Financial Agents App: Architectural Document

Financial Application Development Team

August 22, 2025

Contents

1 Overview

The Financial Agents App is a web-based financial analysis and planning tool built using the Streamlit framework. It provides users with a comprehensive platform to analyze portfolios, track financial goals, calculate taxes, and retrieve market and news insights. The application is designed with a modular architecture, leveraging multiple agents for specialized tasks, a workflow router for query dispatching, and external data sources for real-time financial information. The user interface (UI) is styled with a peacock blue theme, ensuring a consistent and visually appealing experience.

This document outlines the system architecture, key components, data flow, and design decisions. It is intended for developers, architects, and stakeholders to understand the application's structure and guide future enhancements.

2 System Architecture

The Financial Agents App follows a modular, agent-based architecture with a clear separation of concerns. The system is divided into four primary layers: User Interface, Workflow, Agents, and Core Services. External data sources (e.g., Yahoo Finance, Google News) provide real-time data, and a Retrieval-Augmented Generation (RAG) system enhances agent responses with contextual information.

2.1 Architecture Diagram

The system architecture is depicted in the following workflow diagram, illustrating the flow of data from user input to response rendering:

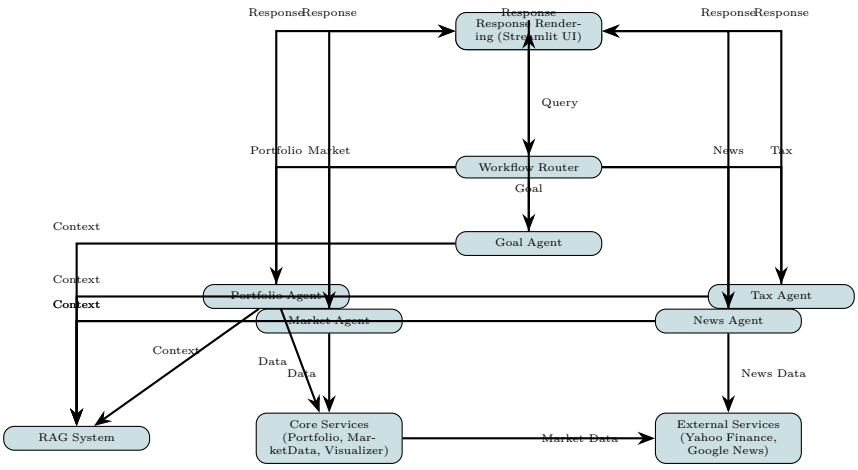

Figure 1: Workflow Diagram for Financial Agents App

3 Components

The application comprises several key components, each with specific responsibilities:

3.1 User Interface (Streamlit)

The UI is built using Streamlit, a Python framework for creating web applications. It features a three-column layout:

- **Column 1**: Displays the app emblem, financial goals form, or W2 tax calculator based on session state.

- **Column 2**: Contains the main query input form and application title.
- **Column 3**: Renders responses (portfolio, market, goals, tax, or news) and a CSV uploader for portfolio data.

The UI uses a custom CSS stylesheet with a peacock blue theme (`#005F73`) for branding, with white text and light gray borders for readability.

3.2 Workflow Layer

The `create_workflow` function (from `workflow.agent_router`) routes user queries to the appropriate agent based on keywords (e.g., "tax" for TaxEducationAgent, "market" for MarketAnalysisAgent). It uses the `AgentState` dictionary to manage query context and responses, ensuring stateless processing.

3.3 Agents Layer

The system includes five specialized agents:

- **PortfolioAnalyzer**: Analyzes user-uploaded portfolios, calculating returns, volatility, and Sharpe ratio.
- **InnerGoalPlanningAgent**: Manages financial goals, computing required monthly savings and projections.
- **TaxEducationAgent**: Calculates federal and state taxes based on W2 inputs and provides tax-related explanations.
- **MarketAnalysisAgent**: Fetches S&P 100 stock data or general market trends (S&P 500, Dow Jones) using `yfinance`.
- **NewsSynthesizerAgent**: Retrieves and analyzes financial news using `gnews`, providing structured insights.

