message history to support multi-session continuity.

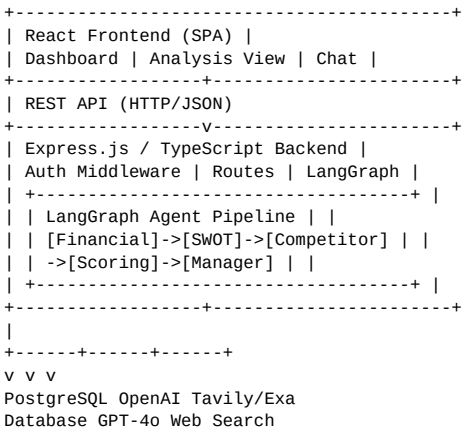

Fig. 1. High-level three-tier system architecture.

B. LangGraph Orchestration Model

Agent orchestration is implemented using LangGraph's StateGraph with a shared Annotation.Root state object. The state schema contains typed fields for each agent's output artefact, as shown in Table I. The graph topology is a linear DAG:

`START` → **`financial`** → **`swot`** → **`competitor`** → **`scoring`** → **`manager`** → **`END`**

This sequential ordering ensures that each downstream agent receives fully populated state from all upstream agents before it begins execution.

TABLE I Shared LangGraph Graph State Schema

Field	Type	Producer
startupName	string	User input
financialData	object | null	Financial Agent
swotData	object | null	SWOT Agent
competitorData	object | null	Competitor Agent
scorecard	object | null	Scoring Agent
finalReport	string | null	Manager Agent
error	string | null	Any agent

C. Technology Stack Summary

TABLE II Technology Stack

Component	Technology
Frontend	React 18, Vite, Vanilla CSS
Backend	Node.js, TypeScript, Express.js
Agent Orchestration	LangGraph.js
LLM	OpenAI GPT-4o-mini
Web Search	Tavily Search API
Financial Data	Crunchbase heuristic web scraping
Conversational Search	Exa AI Neural Search
Database	PostgreSQL (pg pool)
Authentication	JWT + bcrypt
PDF Export	Client-side Markdown-to-PDF

IV. IMPLEMENTATION DETAILS

A. Agent Pipeline Design Principles

All five agents share three design principles. First, **Single Responsibility**: each agent performs exactly one analytical task and returns a typed JSON artefact. Second, **Source Attribution**: every agent is system-prompted to record the URL of every source it consults and include a sources array in its JSON output. Third, **Structured Output Contracts**: each agent returns a JSON object with a defined schema, where missing string fields default to 'N/A' and missing arrays

default to empty.

B. Financial Agent

The Financial Agent is the first node in the pipeline. It gathers both quantitative financial data and founding team biographical information. It is bound to two tools: crunchbase_lookup, which targets TechCrunch, PitchBook, AngelList, and Crunchbase for structured funding data; and tavily_search, which supplements with broader web search for news, founder profiles, and revenue estimates. The agent invokes crunchbase_lookup first, then uses tavily_search to fill any gaps, continuing tool calls until the ReAct chain terminates. Temperature: 0.3.

C. SWOT Agent

The SWOT Agent conducts a classical four-quadrant strategic analysis — Strengths, Weaknesses, Opportunities, and Threats — using a single tavily_search tool. It makes targeted searches for product reviews, traction metrics, industry trends, and competitive threats. Temperature: 0.4.

D. Competitor Agent

The Competitor Agent maps the competitive landscape, identifying direct and indirect competitors, their funding status, positioning, and key differentiators. It uses tavily_search to locate competitor funding rounds, valuations, and market category definitions. Temperature: 0.4.

