## **🎯 MILESTONE 1: INTEGRATION-ONLY STEPS**

### **PHASE 1: File Placement**

# Navigate to client template

cd app/backend/app/my\_agent/

# Remove existing sample files

rm -rf \*

# Copy your entire search\_summarize\_agent structure

cp -r /path/to/your/search\_summarize\_agent/\* ./

**PHASE 2: Critical Integration Changes**

**2.1 Update LangGraph Configuration**

{

"dependencies": ["."],

"graphs": {

"search\_agent": "./app/my\_agent/main.py:graph"

},

"env": ".env",

"python\_version": "3.12"

}

**2.2 Update Your main.py for LangSmith Export**

**# ADD THIS AT THE END OF YOUR EXISTING main.py**

# Export the graph for LangSmith Studio

graph = create\_search\_summarize\_workflow() # or whatever your workflow function is called

# If you want to test with sample data in LangSmith

async def demo\_search\_workflow():

"""Demo function for LangSmith testing"""

# Your sample search requirements

sample\_search\_requirements = {

"verification\_needed": True,

"primary\_objectives": [

"Verify current employment status",

"Confirm medical specialties",

"Validate workplace affiliation"

],

"geographic\_region": "IT",

"confidence\_threshold": 0.85,

"individual\_info": {

"firstName": "PAOLO",

"lastName": "CORVISIERI",

"workplace": "DISTRETTO SANITARIO FIUMICINO",

"specialties": ["18", "29"],

"country": "IT"

}

}

# Run your existing workflow function

return await process\_search\_request(sample\_search\_requirements) # or your function name

**2.3 Update Dependencies in Client Template**

*# MERGE YOUR requirements.txt INTO THE EXISTING dependencies ARRAY*

dependencies = [

*# ... keep all existing client dependencies ...*

*# ADD YOUR SEARCH AGENT DEPENDENCIES FROM requirements.txt*

*# Copy each line from your requirements.txt here*

"langgraph>=0.0.26",

"langchain-core>=0.1.0",

"langchain-openai>=0.0.5",

*# ... add all your other dependencies from requirements.txt*

]

**2.4 Update Environment Variables**

***# COPY ALL VARIABLES FROM YOUR .env FILE TO CLIENT'S .env***

***# Keep all existing client variables and ADD yours***

***# Your Search Agent Configuration***

**AZURE\_OPENAI\_API\_KEY=your-api-key-here**

**AZURE\_OPENAI\_ENDPOINT=https://your-resource.openai.azure.com/**

**AZURE\_OPENAI\_API\_VERSION=2024-02-15-preview**

**AZURE\_OPENAI\_DEPLOYMENT\_NAME=gpt-4**

***# Add any other environment variables from your .env file***

### ***PHASE 3: LangSmith Setup***

#### ***3.1 LangSmith Environment Setup***

# **LangSmith Integration for Milestone 1**

## **1. Install LangSmith CLI**

pip install langsmith

## **2. Add LangSmith Environment Variables to .env**

# Add to app/backend/.env

LANGCHAIN\_TRACING\_V2=true

LANGCHAIN\_ENDPOINT=https://api.smith.langchain.com

LANGCHAIN\_API\_KEY=your-langsmith-api-key

LANGCHAIN\_PROJECT=vr-search-summarize-milestone1

## **3. Run LangGraph Server for LangSmith UI**

# Navigate to app/backend

cd app/backend

# Install langgraph-cli if not installed

pip install langgraph-cli

# Start LangGraph server (this enables LangSmith UI)

langgraph dev

## **4. Access LangSmith UI**

* Open browser to: http://localhost:8123
* You'll see your search\_agent graph
* Can provide inputs directly in the UI
* Visualize each step of execution

## **5. Test Your Agent in LangSmith UI**

1. Navigate to the "search\_agent" graph
2. Click "Run"
3. Provide input JSON like:

{

"search\_requirements": {

"verification\_needed": true,

"primary\_objectives": ["Verify employment", "Confirm specialties"],

"geographic\_region": "IT",

"individual\_info": {

"firstName": "PAOLO",

"lastName": "CORVISIERI",

"workplace": "DISTRETTO SANITARIO FIUMICINO"

}

}

}

1. Watch the workflow execute step-by-step

**PHASE 4: Testing Integration**

# **Testing Your Integration**

## **1. Test Local Execution**

cd app/backend

# Test your agent directly

python -m app.my\_agent.main

# Or if you have a test function

python -c "

import asyncio

from app.my\_agent.main import demo\_search\_workflow

result = asyncio.run(demo\_search\_workflow())

print(result)

"

## **2. Test LangGraph Configuration**

# Verify langgraph.json is valid

langgraph build

# Start development server

langgraph dev

## **3. Test in LangSmith UI**

1. Go to http://localhost:8123
2. Select "search\_agent"
3. Click "Playground" or "Run"
4. Input your test data
5. Watch execution in real-time

