High-Level and Low-Level Design of Multi-Agent System

# High-Level Design (HLD)

1. Overview  
The project is built around a multi-agent system that dynamically answers industry-specific AI and ML use case queries. The system involves several components, such as data retrievers, pre-trained models for natural language processing, and an interactive interface for the user. The core goal of the system is to collect, process, and present relevant data in response to specific queries, such as 'AI use cases in healthcare' or 'AI applications in retail.'  
  
2. System Components  
2.1 User Interface (Gradio)  
- Role: Acts as the front-end for user interaction.  
- Functionality: Users can input queries related to AI and ML applications across industries, and the results are displayed dynamically.  
  
2.2 Core Agents  
- Role: Perform specialized tasks like querying the data, performing AI/ML-based reasoning, and returning responses to users.  
- Agents:  
 - Resource Asset Collection Agent: Collects datasets, research papers, and other resource assets.  
 - Industry Research Agent: Retrieves data related to AI/ML use cases from industry blogs and resources.  
 - Use Case Generation Agent: Generates specific AI use case suggestions and answers from data.  
  
2.3 Retriever (TavilySearch API)  
- Role: Responsible for searching and fetching relevant data from sources like LeewayHertz, IBM, and Google Cloud.  
- Functionality: This component allows agents to query external sources for up-to-date information.  
  
2.4 Pre-trained Language Models (HuggingFaceHub)  
- Role: Process natural language queries and generate appropriate responses.  
- Functionality: These models are fine-tuned for AI/ML-related queries to generate relevant and coherent answers.

## High-Level System Flow

1. User Input: The user types a query on the Gradio Interface.  
2. Query Routing: The query is sent to the Core Agents, which determine which agent is best suited to handle it.  
3. Data Retrieval: The agents use the Retriever to search for relevant content based on the query.  
4. AI Processing: The Pre-trained Language Models process the search results, generating a natural language response.  
5. User Output: The response is displayed on the Gradio Interface for the user.

# Low-Level Design (LLD)

1. Data Flow  
1. User Request: The system receives the user query via the Gradio interface.  
2. Routing the Query:  
- The query is passed to a routing function, which analyzes the query to determine which agent to invoke.  
- For example, if the query asks for 'AI use cases in healthcare,' the system invokes the Industry Research Agent.  
3. Data Retrieval: The agent uses the TavilySearch API to fetch the latest data.  
4. Model Processing: The retrieved data is passed to the HuggingFaceHub models.  
5. Response Output: The processed data is formatted into a response and returned to the user via the Gradio interface.

## 2. Detailed Component Design

2.1 Gradio Interface  
- Functionality:  
 - Receives the user’s input via a text box.  
 - Displays the response after processing.  
- Components:  
 - gr.Interface: Used to create the interface for receiving input and showing output.  
Example Code:  
```python  
import gradio as gr  
def user\_query\_function(query):  
 # Process the query and return the result  
 return agent.process\_query(query)  
gr.Interface(fn=user\_query\_function, inputs='text', outputs='text').launch()  
```

2.2 Core Agents  
- Role: Handle specific tasks related to processing the query.  
Example:  
The Industry Research Agent will:  
- Use TavilySearch API to fetch the latest industry insights.  
- Process the results and return a structured response.  
Example Code for an Agent:  
```python  
class IndustryResearchAgent:  
 def \_\_init\_\_(self, llm, retriever):  
 self.llm = llm  
 self.retriever = retriever  
 def process\_query(self, query):  
 # Step 1: Use the retriever to search the relevant data  
 data = self.retriever.search(query)  
 # Step 2: Process the data using LLM  
 response = self.llm.process(data)  
 return response  
```

2.3 Retriever (TavilySearch API)  
- Role: Fetches relevant articles, research papers, or data based on a user query.  
Example Code:  
```python  
from tavily\_search import TavilySearchAPIRetriever  
retriever = TavilySearchAPIRetriever(api\_key=api\_key)  
class IndustryResearchAgent:  
 def \_\_init\_\_(self, retriever):  
 self.retriever = retriever  
 def process\_query(self, query):  
 # Get relevant data from Tavily  
 results = self.retriever.search(query)  
 return results  
```

2.4 Pre-trained Language Models (HuggingFaceHub)  
- Role: Generates responses using large pre-trained language models from HuggingFace.  
Example Code:  
```python  
from langchain\_community.llms import HuggingFaceHub  
huggingface\_token = 'YOUR\_HUGGINGFACE\_API\_TOKEN'  
llm = HuggingFaceHub(repo\_id='bert-base-uncased', huggingfacehub\_api\_token=huggingface\_token)  
class IndustryResearchAgent:  
 def \_\_init\_\_(self, llm, retriever):  
 self.llm = llm  
 self.retriever = retriever  
 def process\_query(self, query):  
 # Retrieve relevant data  
 data = self.retriever.search(query)  
 # Generate response from HuggingFace model  
 response = self.llm(data)  
 return response  
```

## 3. Key System Components Interactions

Gradio Interface ➔ Query Routed to Agent ➔ Retriever fetches data ➔ LLM processes and returns output ➔ Results displayed on Gradio Interface.  
  
Dataflow:  
Query: User → Gradio  
Processing: Gradio → Agent → Retriever → HuggingFaceHub LLM  
Output: HuggingFaceHub → Agent → Gradio → User.