HEALTH-BRIDGE Backend Code Review & Fixes

Date: January 25, 2026 **Reviewed by:** Claude Code Assistant

Table of Contents

1. Initial Error

When running `python backend/test_integration.py`, the following error occurred:

```
pydantic_core._pydantic_core.ValidationError: 1 validation error for Agent
tools.0
  Input should be a valid dictionary or instance of BaseTool
  [type=model_type, input_value=Tool(name='Retrieve Guide... at 0x...),
input_type=Tool]
```

Cause: Framework incompatibility between LangChain tools and CrewAI agents.

2. Root Cause Analysis

File	Framework Used	Issue
`tools.py`	LangChain (`Tool`, `StructuredTool`)	Incompatible with CrewAI
`agents.py`	CrewAI (`Agent`)	Expects CrewAI's `BaseTool`

CrewAI's `Agent` class expects tools to be either:

- A dictionary, OR
- An instance of CrewAI's `BaseTool`

But the code was passing LangChain's `Tool` objects.

3. Changes Made

3.1 tools.py - Converted to CrewAI Format

Before:

```
from langchain_core.tools import Tool, StructuredTool

retrieve_guidelines = Tool.from_function(
    func=_retrieve_guidelines,
    name="Retrieve Guidelines",
    ...
)
```

After:

```
from crewai.tools import tool

@tool("Retrieve Guidelines")
def retrieve_guidelines(query: str) -> str:
    """Search medical guidelines..."""
    # ... implementation
```

Additional improvements:

- Added lazy initialization for `SemanticMemory` and `VectorRetriever`
- Added error handling with try/except blocks
- Fixed KeyError risk using `.get()` method

3.2 agents.py - Added Gemini LLM Configuration

Before:

```
from crewai import Agent
# Used default OpenAI
```

After:

```
import os
from crewai import Agent, LLM

class HealthBridgeAgents:
    def __init__(self):
        model = os.getenv("LLM_MODEL", "gemini/gemini-1.5-flash")
        temperature = float(os.getenv("LLM_TEMPERATURE", "0.7"))
        self.llm = LLM(model=model, temperature=temperature)
```

3.3 chat.py - Added user_id Interpolation

Changes:

- Task descriptions now include `{user_id}` placeholder
- `kickoff()` now passes `inputs={"user_id": user_id}`
- Commented out unused model imports

3.4 test_integration.py - Secure API Key Loading

Before:

```
os.environ["OPENAI_API_KEY"] = "sk-proj-dummy-key"
```

After:

```
from dotenv import load_dotenv
load_dotenv()

if not os.getenv("GEMINI_API_KEY"):
    raise ValueError("GEMINI_API_KEY not set...")
```

3.5 semantic_memory.py - Fixed Timestamp & Compatibility

Changes:

- Added `datetime` import
- Fixed empty timestamp: `datetime.now().isoformat()`
- Fixed Python 3.8 compatibility: `List[str]` instead of `list[str]`

3.6 retriever.py - Fixed Relevance Score

Before:

```
"relevance_score": 1 - distance  # Can be negative!
```

After:

```
"relevance_score": max(0.0, 1.0 - distance)  # Clamped to non-negative
```

---

4. Code Review Summary

Issues Found & Fixed

Priority	File	Issue	Status
CRITICAL	tools.py	LangChain/CrewAI incompatibility	FIXED
CRITICAL	test_integration.py	Placeholder API key	FIXED
CRITICAL	semantic_memory.py	Empty timestamp default	FIXED
RUNTIME	tools.py	KeyError risk on `m['text']`	FIXED
RUNTIME	semantic_memory.py	Python 3.9+ syntax	FIXED
RUNTIME	retriever.py	Negative relevance_score	FIXED
LOGIC	agents.py	Unused imports	FIXED (commented)
LOGIC	chat.py	Unused imports	FIXED (commented)
IMPROVE	agents.py	Hardcoded LLM model	FIXED (env var)
IMPROVE	tools.py	No error handling	FIXED

5. Final File States

tools.py

```
from crewai.tools import tool
from typing import Optional

_memory: Optional["SemanticMemory"] = None
_retriever = None
_retriever_initialized = False

def get_memory():
    """Lazy initialization for SemanticMemory."""
    global _memory
    if _memory is None:
        from app.core.memory.semantic_memory import SemanticMemory
        _memory = SemanticMemory()
    return _memory

def get_rag_retriever():
    """Lazy initialization for RAG retriever."""
    global _retriever, _retriever_initialized
    if not _retriever_initialized:
        try:
            from app.core.rag.retriever import get_retriever
            _retriever = get_retriever()
        except Exception as e:
            print(f"Warning: RAG Retriever init failed: {e}")
            _retriever = None
        _retriever_initialized = True
```

```
    return _retriever

@tool("Retrieve Guidelines")
def retrieve_guidelines(query: str) -> str:
    """Search medical guidelines..."""
    retriever = get_rag_retriever()
    if not retriever:
        return "RAG Unavailable (Init Failed)"
    try:
        results = retriever.search_guidelines(query, k=3)
        if not results:
            return "No relevant guidelines found."
        formatted = "\n".join([f"- {r['content']} (Score:
{r['relevance_score']:.2f})" for r in results])
        return f"Guideline Results:\n{formatted}"
    except Exception as e:
        return f"RAG Error: {e}"

@tool("Recall User Memory")
def recall_memory(user_id: str, query: str) -> str:
    """Search past conversations..."""
    try:
        memory = get_memory()
        results = memory.recall_memories(user_id, query)
        if not results:
            return "No specific memories found."
        return "\n".join([f"- {m.get('text', 'N/A')}" for m in results])
    except Exception as e:
        return f"Memory recall error: {e}"

@tool("Save User Constraint")
def save_constraint(user_id: str, constraint: str) -> str:
    """Save a permanent constraint..."""
    try:
        memory = get_memory()
        memory.store_memory(user_id, constraint, metadata={"type": "constraint"})
        return "Constraint saved."
    except Exception as e:
        return f"Failed to save constraint: {e}"
```

Setup Instructions

1. **Create `.env` file in `backend/` directory:**

```
GEMINI_API_KEY=your-actual-api-key
LLM_MODEL=gemini/gemini-1.5-flash
LLM_TEMPERATURE=0.7
```

2. **Install dependencies:**

```
pip install crewai litellm python-dotenv chromadb sentence-transformers
```

3. **Run the test:**

```
python backend/test_integration.py
```

Environment Variables

Variable	Default	Description
`GEMINI_API_KEY`	(required)	Your Gemini API key
`LLM_MODEL`	`gemini/gemini-1.5-flash`	LLM model to use
`LLM_TEMPERATURE`	`0.7`	LLM temperature
`CHROMA_DB_PATH`	`./data/chroma_memory`	ChromaDB storage path
`CHROMA_PERSIST_DIR`	`./data/chroma`	RAG vector store path

Notes

- The Pydantic models (`Profile`, `RiskAssessment`, etc.) are defined but not enforced. To use them, uncomment the imports and add `output_pydantic=ModelClass` to Task definitions.
- The `user_id` is passed via CrewAI's `kickoff(inputs={...})` for task interpolation.
- Memory tools require the `user_id` to be passed by the LLM agent based on task instructions.