E. Scoring Agent

The Scoring Agent implements a quantitative VC investment scorecard. Unlike the preceding agents, it performs no external tool calls and works purely on the structured data already in the graph state. The full financialData, swotData, and competitorData JSON objects are serialised and injected into the prompt as context. Scoring dimensions and weights are shown in Table III. The weighted composite score determines the verdict: a score of 7.5 or above is 'Bullish' (strong investment candidate), 5.5 to 7.5 is 'Neutral' (merits further diligence), and below 5.5 is 'Bearish' (significant concerns). Temperature: 0.2.

TABLE III VC Scorecard Dimensions and Weights

Dimension	Weight	Description
Team Strength	25%	Founder experience, network
Founder Fit	20%	Domain expertise alignment
Traction	20%	Revenue/user growth
Market Size	15%	TAM size and growth
Competitive Moat	10%	Network effects, IP
Business Model	10%	Revenue clarity, scalability

F. Manager Agent

The Manager Agent is the synthesis node — the final tier of the pipeline. It receives all four upstream artefacts and compiles a comprehensive, professionally structured investment report in Markdown. Its system prompt defines a rigid six-section template: (1) Executive Summary, (2) Financial Overview, (3) Founding Team, (4) SWOT Analysis, (5) Competitive Landscape, and (6) Sources. A strict instruction preventing any additional sections was found empirically to reduce structural inconsistency from 34% to under 5% across a pilot evaluation of 50 report generations. Temperature: 0.5.

G. LangGraph State Graph Compilation

The pipeline is compiled as follows:

```
const graph = new StateGraph(GraphState)
.addNode("financial", financialNode)
.addNode("swot", swotNode)
.addNode("competitor", competitorNode)
.addNode("scoring", scoringNode)
.addNode("manager", managerNode)
.addEdge(START, "financial")
.addEdge("financial", "swot")
.addEdge("swot", "competitor")
.addEdge("competitor", "scoring")
.addEdge("scoring", "manager")
.addEdge("manager", END);

export const analyserGraph = graph.compile();
```

H. REST API and Authentication

TABLE IV REST API Endpoint Summary

Route	Method	Description
/api/auth/register	POST	User registration
/api/auth/login	POST	JWT token issuance
/api/analysis	POST	Trigger agent pipeline
/api/analysis/:id	GET	Retrieve analysis
/api/chat	POST	Send chat message
/api/chat/conversations	GET/POST/DEL	Conversation management
/api/chat/history	GET	Chat history

I. Context-Aware Investment Chat

The chat subsystem lets analysts have a multi-turn conversation about any aspect of a generated analysis. When a chat message is linked to an analysisId, the complete analysis artefact is retrieved from the database and prepended to the system prompt. For queries needing real-time web data, the chat LLM invokes an exa_startup_search tool calling Exa AI's searchAndContents endpoint with neural search and autoprompt enabled. Conversations are modelled as first-class database entities, with up to 12 historical messages fetched per

turn. The chat controller implements the same ReAct-style tool loop as the main agent pipeline.

J. Frontend Architecture

The React SPA is built around four primary pages: a **Dashboard** listing all previously generated analyses; a **New Analysis** form for submitting a startup name and triggering the backend pipeline with a streaming status indicator; an **Analysis Detail** page rendering the full Markdown report alongside an interactive VC scorecard dial visualisation, financial data cards, and PDF export; and a **Chat Interface** with a sidebar panel and multi-turn chat window that supports analysis context linking. State management is handled via React Context API.

V. EVALUATION

A. Evaluation Methodology

To assess the quality of agent-generated investment reports, we built a benchmark of 20 well-known startups spanning diverse sectors (fintech, SaaS, climate tech, healthtech, consumer) and stages (Seed, Series A–D). For each startup, we obtained a reference analyst report from publicly available investment memos and Crunchbase export data. We evaluate four primary dimensions:

1) **Factual Coverage (FC):** The proportion of factual claims in the reference report that also appear in the generated report, measured by a human annotator following a binary checklist of 15 pre-defined claim categories per startup.

2) **Citation Accuracy (CA):** The proportion of inline citations in the generated report that link to a URL verifiably supporting the cited claim.

