**TCS Forecasting Agent — AI System Documentation**

**1️⃣ Overview**

The TCS Forecasting Agent is a multi-tool, LangChain-based AI system that autonomously analyzes recent financial reports and management commentary to produce qualitative and quantitative forecasts for Tata Consultancy Services (TCS). It combines structured financial data extraction (from PDFs) with natural language reasoning over management statements, validated through a local or cloud-based LLM.

**2️⃣ AI Stack and Reasoning Approach**

The agent uses the LangChain ReAct agent architecture to orchestrate stepwise reasoning (Think → Act → Observe → Reflect → Answer). It integrates multiple tools to extract, analyze, and synthesize both quantitative financial metrics and qualitative management commentary.

**3️⃣ Tools and Models Employed**

|  |  |  |
| --- | --- | --- |
| Component | Technology | Description |
| LLM Provider | Ollama / OpenRouter | LLM reasoning engine (models: llama3.1:8b, phi4, deepseek-r1:70b) |
| LangChain | ReAct Agent Framework | Handles reasoning and tool orchestration |
| FinancialDataExtractorTool | Custom Tool | Extracts financial metrics from PDF reports using Camelot, pdfplumber, or OCR |
| QualitativeAnalysisTool | Custom Tool | Analyzes management commentary for sentiment and forward guidance |
| OCR | pdf2image + pytesseract | Used for scanned financial documents |
| Embeddings | all-MiniLM-L6-v2 | Used for semantic understanding and RAG extension |
| LLM Wrapper | OllamaLLM | Custom wrapper for local inference |
| Function-calling | LangChain @tool | Allows tools to be invoked via structured reasoning steps |

**4️⃣ End-to-End Achievement**

The system retrieves and parses recent TCS reports, extracts numerical and qualitative insights, and synthesizes a unified forecast in structured JSON. The pipeline includes data retrieval, parsing, LLM-based reasoning, and structured output validation.

**5️⃣ Guardrails and Evaluation**

To ensure stability and reliability, multiple guardrails were added:  
- JSON schema validation for forecast outputs  
- Timeout control using asyncio.wait\_for  
- Error handling and logging for failed LLM calls  
- Retry mechanisms for 429 rate limit responses  
- Deterministic enrichment fallback if LLM fails  
- Prompt control enforcing schema adherence and reasoning order

**6️⃣ Limits and Tradeoffs**

Several challenges were encountered during implementation:  
- OpenRouter rate limits → mitigated by switching to Ollama local LLM  
- Long reasoning chains causing timeouts → reduced by structured prompts  
- Event loop conflicts in FastAPI → solved using nest\_asyncio  
- Parsing errors due to multi-line output → handled by schema checks  
- Large model latency on local hardware → optimized by using phi4 or llama3.1:8b

**7️⃣ Future Improvements**

- Integrate LangGraph for more advanced tool orchestration  
- Add vector-based RAG for multi-report retrieval  
- Integrate live financial APIs for real-time data  
- Add self-evaluation and confidence scoring for forecasts  
- Expand support to other tickers (Infosys, Wipro, etc.)