Sample Query Results for extract_intent LLM Pipeline

Query 1: Basic Chat-Type Input

```
query = "Capital of India"
result = extract_intent(query)
print("🔍 Result:")
print(json.dumps(result, indent=2))
```

Output:

```
{
  "type": "chat",
  "output": "The capital city of India is New Delhi. Is there anything else
you'd like to talk about?"
}
```

Query 2: Task-Type Input with Recognized Parameters

```
query = "What is My and bobs attendence of last month"
result = extract_intent(query)
print("🔍 Result:")
print(json.dumps(result, indent=2))
```

Output:

```
{
  "type": "task",
  "raw_output": "intent=attendance, params=employee1=My, month=last_month,
employee2=Bob."
}
```

Parsed Usage in MCP Server:

- **Intent:** `attendance`
- **Params:** `{ employee1: "My", employee2: "Bob", month: "last_month" }`
- These parameters can be used directly to trigger a lookup into an internal or SQL-based attendance database within the MCP system.

Query 3: Chat Query Highlighting Model Limitations

```
query = "Can this llm model be like the user specific? if i mention my name as alex then whenever a query is made where i dont give my name directly but use words like me or my or i then it directly guess it with name alex"
result = extract_intent(query)
print("🔍 Result:")
print(json.dumps(result, indent=2))
```

Output:

```
{
  "type": "chat",
  "output": "I see what you mean. However, the current model isn't designed to understand user-specific contexts based on names like that. It can't make assumptions about the user based on previous interactions without being explicitly told. But we're always looking for ways to improve, so let's keep this in mind for future developments."
}
```

Limitation Highlighted:

- The current stateless nature of the LLM means it cannot track identity or user-specific context like pronouns.
- This illustrates a potential future improvement area requiring persistent user-session memory or context caching.

Strategy Behind Robust Prompting:

To mitigate hallucination and ensure structured responses, the pipeline uses a **controlled prompt format**:

```
You're an assistant that classifies user messages into either:
- A chat type if it's general conversation
- A task type if it's requesting structured action or information

Respond in one of the following formats only:

type: chat
output: <your natural reply>

OR
```

```
type: task
output: intent=<intent_name>, params=<key1=value1, key2=value2, ...>
```

This strategy enforces formatting discipline in the LLM output, reducing ambiguity and simplifying downstream parsing logic. Even in cases of hallucinated or noisy completions, regular expression-based extraction and parameter fallback recovery make the system resilient.

Conclusion:

- Mistral-7B with prompt-based routing provides a reliable base for LLM-driven intent parsing.
- Parameters obtained through structured extraction are immediately usable in MCP backend systems.
- However, context-aware personalization still remains a limitation, underlining the need for more advanced or fine-tuned LLMs in production scenarios.