Agentic AI Solution Architecture

This section details the technical architecture for FinAssist A.I.'s multi-agent system, including LLM orchestration, inter-agent communication protocols, RAG integration for market data, and compliance-focused guardrails.

Multi-Agent LLM Orchestration

FinAssist A.I. implements a **Sequential Multi-Agent Architecture** where three specialized LLM-powered agents collaborate through a central coordinator. Each agent maintains distinct system prompts, tool access, and output schemas while sharing context through a structured handoff protocol.

Agent-to-LLM Mapping

Agent	LLM Configuration	Rationale
Client Intelligence Agent (CIA)	GPT-4 Turbo (temp: 0.2)	Low temperature for deterministic extraction of financial data; strong structured output capabilities
Market & Trend Agent (MTA)	Claude 3.5 Sonnet (temp: 0.3)	Superior reasoning for multi-source synthesis; 200K context for ingesting market reports; excellent at nuanced analysis
Portfolio Strategy Agent (PSR)	GPT-4 Turbo (temp: 0.4)	Balance of creativity for strategy generation with structured compliance output; function calling for Monte Carlo integration
Coordinator Layer	LangGraph State Machine	Manages agent sequencing, state persistence, conditional routing, and error recovery

Inter-Agent Communication Protocol

Agents communicate through a structured JSON handoff protocol managed by the LangGraph coordinator. Each agent receives upstream context and produces typed outputs consumed by downstream agents.

Data Flow: Client → Market → Strategy

1. **CIA Output → MTA Input:** Client Financial Genome (JSON schema: risk_tolerance_index, liquidity_needs, time_horizon, tax_considerations, suitability_flags)
2. **MTA Output → PSR Input:** Market Context Package (JSON schema: asset_class_rankings, scenario_projections, volatility_assessment, sector_momentum)
3. **PSR Output → Final Report:** Advisor Recommendation Report (JSON schema: strategy_options[], risk_metrics, monte_carlo_results, compliance_attestation)

Shared Memory Architecture

A Redis-backed state store maintains conversation context, allowing agents to reference previous interactions and enabling the coordinator to implement checkpointing for long-running analyses. State includes:

- Client session context (persisted across planning sessions)
- Intermediate agent outputs (for debugging and audit trails)

- RAG retrieval cache (to avoid redundant API calls)
- Human feedback signals (CFP corrections feed back to improve prompts)

RAG Integration for Market Intelligence

The Market & Trend Agent (MTA) leverages RAG to synthesize real-time market data with historical context and research reports.

Component	Specification
Data Sources	Bloomberg API (real-time quotes), Federal Reserve FRED (economic indicators), SEC EDGAR (filings), firm research library (PDF reports)
Embedding Model	FinBERT embeddings (fine-tuned for financial domain) for research reports; text-embedding-3-small for general queries
Vector Database	Weaviate with hybrid search (BM25 + vector); separate collections for market data, research, and regulatory filings
Update Frequency	Market data: 15-minute refresh; Economic indicators: daily; Research reports: on ingestion; SEC filings: within 4 hours of publication
Retrieval Strategy	Query decomposition (split complex market questions into sub-queries); multi-index search across data types; temporal weighting (recent data weighted 2x)

Agent Prompt Templates

Client Intelligence Agent System Prompt

```
You are a fiduciary-minded financial analyst specializing in client profiling. Extract and normalize financial data from provided inputs. Flag missing or contradictory information. Never assume values not explicitly stated.

Output a Client Financial Genome with: risk_tolerance_index (1-10), liquidity_needs (enum), time_horizon_years, tax_bracket, suitability_flags[], data_quality_score, missing_fields[].
```

Portfolio Strategy Agent System Prompt

```
You are a quantitative portfolio strategist. Given the Client Financial Genome and Market Context, generate 3-4 distinct strategy options (Conservative, Moderate, Aggressive, Tax-Optimized). Each strategy must include: asset allocation percentages, expected return range, risk metrics (Sharpe estimate, max drawdown), and a plain-English rationale suitable for client presentation. Never guarantee returns. Always include appropriate disclaimers.
```

External Tool Integration

Agents access external tools via OpenAI function calling or LangChain tool definitions:

- **Monte Carlo Engine:** Python-based simulation (10,000 iterations) called by PSR agent via function call; returns probability distributions for portfolio outcomes
- **Risk Calculator:** Computes Sharpe ratio, Sortino ratio, Value-at-Risk (VaR), and maximum drawdown from proposed allocations
- **Market Data API:** Real-time quotes, historical prices, and economic indicators retrieved by MTA agent
- **Compliance Checker:** Validates strategy outputs against FINRA/SEC suitability rules; flags non-compliant recommendations before delivery

Compliance Guardrails & Output Validation

Given the regulated nature of financial advice, FinAssist implements multiple validation layers:

Guardrail	Implementation
No Return Guarantees	Regex filter blocks outputs containing "guaranteed," "certain return," "risk-free" unless in disclaimer context
Suitability Validation	Strategies cross-checked against client risk tolerance; aggressive strategies flagged for conservative clients
Disclaimer Injection	Post-processing appends FINRA-compliant disclaimers to all client-facing outputs
Audit Trail	Complete reasoning chain logged: client inputs → agent outputs → final recommendation; 7-year retention for regulatory compliance
Human-in-the-Loop	CFP must approve all recommendations before client presentation; override capability at every step

Infrastructure & Performance

- **End-to-End Latency:** < 15 seconds for full Client → Market → Strategy pipeline (excluding Monte Carlo, which streams results)
- **Deployment:** AWS (SOC 2 Type II compliant) or Azure; API endpoints via AWS API Gateway with WAF protection
- **Scaling:** Horizontal scaling via Kubernetes; agent instances scale independently based on queue depth
- **Cost Optimization:** Caching layer for repeated market queries; prompt compression for context efficiency; GPT-4 Turbo for cost-effective long-context operations

[End of AI Solution Architecture Section]