Multi-Agent Career Advisor

AI-Powered Career Guidance Platform

Built with Microsoft Semantic Kernel

Technologies: Python, Next.js, Azure OpenAI, Cosmos DB, Supabase

Architecture: Multi-Agent System with Semantic Kernel

1. Project Overview

Multi-Agent Career Advisor is an intelligent career guidance platform that leverages multiple specialized AI agents to provide personalized career advice, skills analysis, market research, and application strategies.

The system uses Microsoft Semantic Kernel to orchestrate multiple AI agents, each with specific expertise, working together to deliver comprehensive career guidance.

Key Features:
- Multi-agent AI architecture with 4 specialized agents
- Real-time job market analysis and recommendations
- Personalized skills gap analysis and learning paths
- Application strategy optimization
- Persistent memory using Azure Cosmos DB
- Secure authentication via Supabase (GitHub/Google OAuth)
- Modern React/Next.js frontend with real-time updates

2. Technology Stack

Backend:

- Python 3.13
- Microsoft Semantic Kernel (AI orchestration framework)
- FastAPI (REST API framework)
- Azure OpenAI (GPT-4o-mini for LLM, text-embedding-ada-002 for embeddings)
- Azure Cosmos DB (NoSQL database for persistence)
- Supabase (Authentication - GitHub/Google OAuth)

Fron

- Next.js 15 (React framework)
- TypeScript
- Tailwind CSS (styling)
- Supabase Client (authentication)

AI/M

- Microsoft Semantic Kernel (agent orchestration)
- Azure OpenAI GPT-4o-mini (language model)
- Azure OpenAI text-embedding-ada-002 (vector embeddings)
- Semantic Memory (RAG-based retrieval)

Clou

- Azure OpenAI Service
- Azure Cosmos DB
- Supabase (Auth + Database)

3. System Architecture

The system follows a multi-agent architecture pattern:

```
USER REQUEST --> FastAPI Backend --> Semantic Kernel --> Agent Orchestration
                                          |
             +-----------------------------------------+
             |              |              |
         Career Advisor   Market Researcher   Skills Coach
             |              |              |
             +-----------------------------------------+
                            |
                   Application Strategist
                            |
                   Azure Cosmos DB (Memory)
```

Agent Descriptions:

1. Career Advisor Agent: Main orchestrator that coordinates other agents and provides holistic career guidance based on user profile and goals.

2. Market Researcher Agent: Analyzes job market trends, salary data, industry growth, and demand for specific skills and roles.

3. Skills Coach Agent: Evaluates current skills, identifies gaps, recommends learning resources, and creates personalized development plans.

4. Application Strategist Agent: Optimizes job applications, resume tailoring, interview preparation, and networking strategies.

4. Project Structure

```
multi-agent-career-advisor/
|-- api/
|   |-- main.py                 # FastAPI application
|-- src/
|   |-- kernel_config.py        # Semantic Kernel configuration
|   |-- agents/
|   |   |-- base_agent.py       # Base agent class
|   |   |-- career_advisor.py   # Career Advisor agent
|   |   |-- market_researcher.py # Market Researcher agent
|   |   |-- skills_coach.py     # Skills Coach agent
|   |   |-- application_strategist.py # Application Strategist
|   |-- auth/
|   |   |-- supabase_auth.py    # Supabase authentication
|   |   |-- oauth_manager.py    # OAuth management
|   |-- database/
|   |   |-- cosmos_manager.py   # Cosmos DB operations
|   |-- memory/
|   |   |-- career_memory.py    # Semantic memory management
|   |-- planning/
|   |   |-- career_planner.py   # Career planning logic
|   |-- plugins/
|       |-- job_intelligence/   # Job scraping & analysis
|-- frontend/
|   |-- src/
|       |-- app/
|       |   |-- page.tsx        # Main page
|       |   |-- layout.tsx      # App layout
|       |-- components/
|       |   |-- Dashboard.tsx   # Main dashboard
|       |   |-- Login.tsx       # Login component
|       |-- lib/
|           |-- supabase.ts     # Supabase client
|-- episodes/                   # Development episodes/tutorials
|-- requirements.txt            # Python dependencies
|-- .env.example               # Environment variables template
```

