Agent System & LLM Recommendations for Freight Forwarding

🤖 Recommended Agent Systems

1. LangChain (Recommended ⭐)

Best for: Production-ready, flexible conversational AI

Pros:

- Excellent for conversational flows with memory
- Built-in document handling and retrieval
- Easy integration with multiple LLMs
- Great for structured workflows (quote process)
- Active community and extensive documentation

Use Case Fit:

- Perfect for your step-by-step quote process
- Handles document validation and classification
- Memory management for conversation context
- Tool/function calling for API integration

python

```
from langchain.agents import AgentExecutor, create_openai_functions_agent
from langchain.memory import ConversationBufferMemory
from langchain.tools import Tool
```

---

2. LlamaIndex

Best for: Document-heavy operations

Pros:

- Excellent for document retrieval and analysis
- Great for validating shipping documents
- RAG (Retrieval Augmented Generation) capabilities
- Good for compliance checking

Use Case Fit:

- Document classification (invoice, packing list, etc.)
- Extracting info from shipping documents
- Policy/regulation compliance checks

---

3. AutoGen (Microsoft)

Best for: Multi-agent scenarios

Pros:

- Multiple specialized agents working together
- Good for complex workflows
- Agent-to-agent communication

Use Case Fit:

- One agent for quote creation
- Another for document validation
- Another for pricing/logistics

---

4. Rasa (Open Source)

Best for: On-premise deployment

Pros:

- Complete control over data
- Privacy-first (important for business documents)
- Good NLU capabilities
- Can run entirely offline

Cons:

- Steeper learning curve
- Requires more training data

🧠 Recommended LLMs

Open Source Options (Light & Efficient)

1. Llama 3.1 8B (Recommended ⭐⭐⭐)

Best overall balance

- **Size:** 8B parameters
- **RAM Required:** ~6-8GB
- **Strengths:**
  - Excellent reasoning for business logic
  - Good instruction following
  - Multi-lingual support
  - Commercial use allowed
- **Perfect for:** Your entire workflow

python

```
# Using Ollama (easiest deployment)
from langchain_community.llms import Ollama

llm = Ollama(model="llama3.1:8b")
```

---

2. Mistral 7B v0.3

Best for lightweight deployment

- **Size:** 7B parameters
- **RAM Required:** ~5-6GB
- **Strengths:**
  - Very fast inference
  - Good at structured outputs (JSON)
  - Excellent for classification tasks
- **Perfect for:** Document classification, validation

---

3. Phi-3 Mini (3.8B)

Best for ultra-light deployment

- **Size:** 3.8B parameters
- **RAM Required:** ~3-4GB
- **Strengths:**
  - Runs on CPU efficiently
  - Surprisingly good reasoning
  - Great for specific tasks
- **Perfect for:** Basic conversation, form filling

---

4. Gemma 2 9B

Best for accuracy

- **Size:** 9B parameters
- **RAM Required:** ~8-10GB
- **Strengths:**
  - Excellent instruction following
  - Good at structured tasks
  - Strong safety features
- **Perfect for:** Complex validation logic

💡 My Recommendation for Your Use Case

Best Setup: LangChain + Llama 3.1 8B

Why this combination:

1. **LangChain** provides:
   - Conversation memory for context
   - Easy tool integration (your API calls)
   - Structured output parsing

    - Document loaders for file uploads
2. **Llama 3.1 8B** provides:
    - Good reasoning for validation logic
    - Fast inference (~1-2 sec response)
    - Can run on modest hardware
    - Commercial-friendly license

🚀 Implementation Architecture

```
User Input
    ↓
LangChain Agent (Llama 3.1 8B)
    ↓
    ├── Conversation Memory
    ├── Document Validator Tool
    ├── API Integration Tool
    ├── Quote Calculator Tool
    └── Email Sender Tool
    ↓
Response to User
```

📦 Deployment Options

Option 1: Ollama (Easiest)

bash

```
# Install Ollama
curl -fsSL https://ollama.ai/install.sh | sh

# Pull model
ollama pull llama3.1:8b

# Run locally
ollama serve
```

Option 2: vLLM (Production)

- Better performance
- Batching support

- API server included

Option 3: HuggingFace Transformers

- Maximum control
- Custom optimizations
- Good for fine-tuning

💰 Cost Comparison

Solution	Hardware Cost	Runtime Cost	Setup Time
Llama 3.1 8B (Local)	GPU: $500–1000	$0/month	1–2 hours
GPT-4o Mini (API)	$0	~$50–200/month	30 mins
Claude Sonnet (API)	$0	~$100–300/month	30 mins

🎯 Specific Recommendations by Requirement

For High Accuracy (Business Critical):

- **LangChain + GPT-4o Mini (API)** or **Claude Haiku**
- More reliable, less hallucination
- ~$0.15 per 1M tokens

For Privacy (Sensitive Documents):

- **LangChain + Llama 3.1 8B (Self-hosted)**
- Complete data control
- GDPR compliant

For Cost Efficiency:

- **LangChain + Phi-3 Mini**
- Can run on CPU
- Zero ongoing costs

For Speed:

- **LangChain + Mistral 7B**
- Fast inference
- Good balance

🔧 Alternative: Hybrid Approach

Best of both worlds:

Simple Tasks (Form filling, greetings)
→ Llama 3.1 8B (Local)

Complex Tasks (Document validation, pricing)
→ GPT-4o Mini (API fallback)

This gives you:

- 90% cost savings
- Fallback for complex cases
- Best user experience

📚 Quick Start Code

python

```python
# Install dependencies
# pip install langchain langchain-community ollama chromadb

from langchain_community.llms import Ollama
from langchain.agents import AgentExecutor, create_react_agent
from langchain.memory import ConversationBufferMemory
from langchain.tools import Tool
from langchain.prompts import PromptTemplate

# Initialize LLM
llm = Ollama(
    model="llama3.1:8b",
    temperature=0.7
)

# Create tools for freight operations
def validate_documents(docs):
    """Validate uploaded shipping documents"""
    required = ['invoice', 'packing_list']
    # Your validation logic
    return {"valid": True, "missing": []}

tools = [
    Tool(
        name="DocumentValidator",
        func=validate_documents,
        description="Validates shipping documents"
    )
]

# Create agent with memory
memory = ConversationBufferMemory(
    memory_key="chat_history",
    return_messages=True
)

# Your freight forwarding agent is ready!
```

🎓 Learning Resources

- **LangChain Docs:** https://python.langchain.com/
- **Ollama Setup:** https://ollama.ai/
- **Llama 3.1 Guide:** https://huggingface.co/meta-llama

- **LangChain Agents Tutorial:** https://python.langchain.com/docs/modules/agents/

🤔 Final Verdict

For Production: LangChain + Llama 3.1 8B (via Ollama)

- Self-hosted, fast, accurate, cost-effective
- Perfect for your freight forwarding workflow

For Quick MVP: LangChain + GPT-4o Mini (API)

- Fastest to deploy
- Very reliable
- Switch to self-hosted later