**Autonomous Multi-Agent Job Search and Application System**

**Project Intent and Overarching Goals**

This project aims to create an intelligent, autonomous system that handles the entire job search and application process with minimal human intervention. The core vision is to develop a network of specialized AI agents that work together to discover relevant job opportunities, maintain a comprehensive professional profile, and streamline the application process—all operating seamlessly in the background.

The fundamental goals of this system are:

* **Autonomous Job Discovery**: Enable intelligent agents to find and analyze job opportunities across various platforms without manual intervention. Unlike simple scrapers, these agents should be capable of dynamically navigating any job site given only a URL, applying consistent search parameters across different interfaces.
* **Intelligent Profile Management**: Create and maintain a unified professional profile that aggregates information from multiple sources (resume, LinkedIn, GitHub), recognizing when the same project or experience appears across different platforms. This goes beyond simple keyword extraction to develop a semantic understanding of qualifications.
* **Automated Application Process**: Streamline job applications through intelligent document customization, response generation for application questions, and automated submissions where possible. The system should recognize similar questions across applications and reuse appropriate responses.
* **Background Operation**: Ensure the entire system functions efficiently in the background without requiring active monitoring. The system should operate asynchronously on scheduled intervals to discover and evaluate new job opportunities while maintaining cost efficiency.

The ultimate vision is to create a "job market PR agent and assistant" that eliminates the chaos of maintaining streamlined information across multiple platforms, continuously updates the user's professional profile, and autonomously handles the repetitive aspects of job searching and application.

**System Architecture**

The system is structured as a network of specialized agents that process and refine data in a bottom-up flow, with sub-agents feeding information to manager agents that coordinate activities. This architecture enables efficient information processing and autonomous decision-making throughout the job search and application lifecycle.

A diagram of a job manager

AI-generated content may be incorrect.**Information Flow and Component Relationship**

The system operates through several layers of processing, with information flowing from specialized sub-agents up to manager agents and ultimately to an orchestrator that supervises the entire process:

**Profile Information Collection and Management**: At the foundation level, four specialized sub-agents (Resume, LinkedIn, GitHub, Other) gather and process information about the user's professional background. These agents extract detailed information from their respective sources and feed it upward to the Profile Manager. The Profile Manager synthesizes this information into a comprehensive, semantically rich "Unified Profile" document that represents the user's complete professional identity.

**Job Search and Collection**: In parallel, four specialized sub-agents (LinkedIn, Company, Dynamic, Indeed) actively search for job opportunities across different platforms. Unlike in traditional architectures where managers direct sub-agents, these specialized agents autonomously navigate their respective platforms and feed discovered opportunities upward to the Search Manager. The Search Manager consolidates these findings into a "Potential Job List" document, removing duplicates and standardizing the format.

**Orchestration and Evaluation**: The Orchestrator sits at the center of the system, receiving both the Unified Profile and Potential Job List. It coordinates the overall workflow and passes this information to the Job Evaluator. The Job Evaluator applies sophisticated matching algorithms to rank each job opportunity based on relevance to the profile, posting date, company reputation, and other factors, producing a "Ranked Job List" document.

**Application Process**: The Application Manager receives the Ranked Job List and processes each job opportunity in priority order. For each application, two specialized agents work simultaneously:

* The "App Doc/Form Req" agent analyzes the application requirements and form questions
* The "Company Info" agent researches the company's profile, mission, and relevant projects

These inputs feed into two processing agents:

* The "Self Doc Tuner" customizes the user's resume and cover letter specifically for each application
* The "Form Answer Generator" creates optimized responses to application questions

The system includes an intelligent decision point that determines whether human input is required for certain questions. When needed, it requests user input; otherwise, it proceeds autonomously. The "Actual Applicator" agent then takes the optimized documents and answers to complete and submit the application.

This architecture represents a significant advancement over traditional linear workflows, as it allows for parallel processing, intelligent decision-making, and adaptive responses to the unique requirements of each job opportunity and application process.

**Phased Implementation Plan**

The following numbered entries correspond to each implementation phase of the project, progressing from an initial MVP to a fully autonomous system.