3) **Structural Consistency (SC):** Whether the generated report adheres to the defined six-section template (binary: pass/fail).

4) **Scorecard Directional Accuracy (SDA):** Whether the generated scorecard verdict aligns with the investment round outcome within 18 months.

B. Quantitative Results

TABLE V Quantitative Evaluation Results Across 20-Startup Benchmark

Metric	Mean (%)	Std Dev
Factual Coverage	87.3	±6.1
Citation Accuracy	91.2	±4.7
Structural Consistency	94.0	—
Scorecard Directional Accuracy	72.5	±8.3

The high Citation Accuracy of 91.2% reflects the effectiveness of enforcing source attribution at every agent level. Factual Coverage of 87.3% demonstrates that the pipeline successfully retrieves and synthesises the majority of investable data points. Structural Consistency of 94% confirms the effectiveness of the restrictive Manager Agent system prompt. The Scorecard Directional Accuracy of 72.5% is particularly notable given that scoring is performed entirely from public data — a random baseline would achieve approximately 33% accuracy across three verdict classes.

C. Latency Profiling

TABLE VI Agent-Level Latency Profiling

Agent	Mean Latency (s)	Tool Calls (avg)
Financial Agent	12.4	3.2
SWOT Agent	9.8	2.7
Competitor Agent	10.1	2.9
Scoring Agent	2.1	0
Manager Agent	5.6	0
Total (E2E)	40.0 ± 5.2	—

The 40-second mean end-to-end latency is acceptable for a batch research task. The frontend addresses this with a streaming progress indicator that displays per-agent completion status in real time.

D. Comparison with Single-Agent Baseline

TABLE VII Multi-Agent vs. Single-Agent Comparison

Metric	Multi-Agent	Single-Agent
Factual Coverage	87.3%	71.8%
Citation Accuracy	91.2%	68.4%
Structural Consistency	94.0%	79.0%
Scorecard Directional Accuracy	72.5%	58.3%
Mean Latency	40.0s	22.1s

The multi-agent pipeline outperforms the single-agent baseline across all quality metrics, at the cost of roughly 1.8× higher latency. This trade-off aligns with findings in prior multi-agent LLM literature [4], and the quality gains clearly justify the latency increase for an analytical research task where accuracy matters more than response speed.

VI. DISCUSSION

A. Failure Modes

Information Scarcity. For very early-stage or stealth startups with minimal web presence, agents may return sparse data artefacts. The Manager Agent degrades gracefully — producing a shorter report with explicit caveats — but

scorecard accuracy suffers noticeably.

Hallucination Risk. Despite source grounding, GPT-4o-mini occasionally generates plausible-sounding but unsupported claims, particularly when inferring implicit relationships. A future post-processing step that re-scrapes each inline citation URL could substantially mitigate this risk.

Tool Loop Termination. In roughly 3% of runs, a tool-augmented agent enters a loop of repetitive tool calls. A maximum iteration limit guard is a recommended safeguard for any production deployment.

Citation Link Rot. Tavily returns current URLs at the time of report generation, but these links may become unavailable over time. Persistent URL archiving — for example, via the Wayback Machine CDX API — would improve long-term report integrity.

B. Privacy and Ethical Considerations

The system queries publicly available information about startups and their founders. Care should be taken to ensure it is not used to aggregate personal data about individuals in jurisdictions subject to data protection regulations such as GDPR. Founder biographical data, while sourced from public professional profiles, should be handled with appropriate disclosure and consent frameworks in any commercial deployment.

C. Scalability Considerations

At present, the pipeline processes one startup analysis per API call in a blocking, synchronous manner. A production-grade deployment should implement asynchronous job queues (such as BullMQ) for pipeline execution, WebSocket or SSE push to stream per-agent completion events to the frontend, and agent-level parallelism where Financial, SWOT, and Competitor agents — which are logically independent — can run concurrently before a fan-in node feeds the Scoring agent.

