Product Requirements Document (PRD): AURA - Autonomous Unified Research Assistant

Objective

To build an intelligent, autonomous research application that leverages multi-agent collaboration to search, analyze, and summarize academic materials. The app empowers users to conduct deep research automatically, synthesize findings into coherent essays, and interactively query the summarized material through a memory-enabled RAG chatbot.

Core Components

Supervisor Agent (Orchestrator)

- - Accepts the user's query and orchestrates the overall workflow.
- - Uses Serper API to fetch relevant research papers/articles.
- - Categorizes the research corpus based on thematic clusters or metadata.
- - Assigns batches of papers to subordinate agents for analysis.
- - Collects responses and triggers the Summarizer Agent.

Subordinate Agents (Analyst Agents)

- - Independently analyze assigned documents.
- - Extract core ideas, methods, and findings.
- - Summarize results focusing on clarity, novelty, and gaps.
- - Output structured notes (JSON format).

Summarizer Agent

- - Synthesizes outputs from subordinate agents into a cohesive essay.
- - Writes a coherent, well-structured essay with citations.
- - Saves output in a text file.
- - Triggers the RAG chatbot activation.

RAG Chatbot (Post-Research Interaction Tab)

- - Allows users to ask questions using context from research results.
- - Implements FAISS + OpenAI embeddings for retrieval.
- - Uses GPT-4o with ReAct reasoning and LangGraph memory.
- - Supports follow-up and comparative questions.

Workflow

- - User provides a research topic or question.
- - Supervisor fetches articles and distributes them to subordinate agents.
- - Subordinate agents analyze and return structured outputs.
- - Summarizer Agent compiles results into a cohesive essay and stores it.
- - RAG Chatbot is activated for user interaction.

Tech Stack

Backend: FastAPI (API server), Python

Agents/Logic: LangGraph, Langchain, OpenAI API (GPT-4o), Serper

Frontend: Node.js, TailwindCSS

RAG Engine: FAISS, OpenAI embeddings, GPT-4o (ReAct pattern)

Storage: Local text files, FAISS vector store

Frontend Design

Tab 1: Research Panel

 - Input box for query, progress tracker, and essay download button

Tab 2: RAG Chatbot

 - Chat interface with GPT-4o, context-based retrieval, and export options

Key Autonomy Principles

- - Supervisor decides article distribution autonomously.
- - Subordinate agents work in parallel asynchronously.
- - Summarizer triggers only after all subordinate results are collected.
- - RAG chatbot initializes post summarization.

Deliverables

- - Multi-agent orchestration logic (LangGraph)
- - Autonomous document distribution mechanism
- - Summarization pipeline and essay generator
- - RAG pipeline with FAISS and embeddings
- - Node.js + Tailwind frontend with tabbed interface
- - FastAPI backend endpoints for query, status, summary, and chat

Success Criteria

- - Supervisor autonomously coordinates the workflow.
- - Agents communicate asynchronously and maintain traceability.
- - RAG chatbot references extracted data accurately.
- - Frontend is clean, responsive, and modular.