ChatGPT

Company Research Assistant Implementation Guide

This step-by-step plan outlines how to build a demo-ready **Company Research Assistant** using LangChain's DeepAgents and a Graph-RAG approach. The system has a chat-based UI (with file upload) and a multi-agent backend that scrapes public company data, stores knowledge in ChromaDB, and reasons with sub-agents. The design emphasizes conversational quality, agentic planning, modular prompts, and adaptivity.

1. Architecture Overview and Components

- **Web Chat UI**: A chat interface (e.g. LangChain's Agent Chat UI – a Next.js app) that supports real-time chat, file uploads (PDF/DOC), and context input [1]. It connects to our LangChain agent server.
- **Backend Scraper Service**: Python scripts or tools that gather public data about the target company (e.g. company website, Crunchbase, LinkedIn, news articles) and preprocess documents. These tools feed raw text into our knowledge base.
- **Vector Store (ChromaDB)**: We use Chroma as the semantic memory. All ingested documents (scraped text, user-uploaded files, internal Eightfold docs) are converted to text embeddings and stored in ChromaDB [2] [3]. This enables similarity search (RAG) for relevant context.
- **Knowledge Graph / GraphRAG**: For fallback and analogy-based reasoning, we build or query a simple knowledge graph of companies/industries. Instead of flat text retrieval, a Graph-RAG tool can translate queries into structured graph queries (e.g. via LlamaIndex + Memgraph) [4]. This helps infer insights from a larger, similar company when data on the target is sparse.
- **DeepAgent Core and Sub-Agents**: The main agent orchestrator (using LangChain's DeepAgents framework) handles the user's query. It plans tasks and spawns specialized sub-agents for each analysis component. We incorporate a detailed system prompt, a no-op planning tool (to manage steps), and an internal memory (file system or vector memory) [5] [6].

2. Data Ingestion Module

1. **Scraping Company Data**: Use Python tools (e.g. `requests` / `BeautifulSoup`, Selenium, or APIs) to collect public information on *Company X*. Target sources include the company's website, Crunchbase, LinkedIn (company page), Glassdoor, news articles, and social media. Extract key sections: "About Us", press releases, product descriptions, leadership bios, etc.
2. **Document Loading and Splitting**: Convert scraped text and user-uploaded files (PDFs, DOCX) into plain text. Use LangChain's document loaders (or similar) to chunk long texts into manageable pieces (e.g. 500–1000 token segments).
3. **Embedding and Storage**: Initialize ChromaDB with an embedding model (e.g. OpenAI's embeddings). Add all text chunks as documents:

```
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
```

```
embeddings = OpenAIEmbeddings(model="text-embedding-3-large")
chroma = Chroma(collection_name="company_data",
embedding_function=embeddings)
chroma.add_documents(documents_list)
```

Each document's metadata includes source info (URL, document type) for traceability [3] [2].

4. **Reference Knowledge (Eightfold Info)**: Similarly ingest Eightfold's own knowledge: company pages, product descriptions, case studies, etc. into Chroma. This ensures the agent can retrieve Eightfold-specific details when evaluating fit.
5. **Indexing and Retrieval**: Test retrieval by querying Chroma: e.g. `chroma.similarity_search("Company X overview", k=3)`. This semantic search enables RAG: retrieved snippets are passed to agents for answering questions (consistent with how LangChain vector stores operate [2]).

3. LangChain DeepAgents Setup

1. **Install and Initialize**: Use LangChain's `deepagents` package: `pip install deepagents`. This gives us a DeepAgent base with planning and sub-agent support.
2. **Main Agent System Prompt**: Craft a detailed system prompt that defines the overall assistant role. For example:

   > *"You are an AI Corporate Research Assistant for Eightfold AI. Your goal is to analyze Company X and provide actionable insights for Eightfold's strategy. Use the provided company data and Eightfold knowledge base. You may spawn specialized sub-agents for tasks. If information is missing, ask clarifying questions or search for analog companies."*
   > The prompt should mention conversational style (e.g. polite, concise) and behavior (e.g. confirm steps, explain reasoning).

3. **Planning Tool**: Enable a Todo/Planning tool (a no-op list) so the agent can outline its plan. This helps break down tasks sequentially [7]. For instance, the agent might first plan to "Research Company X overview," then "Evaluate product fit," etc.
4. **File System / Memory**: Use the deepagents' built-in memory or file store to accumulate notes and intermediate results [5]. For example, results from one sub-agent (like a summary of Company X) can be saved and reused by others.
5. **Agent Creation**: Instantiate the main agent with LangChain code. Example skeleton:

```
from deepagents import DeepAgent

main_agent = DeepAgent(
    name="CompanyResearchAgent",
    prompt=system_prompt,
    tools=[chroma_retriever, graph_rag_tool, browser_tool, todo_list],
    subagents=[company_info_agent, product_fit_agent, ...],
    memory=chroma_vector_memory
)
```

Here, `chroma_retriever` is a custom tool that performs similarity search on ChromaDB, and `graph_rag_tool` is a tool for querying the knowledge graph if used.

