Agent Assignment Description [Deadline 13 May 6pm]

Candidates will build a **multi-source, multi-agent finance assistant** that delivers spoken market briefs via a Streamlit app. They'll implement advanced data-ingestion pipelines (APIs, web scraping, document loaders), index embeddings in a vector store for Retrieval-Augmented Generation (RAG), and orchestrate specialized agents (API, scraping, retrieval, analytics, LLM, voice) via FastAPI microservices. Voice I/O pipelines will leverage open-source toolkits , and text agents will be built with Langgraph and CrewAI [Any of your choice]. All code must be open-source, documented with AI-tool usage logs, and deployed on Streamlit. Make use of openly available MCPs wherever possible. You are free to use any framework not just langchain/langgraph.

Use Case: Morning Market Brief

Every trading day at 8 AM, a portfolio manager asks:

> "What's our risk exposure in Asia tech stocks today, and highlight any earnings surprises?"
> The system responds verbally:
> "Today, your Asia tech allocation is 22 % of AUM, up from 18 % yesterday. TSMC beat estimates by 4 %, Samsung missed by 2 %. Regional sentiment is neutral with a cautionary tilt due to rising yields."

Architecture

Agent Roles

- **API Agent**: Polls real-time & historical market data via AlphaVantage or Yahoo Finance
- **Scraping Agent**: Crawls filings using Python loaders . [Finding out simpler solution would give you bingo points!!] [MCP if you may]
- **Retriever Agent**: Indexes embeddings in FAISS or Pinecone and retrieves top-k chunks
  Analysis Agent: .
- **Language Agent**: Synthesizes narrative via LLM using LangChain's retriever interface
  Voice Agent: Handles STT (Whisper) → LLM → TTSpipelines

Orchestration & Communication

- Microservices built with **FastAPI** for each agent
- Routing logic: voice input → STT → orchestrator → RAG/analysis → LLM → TTS or text.

- Fallback: if retrieval confidence < threshold, prompt user clarification via voice.

Deliverables

- **GitHub Repository** with modular code (`/data_ingestion`, `/agents`, `/orchestrator`, `/streamlit_app`), dependency files, and Docker configuration.
- **`README.md`**: Architecture diagrams, setup & deployment instructions, framework/toolkit comparisons, performance benchmarks.
- **`docs/ai_tool_usage.md`**: Detailed log of AI-tool prompts, code generation steps, and model parameters.
  (Optional) Demo video or GIFs of end-to-end voice and text interactions.
- Deploy this agent somewhere and provide the share the URL [you will only be qualified if the agent is deployed]..simple streamlit application also works.

Evaluation Criteria

- **Technical Depth**: Robust data pipelines, RAG accuracy, and quantitative analysis.
- **Framework Breadth**: Usage of ≥2 toolkits per agent category.
  Code Quality: Modularity, readability, tests, and CI setup.
- **Documentation**: Clarity, completeness, and transparency about AI-tool assistance.
- **UX & Performance**: Intuitive Streamlit UI, low latency, and reliable voice I/O.

Good luck building a sophisticated, open-source multi-agent finance assistant that speaks!