## **4. Verify All Tools Are Visible**

In LangSmith UI, you should see:

* Tool selection step
* Each tool execution (Italy, France, Hospital, LinkedIn, Web)
* Summarization step
* Final results

## **5. Test Different Scenarios**

Create different input scenarios:

* Italian medical professional
* French medical professional
* Hospital/workplace verification
* Different confidence thresholds

### **PHASE 5: Integration Checklist**

# **🎯 Milestone 1 Integration Checklist**

## **✅ File Structure**

* [ ] Copy your entire search\_summarize\_agent/ to app/backend/app/my\_agent/
* [ ] Verify all your files are in place:
  + [ ] main.py
  + [ ] config.py
  + [ ] agents/search\_summarize\_agent.py
  + [ ] utils/state\_models.py & helpers.py
  + [ ] prompts/search\_summarize\_prompts.py
  + [ ] tools/ directory with all 5 tools

## **✅ Configuration Updates**

* [ ] Update app/backend/langgraph.json to point to your main.py
* [ ] Merge your requirements.txt into app/backend/pyproject.toml
* [ ] Merge your .env into app/backend/.env
* [ ] Add LangSmith environment variables

## **✅ Code Integration**

* [ ] Add graph = create\_search\_summarize\_workflow() export to your main.py
* [ ] Ensure your workflow function is properly exposed
* [ ] Verify all imports work with client's structure

## **✅ LangSmith Setup**

* [ ] Install langgraph-cli
* [ ] Set up LangSmith API key
* [ ] Run langgraph dev successfully
* [ ] Access UI at http://localhost:8123

## **✅ Testing**

* [ ] Test local execution of your agent
* [ ] Verify LangSmith UI shows your graph
* [ ] Test input/output in LangSmith UI
* [ ] Verify all tools are visible and executable
* [ ] Test different search scenarios

## **✅ Demo Preparation**

* [ ] Prepare sample input data for demo
* [ ] Test end-to-end workflow in LangSmith
* [ ] Verify all agent steps are visible
* [ ] Ensure search results and summary are properly displayed

## **🚀 Ready for Milestone 1 Demo!**

### **Demo Flow:**

1. Open LangSmith UI (http://localhost:8123)
2. Show the search\_agent graph structure
3. Input sample search requirements
4. Execute workflow step-by-step
5. Show tool selection and execution
6. Display final summarization results
7. Demonstrate different search scenarios

## **🎯 INTEGRATION SUMMARY**

**For Milestone 1, you only need to:**

1. **Copy your code into app/backend/app/my\_agent/**
2. **Update 3 configuration files: langgraph.json, pyproject.toml, .env**
3. **Add graph export to your main.py**
4. **Install LangSmith CLI and run langgraph dev**
5. **Demo in LangSmith UI at http://localhost:8123**

**No code changes needed - just integration configuration! Your existing Search & Summarize Agent will work perfectly with LangSmith visualization.**

**Would you like me to clarify any specific integration step?**

**NO! For Milestone 1 (Search & Summarize Agent only), follow this simplified mapping:**

## **🎯 MILESTONE 1 SPECIFIC MAPPING**

**YOUR SEARCH\_SUMMARIZE\_AGENT/ → CLIENT TEMPLATE LOCATION**

**├── main.py → app/backend/app/my\_agent/agent.py**

**├── config.py → MERGE with app/backend/app/config.py**

**├── requirements.txt → MERGE with app/backend/pyproject.toml**

**├── .env → MERGE with app/backend/.env**

**├── agents/**

**│ └── search\_summarize\_agent.py → app/backend/app/my\_agent/agents/search\_summarize\_agent.py**

**├── utils/**

**│ ├── state\_models.py → app/backend/app/my\_agent/utils/state.py**

**│ └── helpers.py → app/backend/app/my\_agent/utils/helpers.py**

**├── prompts/**

**│ └── search\_summarize\_prompts.py → app/backend/app/my\_agent/prompts/search\_summarize\_prompts.py**

**└── tools/ → app/backend/app/my\_agent/tools/**

**├── italy\_trusted\_sources.py**

**├── france\_trusted\_sources.py**

**├── hospital\_sources.py**

**├── linkedin\_professional.py**

**└── untrusted\_web\_search.py**

## **🔍 COMPLETE FILE/FOLDER ANALYSIS FOR MILESTONE 1**

**Let me check every single folder/file in the client template and tell you what needs changes:**

### **📁 app/backend/ (Root Level)**

| **File/Folder** | **Changes Required?** | **Action** |
| --- | --- | --- |
| **langgraph.json** | **✅ YES** | **Update to point to your agent** |
| **pyproject.toml** | **✅ YES** | **Add your dependencies** |
| **.env.example** | **✅ YES** | **Add your env variables** |
| **Dockerfile** | **❌ NO** | **Keep as-is** |
| **openapi.json** | **❌ NO** | **Auto-generated, no changes** |
| **uv.lock** | **❌ NO** | **Auto-generated, no changes** |