4. Sub-Agent Modules and Prompts

We define **modular sub-agents**, each focused on one analysis area. Each sub-agent has a clear role, its own prompt template, and access to the vector store. Prompts should specify the task, available context, and desired output format.

- **CompanyOverviewAgent** – *Identify core business and value opportunities.*
- **Role:** Corporate Analyst. Summarize what Company X *does*, its industry, size, and any public goals.
- **Prompt Example:**

```
You are a corporate research analyst for Eightfold. Using the retrieved
documents on Company X, summarize its primary products/services, target
market, and strategic objectives. Also, note any key challenges or needs
mentioned.
Then identify 2–3 ways Eightfold's talent intelligence solutions could add
value (e.g., hiring efficiency, DEI initiatives) to this company.
```

- **Expected Output:** A concise paragraph plus bullet points linking Eightfold's offerings to Company X's needs.

- **ProductFitAgent** – *Map product to goals.*

- **Role:** Product Strategist. Determine how Eightfold's offerings align with Company X's stated goals or needs.
- **Prompt Example:**

```
Acting as an AI product expert, analyze Company X's goals (from the data)
and evaluate how Eightfold's products (e.g., talent acquisition AI,
workforce planning tools) fit those goals. Provide specific examples (e.g.,
"If Company X wants to improve diversity hiring, Eightfold's DEI dashboard
could...").
```

- **Expected Output:** A short report explaining feature-fit, possibly with examples or case analogies.

- **GoalsAgent** – *Extract long-term goals.*

- **Role:** Strategic Advisor. List the company's long-term strategic or hiring goals based on news or filings.
- **Prompt Example:**

```
You are a strategy analyst. Identify Company X's long-term objectives (from
annual reports, press, etc.). For each goal, explain its relevance to HR or
```

```
workforce (e.g., expansion requiring hiring). Present as bullet points with
sources cited if possible.
```

- **Expected Output:** Bulleted goals like “Expand into EMEA by 2026 – implies hiring in EU offices” etc.

- **DeptMappingAgent** – *Find entry points and departments.*

- **Role:** Org Structure Specialist. Identify which departments or roles at Company X are most relevant (e.g., HR, Engineering, Sales) and key decision-makers.
- **Prompt Example:**

```
You are an organizational consultant. Given Company X’s size and industry,
list the top 3 departments (or personas) where Eightfold’s platform would
engage (e.g., “Head of Recruiting, Chief People Officer”). Explain why each
is a good entry point.
```

- **Expected Output:** A list of departments/personas with short rationales.

- **SynergyAgent** – *Analyze partnership synergy.*

- **Role:** Business Development Expert. Examine how Eightfold’s strengths (AI hiring, talent management) create synergy with Company X’s needs (e.g. skill gaps, growth plans).
- **Prompt Example:**

```
Acting as a business development lead, analyze synergies between Company X
and Eightfold. For example, if Company X is scaling rapidly, highlight how
Eightfold’s AI can help. Cite examples or analogies from similar companies
if possible.
```

- **Expected Output:** Narrative points, possibly comparing to industry analogues.

- **PricingAgent** – *Suggest pricing alignment.*

- **Role:** Pricing Analyst. Recommend how Eightfold could tailor its pricing (enterprise vs mid-market models) to fit Company X’s scale and budget.
- **Prompt Example:**

```
You are a pricing strategist. Based on Company X’s approximate size/market
(from data), suggest what level of Eightfold’s pricing plan would fit.
Consider any public info (e.g., funding, revenue) to justify a model (e.g.
“As a startup with $XM funding, they fit an SMB plan”).
```

- **Expected Output:** Recommendation paragraph with justification.

- **ROIAgent** – *Estimate ROI and impact.*

- **Role:** Financial Analyst. Project the timeline for ROI (e.g., 6-month, 1-year gains) from adopting Eightfold.
- **Prompt Example:**

```
Acting as a financial analyst, estimate the ROI timeline if Company X
implements Eightfold. Use general metrics (like % reduction in time-to-
hire, retention improvement). Provide estimates of business impact in 6
months and 1 year (e.g., “Reducing turnover by 5% could save $Y”).
```

- **Expected Output:** A brief forecast (6-mo and 1-yr) with reasoning or simple calculations.