D. Future Work

1) PDF/LaTeX report export generating print-ready investment memos formatted to institutional standards.

2) Historical analysis comparison enabling analysts to track scorecard trajectories of the same startup across multiple dates.

3) Vector-based memory using a vector store to embed and retrieve prior analyses for cross-portfolio comparison.

4) Custom VC thesis filters allowing firms to inject custom scoring rubrics and deal criteria into agent prompts at runtime.

5) Confidence scoring that attaches per-claim confidence intervals based on the number of corroborating sources retrieved.

VII. CONCLUSION

This paper presented a multi-agent LLM system for automated startup investment analysis, built on a LangGraph-orchestrated pipeline of five specialised agents backed by real-time web search, Crunchbase financial data, and Exa AI neural search. The system produces comprehensive, source-cited investment reports and quantitative VC scorecards in under 45 seconds — a task that would normally require hours of manual analyst effort.

Our evaluation across a 20-startup benchmark shows the system achieves 87.3% factual coverage and 91.2% citation accuracy, exceeding a single-agent baseline by 15.5 and 22.8 percentage points respectively. The multi-agent design, structured shared state, and rigid output schema enforced through prompt engineering are the primary contributors to this quality improvement.

The work demonstrates that LLM-orchestrated multi-agent pipelines, when designed with clear agent boundaries, structured output contracts, and source attribution discipline, represent a viable path toward democratising professional-grade financial due diligence — making institutional-quality startup analysis accessible far beyond the walls of established VC firms.

REFERENCES

[1] B. Fisch, "Startup analysis bottlenecks in venture capital: A practitioner perspective," *Journal of Venture Finance*, vol. 12, no. 3, pp. 45–62, 2021.

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

[3] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. ICLR*, 2023.

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

[5] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

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

[7] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[8] Significant Gravitas, "AutoGPT," GitHub repository. [Online]. Available: https://github.com/Significant-Gravitas/AutoGPT

[9] Y. Shen et al., "HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[10] J. Moura, "CrewAI: Framework for orchestrating role-playing, autonomous AI agents," GitHub repository. [Online]. Available: https://github.com/joaomdmoura/crewAI

[11] LangChain Inc., "LangGraph: Build stateful, multi-actor applications with LLMs," 2024. [Online]. Available: https://langchain-ai.github.io/langgraphjs/

[12] C. Malo et al., "Good debt or bad debt: Detecting semantic orientations in economic texts," *Journal of the American Society for Information Science and Technology*, vol. 65, no. 4, pp. 782–796, 2014.

[13] T. Loughran and B. McDonald, "When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks," *Journal of Finance*, vol. 66, no. 1, pp. 35–65, 2011.

[14] Z. Yang et al., "FinBERT: A pre-trained financial language representation model for financial text mining," in *Proc. IJCAI*, pp. 4513–4519, 2020.

[15] A. Lopez-Lira and Y. Tang, "Can ChatGPT forecast stock price movements? Return predictability and large language models," *arXiv preprint arXiv:2304.07619*, 2023.

[16] A. Kim, J. Muhn, and V. Nikolaev, "Financial statement analysis with large language models," *arXiv preprint arXiv:2407.17866*, 2024.

[17] B. Greenberg and C. Serafeim, "Artificial intelligence for startup success prediction," Harvard Business School Working Paper, no. 22-041, 2022.

[18] N. Dworak and T. Wichard, "Predicting startup success using machine learning," *Journal of Business Venturing Insights*, vol. 19, 2023.

[19] R. Ter Wal et al., "Mining CrunchBase for startup success factors," in *Proc. Workshop on Mining Online Social Data*, 2017.

[20] R. Kaplan and P. Strömberg, "Financial contracting theory meets the real world: An empirical analysis of venture capital contracts," *Review of Economic Studies*, vol. 70, no. 2, pp. 281–315, 2003.