### **📁 app/backend/app/ (Main App)**

| **Folder** | **Changes Required?** | **Action** |
| --- | --- | --- |
| **my\_agent/** | **✅ YES** | **Replace entirely with your code** |
| **api/** | **❌ NO** | **Keep existing (no API for Milestone 1)** |
| **core/** | **❌ NO** | **Keep existing** |
| **tests/** | **❌ NO** | **Keep existing (optional: add your tests)** |
| **config.py** | **⚠️ MAYBE** | **Only if your config.py has unique settings** |
| **dependencies.py** | **❌ NO** | **Keep existing** |
| **main.py** | **❌ NO** | **Keep existing (no API changes)** |
| **models.py** | **❌ NO** | **Keep existing** |

### **📁 app/backend/app/api/**

**❌ NO CHANGES - Keep all existing files (routes, dependencies, etc.)**

### **📁 app/backend/app/core/**

**❌ NO CHANGES - Keep all existing files (db, documents, evaluation, security, storage)**

### **📁 Other Folders**

| **Folder** | **Changes Required?** |
| --- | --- |
| **docs/** | **❌ NO** |
| **guards/** | **❌ NO** |
| **rag/** | **❌ NO** |

## **🎯 EXACT CHANGES NEEDED:**

### **1. Environment File (.env vs .env.example)**

**# Copy .env.example to .env first**

**cp app/backend/.env.example app/backend/.env**

**# Then ADD your variables to app/backend/.env**

**# Your Azure OpenAI settings**

**AZURE\_OPENAI\_API\_KEY=your-key**

**AZURE\_OPENAI\_ENDPOINT=your-endpoint**

**# ... etc**

### **2. Only 3 Files Need Updates:## 📋 COMPLETE INTEGRATION CHECKLIST:**

**✅ REPLACE: app/backend/app/my\_agent/ (entire folder)  
 ✅ UPDATE: app/backend/langgraph.json  
 ✅ UPDATE: app/backend/pyproject.toml  
 ✅ CREATE: app/backend/.env (from .env.example + your variables)  
 ❌ DON'T TOUCH: Everything else!**

## **🚀 SUMMARY:**

* **4 changes total (replace my\_agent + update 3 files)**
* **No API, core, dependencies, models changes needed**
* **No Docker, openapi.json changes needed**
* **Yes, .env.example → .env + add your variables**

**This keeps Milestone 1 simple - just your agent + minimal integration!**

**Perfect! I can see your complete code. The tools aren't needed for LangSmith integration. Here are your exact files with only the minimal LangSmith changes added:**

## **1. Updated main.py## 2. Updated search\_summarize\_agent.py## 3. All Other Files (NO CHANGES NEEDED)**

**Your other files are already perfect for LangSmith:**

* **state\_models.py ✅ Already has messages field**
* **helpers.py ✅ No changes needed**
* **config.py ✅ No changes needed**
* **batch\_search\_processor.py ✅ No changes needed**
* **test\_search\_agent.py ✅ No changes needed**
* **All init.py files ✅ No changes needed**

## **🎯 SUMMARY OF CHANGES MADE:**

1. **main.py:**
   * **Added HumanMessage import**
   * **Added initial message to process\_search\_request**
   * **Added graph export at the end**
   * **Added demo\_search\_langsmith() function**
2. **search\_summarize\_agent.py:**
   * **Added emoji messages in each method for better LangSmith visualization**
   * **No logic changes - just visual improvements**

**That's it! Your code is now ready for LangSmith integration with minimal changes that preserve all your existing functionality.**

## **🎯 INTEGRATION-SPECIFIC CHANGES NEEDED**

### **1. Import Path Updates (CRITICAL)**

# =============================================================================

# IMPORT PATH UPDATES - REPLACE IN YOUR FILES

# =============================================================================

# 1. In main.py (now agent.py) - REPLACE THESE LINES:

# OLD:

from utils.state\_models import SearchAgentState, SearchWorkflowStatus

from agents.search\_summarize\_agent import SearchAndSummarizeAgent

# NEW:

from app.my\_agent.utils.state\_models import SearchAgentState, SearchWorkflowStatus

from app.my\_agent.agents.search\_summarize\_agent import SearchAndSummarizeAgent

# 2. In search\_summarize\_agent.py - REPLACE THESE LINES:

# OLD:

from config import Config

from utils.helpers import safe\_json\_parse

from utils.state\_models import SearchAgentState, SearchWorkflowStatus

from prompts.search\_summarize\_prompts import (

TOOL\_SELECTION\_PROMPT,

SUMMARIZATION\_PROMPT

)