Each sub-agent's prompt should allow it to retrieve relevant context via the ChromaDB tool. For example, include in the prompt: *"Use the company data retrieved for Company X via Chroma (provided as context)."* Agents can call a custom `chroma_retriever(query)` tool to fetch snippets from the vector store, which they then incorporate in answers.

5. GraphRAG and Similar-Company Inference

- **Analog Company Strategy**: If Company X is small or obscure, we fall back to finding a larger *analogous company*. Build a simple knowledge graph of known companies (e.g., using LlamaIndex property graph or a CSV of companies by industry/size).
- **Graph-RAG Tool**: Create a `GraphRAGTool` that translates natural questions into graph queries (e.g., Cypher). For instance, given Company X's industry, the tool finds the top companies in that sector. This uses the Graph-RAG pattern [4].
- **Similarity Agent**: A sub-agent "SimilarityAgent" with a prompt like:

```
Use the knowledge graph of industries to find 2 larger companies similar to
Company X (by industry, size, or products). Explain why they are similar
and what insights about their strategy might apply to Company X.
```

  The agent would use `graph_rag_tool.query(…)` to get candidates (e.g., "TechCrunch sector = SaaS, $50M ARR").
- **Inference from Analogs**: Once analog companies are identified, other sub-agents can re-run their analyses on the analogs (or summary thereof). For example, the **CompanyOverviewAgent** can append: *"Given analog [Company Y], note any additional patterns."* This helps infer missing info.

6. Integration of Modules and Data Flow

1. **User Query**: The user enters "Research Company X" in the chat UI. If the user uploads files or text, those are added to Chroma and memory first.
2. **Main Agent Planning**: The main DeepAgent loads the user's query and begins by generating a plan (using the Todo tool). It may list steps like: "Fetch data, run CompanyOverviewAgent, run ProductFitAgent, synthesize results."

3. **Sub-Agent Execution**: For each step, the main agent calls a sub-agent. The sub-agent receives its prompt (as above) along with context retrieved from Chroma. It runs an LLM call to generate an answer.
4. **Memory and Looping**: Outputs from sub-agents are stored in the agent's memory (file store or variables). The main agent can use these to avoid re-computation and to summarize at the end.
5. **GraphRAG Calls**: If a sub-agent detects "insufficient data", it can invoke the GraphRAG tool or the SimilarityAgent. For example:

```
if chroma.search("Company X overview") yields few docs:
    analog_info = SimilarityAgent.run("Find similar company to X")
    // Use analog_info as additional context
```

6. **Final Synthesis**: The main agent assembles sub-agent outputs into a coherent report. It structures the answer as a conversational explanation or summary (per the user's request), possibly numbered or bullet-pointed.

7. Conversational Quality and Adaptability

- **Clarifying Questions**: If the user's query is vague (e.g. "Company X" with no context), the system should ask clarifying questions. For example, the main agent's prompt can include: "If key info is missing (like industry or location), ask the user." This ensures adaptability in conversation.
- **Tone and Coherence**: In system prompts for all agents, enforce a professional yet conversational tone. Use bullet lists or numbered steps in responses for clarity.
- **Memory of Context**: The agent should reference user-provided context explicitly. E.g., "According to the user-uploaded PDF on Company X's roadmap..." demonstrates it is using given context.
- **Dynamic Flow**: Allow the agent to jump between tasks based on findings. For instance, if CompanyOverviewAgent finds an unexpected company focus, the main agent might insert a new subtask to research that area. This is enabled by the planning loop (todo tool) in DeepAgents [7].
- **Error Handling**: If a tool call fails (e.g., no internet result), the agent's prompt instructs it to proceed differently ("If web search fails, use internal knowledge base").

8. Implementation Tips and Integration

- **LangChain DeepAgents**: Use the `create_agent()` API or similar to tie together the system prompt, tools, and subagents. Each sub-agent can be an `Agent` object with its own prompt template.
- **Tools Integration**: Wrap Chroma's similarity search as a callable tool (e.g. `Tool(name="ChromaSearch", func=lambda q: chroma.similarity_search(q, k=3))`) that agents can invoke. Similarly, wrap your scraper or a generic web-search tool to fetch fresh info if needed.
- **Graph-RAG Integration**: If using a real knowledge graph (like Memgraph via LlamaIndex), implement a `GraphRAGTool` as shown in [4†L25-L33]. This tool's LLM-driven query should output structured answers that are then fed back to the agent.
- **Prompt Templates**: Keep prompts concise but detailed. Use placeholders (e.g. `{company_name}`) to dynamically insert the target company. You can define a template with LangChain's PromptTemplate for reusability.

