Agent Framework and Orchestration

CrewAI can be used as the orchestration framework for Mandrake's Deep-Research Agent. CrewAI is an open-source, Python-native library designed for modular, multi-agent workflows. It enables us to define specialized agents (e.g., trait parser, database searcher, evidence validator, Neo4j integrator), each with its own memory and toolset. This modular approach makes the workflow transparent, explainable, and easy to extend. CrewAI's lightweight design ensures compatibility with Mandrake's requirements (Python 3.10+, laptop runtime) and supports robust tool-calling and agent memory.

LLM Choice and Reasoning Patterns

Either Gemini or OpenAI's ChatGPT can be used as the primary LLMs. Both of these would be a good choice for proper structured output and good accuracy. Both can handle complex prompts and can give outputs in the desired format (e.g., JSON). Both of these have good tool calling support and low latency as well. However, these can't be run locally. Llama, for example would be a good option for running locally and would have lesser cost as there won't be a need for purchasing tokens, but there would be higher latency with llama.

With RAG, the agent first retrieves relevant documents or entities from external bioinformatics databases like Ensembl Plants or Uniprot based on the user's trait and species query. The retrieved data is then passed to the LLM, which summarizes, extracts, and prioritizes gene trait associations, ensuring that all outputs are grounded in verifiable sources. In parallel, function/tool-calling enables the LLM to invoke structured Python functions—such as Cypher queries for Neo4j or REST API calls to external resources directly from within its reasoning process.

Tool Calling Design

The agent uses the official Python Neo4j driver to run Cypher queries for reading from and writing to the knowledge graph. It should be able to call external APIs like the REST API of Ensembl Plants or UniProt API and extract information and update it in the Neo4j graph using Cypher queries that will be specified to it. From the user query, LLM will extract the information to be used as keywords for the external APIs. Then, external APIs will be called and the new information will be added as new relations to the Neo4j graph through Cypher commands.

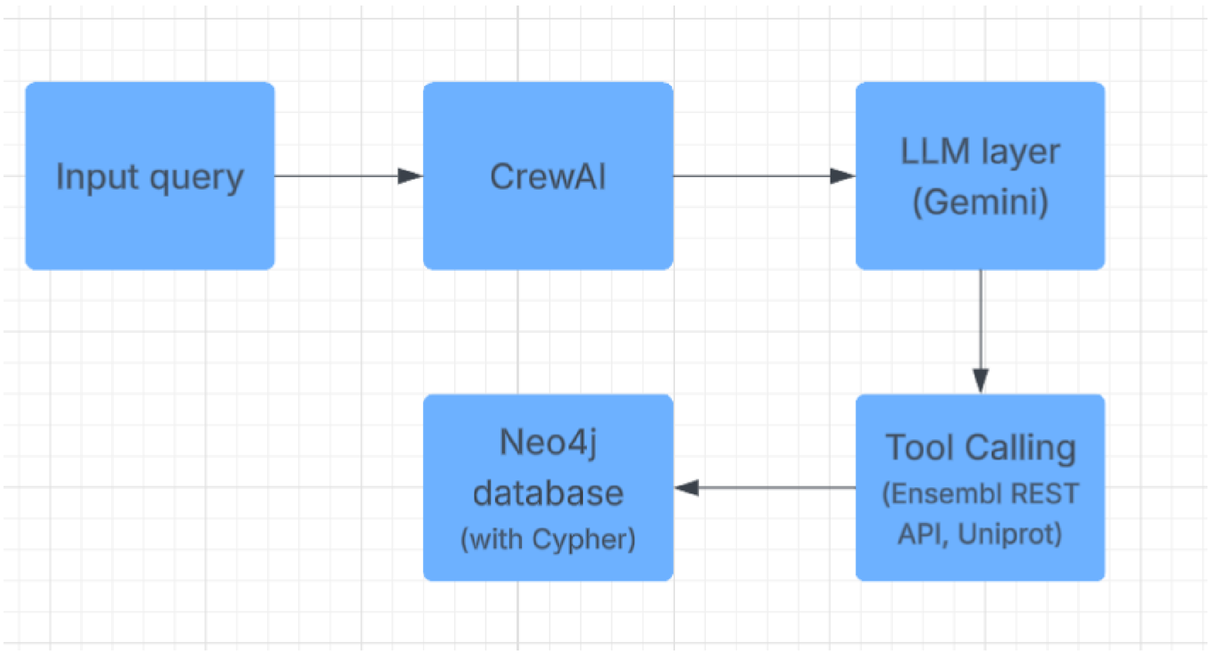
Cloud vs on-prem benefits

Token cost per 1000 tokens for Gemini is \$0.0001 -\$0.0004 and for ChatGPT, it is \$0.002-\$0.008. For on-prem models like Llama, this cost would be 0. However, Gemini and ChatGPT have much higher accuracy because of the scale of the model and have much lower latency because of the hardware constraints while using on-prem models and the extra cold start time needed for models like Llama when used on-prem. Cloud LLMs like ChatGPT and Gemini also provide very optimised infrastructure due to which speed and accuracy are higher. Overall, because of higher accuracy and lower latency, Gemini could be a good model to use.

Scaling path and failure recovery

By making sure that each task is modular and stateless, parallel execution can be enabled and can be done with multiple processes or agents. Since CrewAI is used with multiple agents, workloads can be distributed evenly.

If a database call or an API call fails, the agent can be made to attempt a fixed number of retries to prevent failure. If a particular API is unavailable, a fallback API can be used to get the information. Keeping a log of the errors will also help while looking into repeated failures.


Input query
CrewAI
LLM layer (Gemini)
Tool Calling (Ensembl REST API, Uniprot)
Neo4j database (with Cypher)