5. Key Components

Semantic Kernel Configuration (kernel_config.py):

- Initializes Azure OpenAI chat completion service
- Configures text embedding service for semantic memory
- Sets up kernel plugins and memory stores
- Manages API connections and authentication

Age

- BaseAgent: Abstract base class with common functionality
- Each agent has specific system prompts and capabilities
- Agents communicate through Semantic Kernel orchestration
- Supports both synchronous and streaming responses

Mem

- Uses Azure Cosmos DB for persistent storage
- Semantic memory for context-aware retrieval
- Stores user profiles, conversation history, and career data
- Vector embeddings for similarity search

Auth

- Supabase Authentication integration
- Supports GitHub OAuth
- Supports Google OAuth
- JWT token validation
- Secure session management

6. API Endpoints

POST /api/chat
- Main chat endpoint for career guidance
- Accepts user messages and returns AI responses
- Orchestrates multiple agents based on query type

POST /api/analyze-skills
- Analyzes user skills and identifies gaps
- Returns skill assessment and recommendations

POST /api/market-research
- Provides job market insights
- Returns trends, salary data, and opportunities

POST /api/application-strategy
- Generates application optimization strategies
- Tailored resume and interview advice

GET /api/user/profile
- Retrieves user profile and history
- Requires authentication

POST /api/auth/callback
- OAuth callback handler
- Processes GitHub/Google authentication

7. Setup Instructions

1. Clone the repository:
```
git clone https://github.com/username/genai-bootcamp.git
cd genai-bootcamp/projects/multi-agent-career-advisor
```

2. Create virtual environment:
```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Python dependencies:
```
pip install -r requirements.txt
```

4. Configure environment variables:
```
cp .env.example .env
# Edit .env with your API keys
```

5. Start the backend:
```
cd api && uvicorn main:app --reload
```

6. Install frontend dependencies:
```
cd frontend && npm install
```

7. Configure frontend environment:
```
cp .env.example .env.local
# Edit .env.local with Supabase credentials
```

8. Start the frontend:
```
npm run dev
```

9. Open http://localhost:3000 in your browser

8. Environment Variables

Backend (.env):

```
AZURE_OPENAI_ENDPOINT=https://your-resource.openai.azure.com/
AZURE_OPENAI_API_KEY=your-api-key
AZURE_OPENAI_DEPLOYMENT_NAME=gpt-4o-mini
AZURE_OPENAI_API_VERSION=2024-10-21
AZURE_OPENAI_EMBEDDING_DEPLOYMENT_NAME=text-embedding-ada-002
COSMOS_CONNECTION_STRING=your-cosmos-connection-string
SUPABASE_URL=https://your-project.supabase.co
SUPABASE_ANON_KEY=your-supabase-anon-key
```

Fror

```
NEXT_PUBLIC_SUPABASE_URL=https://your-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-supabase-anon-key
NEXT_PUBLIC_API_URL=http://localhost:8000
```

9. Skills Demonstrated

This project demonstrates proficiency in:

AI/ML Engineering:
- Multi-agent AI system design and implementation
- Microsoft Semantic Kernel framework
- Prompt engineering and optimization
- RAG (Retrieval Augmented Generation) patterns
- Vector embeddings and semantic search

Cloud & Backend:
- Azure OpenAI Service integration
- Azure Cosmos DB (NoSQL database)
- FastAPI REST API development
- OAuth 2.0 authentication flows
- Microservices architecture

Frontend Development:
- Next.js 15 with App Router
- TypeScript
- Tailwind CSS
- Real-time UI updates
- Authentication state management

DevOps & Best Practices:
- Environment configuration management
- Git version control
- Project documentation
- Code organization and modularity