- **Modularity**: Each sub-agent's code can be a function or class. For example, `CompanyOverviewAgent = Agent(name="Overview", prompt=..., tools=[ChromaSearch])`. This makes it easy to add/remove modules.
- **Graph of Thoughts**: To implement a **graph-of-thought** style reasoning, you can store intermediary conclusions in the agent's state or memory, and let the agent revisit previous nodes. The DeepAgents framework's virtual file system can store such reasoning "nodes" [5].

9. Example Prompt Templates

Here are sample prompt templates (in quotes) for clarity:

- **Main Orchestrator Prompt:**

  > "You are a Company Research Assistant. The user asks about **{company_name}**. Use the knowledge base (Chroma search results and any user-provided files) to perform a deep analysis. Plan your steps (listing them), then execute each by calling the appropriate sub-agent. Finally, present a concise summary of findings. Use a helpful, professional tone."

- **CompanyOverviewAgent Prompt:**

```
You are a corporate analyst. Summarize {company_name}'s business model, industry, and key objectives based on the provided data. Then identify 2–3 areas where Eightfold AI's solutions (e.g., AI hiring, talent management) could deliver value to this company.
```

- **ProductFitAgent Prompt:**

```
You are a product strategist. Given {company_name}'s goals and challenges (from data), evaluate how Eightfold's products align with those needs. Provide specific examples or scenarios.
```

- **GoalsAgent Prompt:**

```
You are a strategic consultant. List the long-term goals or strategic initiatives of {company_name} (from articles and filings). For each goal, note any implications for hiring or workforce planning.
```

- **SimilarityAgent (GraphRAG) Prompt:**

```
You are an industry research specialist with access to a company knowledge graph. Find 2 larger, similar companies to {company_name} (match by industry and size). For each, briefly explain the similarity.
```

10. Ensuring Evaluation Criteria

- **Conversational Quality:** The UI and prompts should encourage natural dialogue. Agents will use clear language, bullets/lists, and check for user comprehension. If the user asks follow-ups, the agent can call relevant sub-agents again or explain previous results.
- **Agentic Behavior:** Using LangChain DeepAgents, the system actively plans tasks, spawns sub-agents, and uses tools (Chroma search, web search, etc.) in a loop [7] [6]. Each sub-agent has a distinct role and full autonomy over its answer.
- **Technical Implementation:** The guide outlines concrete code integration points (Chroma setup, DeepAgents API, tool definitions) and uses industry tools (e.g. LlamaIndex/Graph, Chroma, LangChain) with citations [3] [2].
- **Intelligence and Adaptability:** By combining RAG (vector search) and GraphRAG, the agent can handle sparse data by analogies [4]. The modular prompts allow easy refinement. The use of memory and planning makes the agent adaptable to multi-step reasoning and user feedback.

11. Testing and Iteration

1. **Dry Run**: Start with a well-known company (e.g., "Acme Corp") and step through each agent's output. Verify that retrieval yields relevant context from Chroma.
2. **Prompt Tuning**: Adjust prompt phrasing and instruction specificity to improve answer accuracy. Add examples or style guidelines as needed.
3. **Human Feedback**: Engage in the chat UI as a user: try vague queries, supply additional docs, request clarifications. Ensure the agent responds appropriately (asks questions or refines answers).
4. **Edge Cases**: Test with a tiny startup (to trigger GraphRAG fallback) and a large enterprise (for direct RAG). Confirm the SimilarityAgent logic kicks in when direct info is limited.

By following these steps and modularizing each component, you will have a demo-ready **Company Research Assistant**: a multi-agent system with a rich chat interface that deeply investigates target companies and provides strategic recommendations for Eightfold AI.

Sources: Concepts and tools are based on LangChain DeepAgents documentation [5] [6], LangChain UI guide [1], and Graph-RAG methodologies [4]. Vector store usage is informed by ChromaDB tutorials [3] [2].

---

1 Agent Chat UI - Docs by LangChain
https://docs.langchain.com/oss/python/langchain/ui

2 Vector stores - Docs by LangChain
https://docs.langchain.com/oss/python/integrations/vectorstores

3 Embeddings and Vector Databases With ChromaDB – Real Python
https://realpython.com/chromadb-vector-database/

4 Beyond LLMs: Building a Graph-RAG Agentic Architecture for 70% Faster ECM Automation | by Rahul Kumar | Nov, 2025 | Medium
https://medium.com/@hellorahulk/beyond-llms-building-a-graph-rag-agentic-architecture-for-70-faster-ecm-automation-299b05d026fb

5 6 7 Deep Agents
https://blog.langchain.com/deep-agents/