

# Technical Specification: Rule-Guided Multi-Agent System for Financial Report Analysis

## 1. Overview

This document outlines the technical specification for a rule-injected, multi-agent AI system designed to analyze financial reports (10-K, 10-Q, earnings reports, etc.) using explicit user-defined rules, accounting domain knowledge, and fraud detection heuristics.

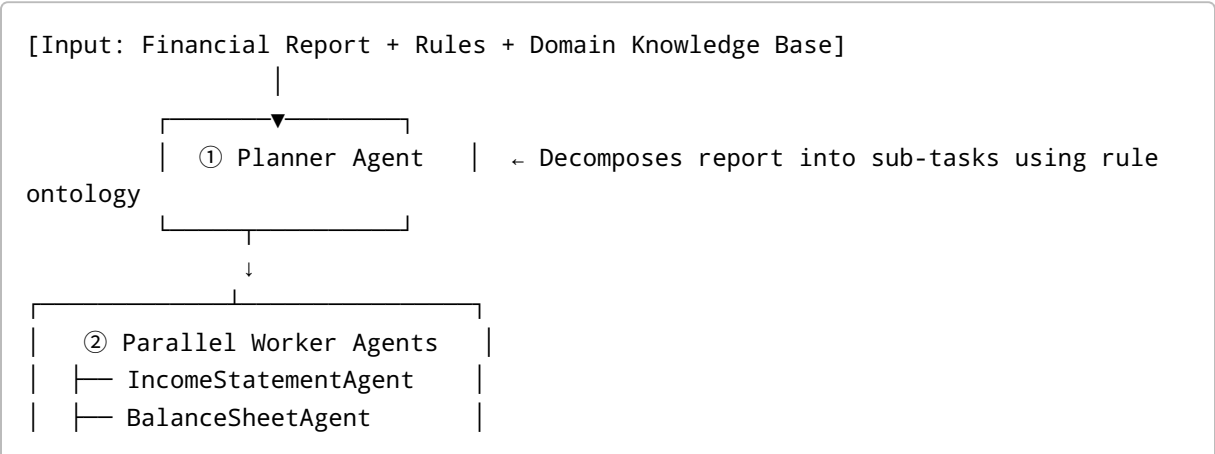
The system leverages large language models (LLMs) orchestrated via a directed state machine architecture to perform task decomposition, parallel section analysis, rule-based verification, feedback loops, and final report synthesis. A key differentiator of this system is its ability to ingest structured and unstructured domain knowledge and apply it throughout the agentic reasoning and verification process.

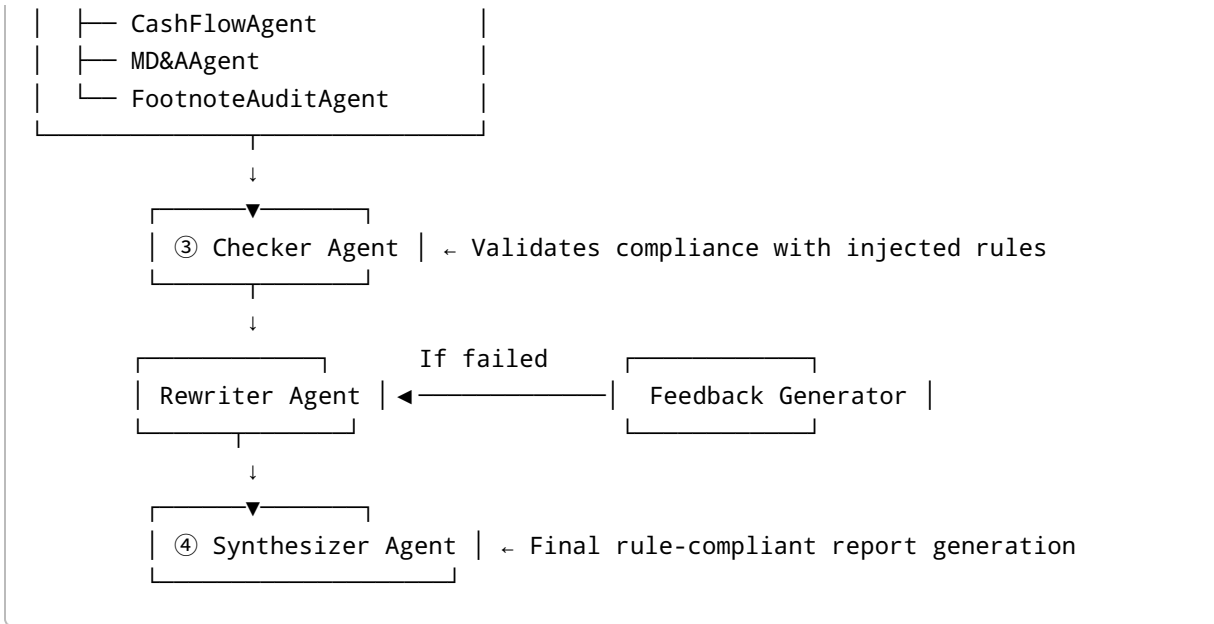
## 2. System Objectives

- Automatically interpret financial reports with a structured, explainable logic chain.
- Inject user-defined domain knowledge, such as:
  - Accounting rules (e.g., GAAP/IFRS)
  - Financial statement analysis techniques
  - Forensic red flags and fraud detection heuristics
- Support flexible customization of interpretation rules for different investor strategies.
- Enforce rule compliance with loopback correction logic.
- Provide auditable, investor-friendly outputs.