# NEW:

from app.config import Config # Use client's config

from app.my\_agent.utils.helpers import safe\_json\_parse

from app.my\_agent.utils.state\_models import SearchAgentState, SearchWorkflowStatus

from app.my\_agent.prompts.search\_summarize\_prompts import (

TOOL\_SELECTION\_PROMPT,

SUMMARIZATION\_PROMPT

)

### **2. Configuration Integration**

# **Configuration Integration**

## **Option 1: Merge your Config class into client's config.py**

# Add to app/backend/app/config.py (in the InitSettings class):

class InitSettings(BaseSettings):

# ... existing client settings ...

# ADD YOUR CONFIG VALUES:

azure\_openai\_api\_key: str = ""

azure\_openai\_endpoint: str = ""

azure\_openai\_api\_version: str = "2024-02-15-preview"

azure\_openai\_deployment\_name: str = "gpt-4"

max\_retries: int = 3

timeout\_seconds: int = 300

tool\_timeout: int = 60

min\_confidence\_threshold: float = 0.70

target\_confidence\_threshold: float = 0.85

## **Option 2: Keep your config.py and update imports**

* Keep your config.py in app/my\_agent/config.py
* Change import in search\_summarize\_agent.py to: from app.my\_agent.config import Config

### **3. Client Template Configuration Updates**

# **Client Template Configuration Updates**

## **1. Update app/backend/langgraph.json**

{

"dependencies": ["."],

"graphs": {

"search\_agent": "./app/my\_agent/agent.py:graph"

},

"env": ".env",

"python\_version": "3.12"

}

## **2. Update app/backend/pyproject.toml**

dependencies = [

# ... keep all existing dependencies ...

# ADD from your requirements.txt:

"langgraph>=0.0.26",

"langchain-core>=0.1.0",

"langchain-openai>=0.0.5",

"python-dotenv>=1.0.0",

"pydantic>=2.0.0",

"aiohttp>=3.8.0",

"tenacity>=8.2.0",

]

## **3. Update app/backend/.env (copy from .env.example first)**

# Copy template first

cp .env.example .env

# Then add your variables:

AZURE\_OPENAI\_API\_KEY=your-key

AZURE\_OPENAI\_ENDPOINT=your-endpoint

AZURE\_OPENAI\_API\_VERSION=2024-02-15-preview

AZURE\_OPENAI\_DEPLOYMENT\_NAME=gpt-4

# LangSmith for UI

LANGCHAIN\_TRACING\_V2=true

LANGCHAIN\_API\_KEY=your-langsmith-key

LANGCHAIN\_PROJECT=search-agent-milestone1

## **📋 COMPLETE INTEGRATION CHECKLIST**

# **🎯 Complete Integration Checklist - Step by Step**

## **✅ STEP 1: File Placement (DONE)**

* [x] Your code is placed in app/backend/app/my\_agent/

## **✅ STEP 2: Import Path Updates (DO THIS NOW)**

* [ ] Update main.py (now agent.py) imports
* [ ] Update search\_summarize\_agent.py imports
* [ ] Update utils/\_\_init\_\_.py imports (if needed)
* [ ] Update agents/\_\_init\_\_.py imports (if needed)

## **✅ STEP 3: Configuration Integration**

* [ ] Choose: Merge config OR keep separate
* [ ] Update import in search\_summarize\_agent.py

## **✅ STEP 4: Client Template Updates**

* [ ] Update app/backend/langgraph.json
* [ ] Update app/backend/pyproject.toml
* [ ] Create app/backend/.env from .env.example
* [ ] Add your environment variables

## **✅ STEP 5: Install Dependencies**

cd app/backend

pip install -e .

# or

pip install langgraph-cli langsmith

## **✅ STEP 6: Test Integration**

cd app/backend

python -c "from app.my\_agent.agent import graph; print('Import successful!')"

## **✅ STEP 7: Start LangSmith UI**

cd app/backend

langgraph dev

## **✅ STEP 8: Access LangSmith UI**

* Open: http://localhost:8123
* Click on "search\_agent"
* Test with sample input

## **🚨 CRITICAL ISSUES TO WATCH:**

1. **Import paths** - Most common integration issue
2. **Environment variables** - Must be set correctly
3. **Graph export** - Must be named graph in agent.py

## **🎯 SUMMARY - WHAT YOU NEED TO DO:**

**Only 3 things are critical for integration:**

1. **Fix import paths** in your files (2-3 files to update)
2. **Update 3 client config files** (langgraph.json, pyproject.toml, .env)
3. **Install dependencies and test**

That's it! Your code logic doesn't need to change - just these integration-specific updates to make it work within the client's template structure.

**The import path updates are the most critical** - without those, nothing will work.

Would you like me to provide the exact updated files with the corrected import paths?