Each agent uses a `ChatOpenAI` model (GPT-4o-mini) for natural language processing and integrates with the RAG system for context.

3.4 Core Services

Core services provide shared functionality:

- **Portfolio**: Manages portfolio data structures and calculations.
- **MarketData**: Interfaces with `yfinance` for real-time market data.
- **PortfolioVisualizer**: Generates plots and metrics using `plotly.express` for portfolio and market data.

3.5 RAG System

The `RAGSystem` retrieves contextual information from JSON files (`fin_goal_planning_agent.json`, etc.) to enhance agent responses. It is initialized in each agent and provides query-relevant context to the LLM.

3.6 External Services

- **Yahoo Finance (`yfinance`)**: Provides stock and index data (e.g., prices, market cap, historical data).
- **Google News (`gnews`)**: Fetches recent financial news articles for the NewsSynthesizer-Agent.

4 Data Flow

The data flow through the system is as follows:

1. The user submits a query or form input (e.g., portfolio CSV, W2 data, or financial goal) via the Streamlit UI.
2. The query is passed to the workflow layer, which updates the `AgentState` and routes it to the appropriate agent based on keywords.
3. The selected agent processes the query:
   - **Portfolio**: Processes CSV data, fetches market data via `MarketData`, and generates metrics.
   - **Goal**: Calculates savings plans using user inputs and stores goals in session state.
   - **Tax**: Computes federal and state taxes using simplified 2025 tax brackets.
   - **Market**: Fetches stock or index data via `yfinance` and generates analysis.
   - **News**: Retrieves articles via `gnews` and synthesizes insights.
4. Agents use `RAGSystem` to retrieve context and `ChatOpenAI` to generate textual analysis.
5. The response is stored in `AgentState["response"]` and returned to the UI.
6. The UI renders the response in Column 3, using `plotly` for visualizations and `st.markdown` for text.

5 Design Decisions

- **Modular Agent Architecture**: Separate agents for each function (portfolio, goals, tax, market, news) allow independent development and scalability.
- **Streamlit for UI**: Chosen for rapid development and Python integration, with custom CSS for branding.
- **GPT-4o-mini**: Selected for cost-effective, high-quality NLP capabilities in generating analysis.
- **yfinance and gnews**: Used for reliable, real-time market and news data without requiring paid APIs.
- **RAG Integration**: Enhances agent responses with preloaded financial context, reducing LLM hallucination.
- **Session State Management**: Streamlit's session state persists user data (e.g., portfolio, goals) across interactions.
- **Error Handling**: Try-except blocks and `tenacity` retries ensure robustness against API failures.
- **Simplified Tax Model**: Uses 2025 tax brackets with standard deductions for user-friendly estimates, with a disclaimer for accuracy.

6 Dependencies

- **Python Libraries**: `streamlit`, `yfinance`, `gnews`, `pandas`, `plotly.express`, `langchain_openai`, `langchain_core`, `tenacity`.
- **Internal Modules**: `workflow`, `core`, `utils`, `agents`.
- **External Services**: Yahoo Finance, Google News.

7 Scalability and Extensibility

- **Scalability**: The agent-based design allows adding new agents without modifying existing ones. The workflow router can be extended to handle new query types.
- **Extensibility**: New data sources (e.g., alternative news APIs) or agents (e.g., crypto analysis) can be integrated by updating the RAG system and workflow logic.
- **Performance**: Caching mechanisms (e.g., session state, potential database integration) can improve response times for frequent queries.

8 Conclusion

The Financial Agents App provides a robust, user-friendly platform for financial analysis and planning. Its modular architecture, leveraging Streamlit, agent-based processing, and external data sources, ensures flexibility and maintainability. The updated workflow diagram, now fully contained within the page, clarifies the data flow, aiding developers in understanding and extending the system. Future enhancements could include additional agents, more sophisticated tax models, or integration with premium data APIs.