1. **LinkedIn Job Search MVP:** We'll establish basic functionality for automated job discovery on LinkedIn, creating the foundation for the broader system. The implementation begins with creating LinkedIn login automation via Playwright, ensuring secure credential handling. We'll develop scraping routines to extract and store LinkedIn jobs in structured JSON format. A basic semantic ranking system using embedded keywords (provided manually at this stage) will help prioritize opportunities. The system will be configured with APScheduler to enable background operation, allowing job searches to run at scheduled intervals without user intervention.

*Success Criteria:* The system successfully logs into LinkedIn, retrieves job listings based on predefined keywords, and presents them to the user in a ranked order through the Streamlit interface.

*Stack:* Streamlit for frontend interface, Playwright for browser automation, JSON for data storage, APScheduler for background tasks, Sentence-Transformers for semantic matching.

1. **Resume-Based Profile Agent:** We'll add backward integration with the user's resume, eliminating the need for manually provided keywords by extracting them directly from resume documents. Development focuses on creating PDF resume parsing capability that handles various resume formats. The system will extract skills, experiences, and education information automatically, then generate embeddings for these elements to enable semantic matching. These dynamically extracted keywords will be integrated into the Search Manager for improved job discovery without requiring manual input.

*Success Criteria:* The system can extract relevant professional information from resume PDFs and use this information to enhance job search queries, eliminating the need for manual keyword specification.

*Stack:* PyMuPDF for extracting text from resume PDFs, Sentence-Transformers for generating embeddings from resume content.

1. **LinkedIn Profile Integration and Profile Manager:** We'll expand profile data sources to include LinkedIn and implement the Profile Manager agent that receives information from the Resume and LinkedIn agents. Development includes LinkedIn profile scraping to extract comprehensive professional data. Following the new architecture, the specialized agents will feed information upward to the Profile Manager, which will aggregate and reconcile information from both sources. The system will create an initial version of the Unified Profile document, capturing the user's professional identity in a structured format.

*Success Criteria:* The system maintains a consistent Unified Profile derived from both resume and LinkedIn data, resolving conflicts between sources and identifying complementary information.

*Stack:* Playwright for LinkedIn profile navigation, Sentence-Transformers for embeddings, LangChain for Profile Manager agent capabilities.

1. **Enhanced Job Ranking and Orchestrator Implementation:** We'll implement the Orchestrator component and develop the Job Evaluator to create the ranked job list. The implementation focuses on developing a comprehensive ranking heuristic that considers keyword matches, posting date, company reputation, and profile relevance. We'll create persistent storage for the Potential Job List and Ranked Job List documents with visualization and filtering capabilities in the Streamlit frontend. This phase establishes the central coordination mechanism that receives information from both the Profile Manager and Search Manager.

*Success Criteria:* The system consistently identifies the most relevant job opportunities and maintains them in an organized, ranked list with appropriate metadata for tracking.

*Stack:* Custom Python modules for multi-factor job ranking, Excel/Google Sheets integration for document storage, CrewAI for orchestrator implementation.

1. **Advanced Unified Profile Management:** We'll enhance the Profile Manager to receive information from additional sources, including GitHub repositories and other platforms. We'll expand the specialized agents to extract information from these additional sources and feed it upward to the Profile Manager. The system will implement semantic aggregation capabilities to identify related experiences across platforms and develop deduplication algorithms that preserve the richest information for each experience. A version history mechanism will track profile evolution over time.

*Success Criteria:* The Unified Profile document incorporates comprehensive information from all sources, recognizing when the same project or experience appears across different platforms.

*Stack:* LangChain for semantic understanding and relationship mapping, CrewAI for managing inter-agent communication and task delegation.

1. **Company Research and Document Customization:** We'll implement the Company Info agent and Self Doc Tuner components that enhance application quality. The Company Info agent will research target companies to extract mission statements, recent projects, and culture information. The Self Doc Tuner will use this information along with the job description to customize resumes and cover letters for specific positions. This phase focuses on creating tailored documents that maximize the candidate's fit for each specific role and company.