## 3. Core Architecture

### 3.1 Workflow Diagram





## 4. Input Components

### 4.1 Financial Documents

- Raw financial statements (PDF, HTML, XBRL)
- Extracted structured data (via PDF parsers / table detectors)

### 4.2 Domain Knowledge and Rule Base

- Financial literacy guides (e.g., "How to Read a 10-K")
- Red-flag checklists (fraud indicators)
- IFRS/GAAP accounting rules (structured or free-form)
- User-defined logical rules, thresholds, and alerting conditions (JSON, YAML, or natural language)

The rule base is parsed and embedded into both the planning and checking stages. RAG (retrieval-augmented generation) mechanisms are used to ensure relevant rule context is injected at every agent level.

## 5. Agent Roles and Prompts

### 5.1 Planner Agent

- **Function:** Breaks down the report into components, informed by the rule ontology.
- **Prompt:** Plans tasks with grounding in investor-defined logic and known risk dimensions.

## 5.2 Worker Agents (Parallel Execution)

Each worker focuses on a report section with prompt, tool access, and rule grounding specific to that section:

- **IncomeStatementAgent:** Revenue anomalies, gross margin logic, matching revenue with accounts receivable changes.
- **BalanceSheetAgent:** Asset/liability consistency, off-balance debt, changes in working capital.
- **CashFlowAgent:** Discrepancy between net income and operating cash flows, CapEx control.
- **MD&AAgent:** Forward-looking narrative consistency, hedging language, risk disclosure quality.
- **FootnoteAuditAgent:** Accounting policy changes, lease liabilities, hidden risk disclosures.

## 5.3 Checker Agent

- **Function:** Validates each section against domain-specific rules.
- **Mechanism:**
  - Ingests explicit rules and financial heuristics.
  - Determines compliance through LLM reasoning + symbolic rule matching.
  - Can return actionable suggestions and initiate repair.

## 5.4 Rewriter Agent

- **Function:** Revises and regenerates flawed sections based on Checker feedback.
- **Prompt:** Uses chain-of-thought with injected correction rationale.

## 5.5 Synthesizer Agent

- **Function:** Merges all validated content into a comprehensive, investor-facing report.
- **Features:**
  - Rule traceability annotations
  - Highlighting of red-flagged insights
  - Optional visualizations







---

## 6. Technology Stack




Component	Technology
LLM Backend	OpenAI GPT-4, Claude, Mistral
Agent Orchestration	LangGraph (state machine-based DAG)
Prompt Management	LangChain / LangSmith
Rule Injection Layer	JSON/YAML rule parser + RAG + prompt fusion
Document Parsing	PyMuPDF, PDFMiner, Pandas
Knowledge Base	Vector DB + RAG (e.g., FAISS + LlamaIndex)

Component	Technology
Output Formatting	Markdown / HTML / PDF ReportGen

## 7. Key Features

-  **Explicit Rule Injection** into planning and checking stages
-  **Custom Red Flag Logic** (user-provided and/or pretrained)
-  **Dynamic Task Planning** informed by rule ontology
-  **Parallel Section Processing** with modular Agents
-  **Multi-Agent Collaboration with Loopback Correction**
-  **Traceable Reasoning Logs** for auditable decision support

## 8. Future Extensions

- Auto-conversion of user documents (books, checklists) into structured rulesets
-  Fine-tuning checker agents with verified rule violations
-  Trend analysis across quarters (QoQ, YoY)
- Cross-session memory for longitudinal company profiling
-  Frontend rule designer for business users (no-code)

## 9. Project Deployment Plan

Phase	Deliverables
P1 - MVP	IncomeStatementAgent + Checker + Synthesizer + Rule RAG flow
P2 - Full Rule Base	Upload + parse user rules, inject into agents and checker
P3 - Agent Graph	Full DAG with loopback correction + domain-specific prompts
P4 - UI & Export	Notion/PDF export, rule annotations, frontend interface

## 10. Summary

This system is not just a document summarizer, but a **rule-driven reasoning engine** for financial reporting. It integrates domain knowledge directly into every layer of analysis, ensuring that outputs are not only accurate, but **justified, rule-compliant, and auditable**. Its modular agent architecture supports iterative correction, intelligent collaboration, and end-user rule injection, distinguishing it from conventional summarization agents or financial dashboards.