class QueryUnderstandingAgent:

def \_\_init\_\_(self):

self.model = genai.GenerativeModel('gemini-1.5-flash')

def \_\_call\_\_(self, state: AgentState) -> AgentState:

"""Understand user query and extract dimensional intent"""

print("Executing QueryUnderstandingAgent...")

query = state["user\_query"]

prompt = f"""

Analyze this business query about utilization metrics and extract the intent.

Return ONLY a JSON object with no other text.

Query: "{query}"

JSON Structure:

{{

"primary\_metric": "utilization" or "nbl" or "shrinkage",

"scope": "overall" or "breakdown" or "comparison",

"breakdown\_dimension": "team" or "edl" or "none",

"entities": ["list", "of", "specific", "items"],

"filters": ["current", "top", "bottom", "trend"],

"question\_type": "value" or "breakdown" or "identification"

}}

Examples:

- "What is NBL percentage?" → {{"primary\_metric": "nbl", "scope": "overall", "breakdown\_dimension": "none", "entities": [], "filters": ["current"], "question\_type": "value"}}

- "Show me top performing teams" → {{"primary\_metric": "utilization", "scope": "breakdown", "breakdown\_dimension": "team", "entities": [], "filters": ["top"], "question\_type": "identification"}}

- "Compare GB and CS utilization" → {{"primary\_metric": "utilization", "scope": "comparison", "breakdown\_dimension": "team", "entities": ["GB", "CS"], "filters": ["current"], "question\_type": "comparison"}}

- "Show me utilization by EDL" → {{"primary\_metric": "utilization", "scope": "breakdown", "breakdown\_dimension": "edl", "entities": [], "filters": ["current"], "question\_type": "breakdown"}}

- "Which teams need attention?" → {{"primary\_metric": "utilization", "scope": "breakdown", "breakdown\_dimension": "team", "entities": [], "filters": ["bottom"], "question\_type": "identification"}}

"""

try:

response = self.model.generate\_content(prompt)

response\_text = response.text.strip()

print(f"Raw LLM Response: {response\_text}")

# Try to extract JSON from the response

json\_text = self.\_extract\_json(response\_text)

print(f"Extracted JSON: {json\_text}")

intent\_data = json.loads(json\_text)

print(f"Parsed Intent Data: {intent\_data}")

# Validate and ensure required fields

intent\_data = self.\_validate\_intent(intent\_data)

state["intent"] = intent\_data

state["execution\_path"].append("query\_understanding")

except Exception as e:

print(f"LLM parsing failed: {e}")

print(f"Using fallback for query: {query}")

state["intent"] = self.\_get\_fallback\_intent(query)

state["execution\_path"].append("query\_understanding\_fallback")

return state

def \_extract\_json(self, text: str) -> str:

"""Extract JSON from LLM response text"""

# Remove markdown code blocks

text = text.replace('```json', '').replace('```', '')

text = text.strip()

# Find JSON object start and end

start\_idx = text.find('{')

end\_idx = text.rfind('}') + 1

if start\_idx != -1 and end\_idx != -1:

return text[start\_idx:end\_idx]

else:

return text # Return as-is if no braces found

def \_validate\_intent(self, intent\_data: Dict) -> Dict:

"""Validate and fill missing fields in intent data"""

required\_fields = {

'primary\_metric': 'utilization',

'scope': 'overall',

'breakdown\_dimension': 'none',

'entities': [],

'filters': ['current'],

'question\_type': 'value'

}

validated = {}

for field, default in required\_fields.items():

if field in intent\_data:

validated[field] = intent\_data[field]

else:

validated[field] = default

print(f"Missing field {field}, using default: {default}")

return validated

def \_get\_fallback\_intent(self, query: str) -> Dict:

"""Enhanced fallback intent detection"""

query\_lower = query.lower()

# Pattern matching for common queries

if any(word in query\_lower for word in ['nbl', 'non-billable', 'non billable']):

return {

"primary\_metric": "nbl",

"scope": "overall",

"breakdown\_dimension": "none",

"entities": [],

"filters": ["current"],

"question\_type": "value"

}

elif any(word in query\_lower for word in ['shrinkage', 'shrink']):

return {

"primary\_metric": "shrinkage",

"scope": "overall",

"breakdown\_dimension": "none",

"entities": [],

"filters": ["current"],

"question\_type": "value"

}

elif any(word in query\_lower for word in ['top', 'best', 'high', 'above']):

if 'team' in query\_lower or 'edl' in query\_lower or 'performing' in query\_lower:

return {

"primary\_metric": "utilization",

"scope": "breakdown",

"breakdown\_dimension": "team",

"entities": [],

"filters": ["top"],

"question\_type": "identification"

}

else:

return {

"primary\_metric": "utilization",

"scope": "overall",

"breakdown\_dimension": "none",

"entities": [],

"filters": ["current"],

"question\_type": "value"

}

elif any(word in query\_lower for word in ['attention', 'need', 'worst', 'low', 'bottom', 'improve', 'struggling']):

if 'team' in query\_lower or 'edl' in query\_lower:

return {

"primary\_metric": "utilization",

"scope": "breakdown",

"breakdown\_dimension": "team",

"entities": [],

"filters": ["bottom"],

"question\_type": "identification"

}

else:

return {

"primary\_metric": "utilization",

"scope": "overall",

"breakdown\_dimension": "none",

"entities": [],

"filters": ["current"],

"question\_type": "value"

}

elif any(word in query\_lower for word in ['compare', 'vs', 'versus', 'compared to']):

entities = []

if 'gb' in query\_lower:

entities.append('GB')

if 'cs' in query\_lower:

entities.append('CS')

if 'pk' in query\_lower:

entities.append('PK')

return {

"primary\_metric": "utilization",

"scope": "comparison",

"breakdown\_dimension": "team",

"entities": entities,

"filters": ["current"],

"question\_type": "comparison"

}

elif any(word in query\_lower for word in ['team', 'edl']) and 'utilization' in query\_lower:

dimension = 'edl' if 'edl' in query\_lower else 'team'

return {

"primary\_metric": "utilization",

"scope": "breakdown",

"breakdown\_dimension": dimension,

"entities": [],

"filters": ["current"],

"question\_type": "breakdown"

}

elif 'by' in query\_lower and ('team' in query\_lower or 'edl' in query\_lower or 'location' in query\_lower):

dimension = 'edl' if 'edl' in query\_lower else 'team'

return {

"primary\_metric": "utilization",

"scope": "breakdown",

"breakdown\_dimension": dimension,

"entities": [],

"filters": ["current"],

"question\_type": "breakdown"

}

else:

# Default to overall utilization

return {

"primary\_metric": "utilization",

"scope": "overall",

"breakdown\_dimension": "none",

"entities": [],

"filters": ["current"],

"question\_type": "value"

}