*Success Criteria:* The system produces customized resumes and cover letters that highlight relevant experience and skills for specific job opportunities, incorporating company-specific language and priorities.

*Stack:* GPT-4 API for document generation and customization, Playwright for company research, CrewAI for agent implementation.

1. **Form Answer Generation and User Input Management:** We'll implement the Form Answer Generator and the decision mechanism that determines when user input is required. The system will analyze application questions, generate appropriate responses based on the Unified Profile, and maintain a repository of previous answers to similar questions. When encountering novel questions that cannot be confidently answered, the system will request user input through the Streamlit interface. This phase creates an intelligent balance between autonomy and necessary human guidance.

*Success Criteria:* The system successfully generates high-quality responses to common application questions, reuses appropriate previous answers, and identifies situations where user input is required.

*Stack:* GPT-4 API for answer generation, custom Python modules for question analysis and similarity detection, LangChain for decision-making logic.

1. **Application Execution and Dynamic Web Navigation:** We'll implement the Actual Applicator agent and enhance the Dynamic Web agent to navigate and interact with arbitrary job application portals. Development creates general-purpose web navigation capabilities that can analyze website structure to identify application forms, input fields, and submission processes. The system will be capable of filling forms, uploading documents, and submitting applications across a wide range of job portals without requiring pre-programmed interaction patterns.

*Success Criteria:* The system can successfully navigate previously unseen job application portals, complete applications with the customized documents and responses, and submit them without user intervention.

*Stack:* Playwright with LangChain/CrewAI agents for autonomous website interaction, GPT-4 API for interpreting website structure and form requirements, custom modules for form filling and document upload.

1. **Background Operation and Optimization:** We'll optimize the system for background operation and improve overall efficiency. Development focuses on creating a more robust database structure using SQLite/DuckDB to replace JSON storage, improving data integrity and query performance. Redis/Celery will provide more sophisticated task queuing and background processing capabilities. We'll implement comprehensive logging and monitoring to track agent performance and identify optimization opportunities. NVIDIA AgentIQ will help evaluate different agent configurations and optimize the overall system architecture.

*Success Criteria:* The system reliably operates in the background, efficiently managing resources and completing tasks without user intervention, while maintaining performance even as the number of tracked job opportunities and applications grows.

*Stack:* SQLite/DuckDB for centralized data management, Redis/Celery for robust background task handling, NVIDIA AgentIQ to evaluate and optimize agent performance.

**Technology Stack Overview**

The system leverages several key technologies chosen specifically for their performance characteristics, compatibility with the M1 architecture, and suitability for agent-based systems:

* **Core Environment:** Python 3.11 in a Conda environment on macOS Sequoia, providing a stable foundation with excellent library support for AI and web automation.
* **Frontend:** Streamlit provides an efficient interface that leverages the M1 Max's capabilities while maintaining development efficiency. It displays job search results, ranked opportunities, and application status while offering interaction points for necessary user input.
* **Agent Frameworks:** LangChain provides the foundation for creating intelligent agents, while CrewAI enables sophisticated inter-agent communication and task delegation. In later phases, NVIDIA AgentIQ will optimize agent performance and evaluate different configurations.
* **Web Automation:** Playwright offers efficient browser automation on ARM architecture, supporting the dynamic web navigation requirements across different platforms and application portals.
* **Semantic Understanding:** Sentence-Transformers enables local embedding model execution for semantic matching between profiles and job descriptions, leveraging the M1 chip's ML acceleration.
* **Language Models:** GPT-4 API provides natural language understanding capabilities for document generation, website analysis, response creation, and application form completion.
* **Background Processing:** APScheduler handles basic task scheduling in early phases, with later migration to Redis/Celery for more robust background operation.
* **Data Storage:** JSON in early phases provides flexibility, with later migration to SQLite/DuckDB for improved data management as the system matures.

This technology stack balances performance requirements with cost considerations, leveraging free and open-source tools where possible while maintaining the capability for background operation and efficient processing.