Full Implementation Architecture

Industry-Grade AI Commerce Assistant (RAG + Tools + LCEL + OpenAI)

This document defines a **production-ready folder structure**, modular architecture, service boundaries, and component responsibilities for a scalable AI Commerce Assistant.

This system integrates:

- OpenAI LLMs
- OpenAI embeddings
- LangChain LCEL pipelines
- Chroma VectorDB
- RAG retrieval
- Tool calling (Weather + Serper + Utilities)
- Hybrid fallback strategy
- Clean separation of concerns

---

1. High-Level System Architecture

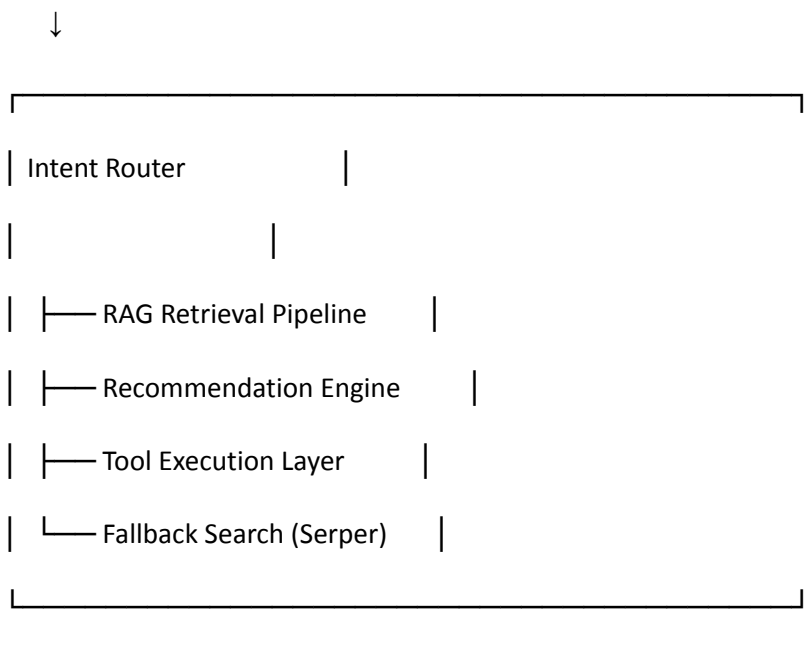
Client (Web / Mobile / Chat)

↓

API Gateway (FastAPI)

↓

Application Layer (Orchestrator)



---

2. Clean Architecture Layers

2.1 Presentation Layer

- FastAPI endpoints
- Request validation
- Response formatting

2.2 Application Layer

- Orchestration logic
- Intent routing
- Pipeline execution
- Tool loop handling

2.3 Domain Layer

- Recommendation rules
- Retrieval policies
- Ranking logic
- Business constraints

2.4 Infrastructure Layer

- VectorDB (Chroma)
- OpenAI API integration
- Serper API integration
- Weather API integration

---

3. Production Folder Structure

ai-commerce-assistant/

│

├── app/

│ ├── main.py

│ ├── config.py

```
│   ├── dependencies.py
│   │
│   ├── api/
│   │   ├── routes.py
│   │   ├── schemas.py
│   │   └── middleware.py
│   │
│   ├── core/
│   │   ├── orchestrator.py
│   │   ├── router.py
│   │   ├── intent_classifier.py
│   │   └── response_builder.py
│   │
│   ├── rag/
│   │   ├── ingestion.py
│   │   ├── chunking.py
│   │   ├── embeddings.py
│   │   ├── vector_store.py
│   │   ├── retriever.py
│   │   └── rag_pipeline.py
│   │
│   ├── recommendations/
│   │   ├── recommender.py
│   │   ├── ranking.py
│   │   └── personalization.py
```

```
│   │
│   ├── tools/
│   │   ├── weather.py
│   │   ├── serper_search.py
│   │   ├── calculator.py
│   │   ├── interest.py
│   │   └── tool_registry.py
│   │
│   ├── llm/
│   │   ├── openai_client.py
│   │   ├── prompts.py
│   │   ├── lcel_pipelines.py
│   │   └── tool_executor.py
│   │
│   ├── database/
│   │   ├── product_loader.py
│   │   └── metadata_filters.py
│   │
│   ├── monitoring/
│   │   ├── logging.py
│   │   ├── metrics.py
│   │   └── tracing.py
│   │
│   └── utils/
│       ├── helpers.py
```

```
│       └── constants.py
│
├── data/
│   ├── raw_catalog.csv
│   └── chroma_db/
│
├── tests/
│   ├── test_rag.py
│   ├── test_tools.py
│   ├── test_router.py
│   └── test_api.py
│
├── scripts/
│   ├── ingest_catalog.py
│   └── rebuild_embeddings.py
│
├── docker/
│   ├── Dockerfile
│   └── docker-compose.yml
│
├── requirements.txt
├── .env
└── README.md
```

---

4. Component-Level Design

4.1 API Layer (`api/`)

routes.py

Responsibilities:

- Expose `/chat`
- Accept user query
- Call orchestrator
- Return structured response

POST /chat

{

"user_id": "123",

"query": "Recommend waterproof shoes under 4000"

}

4.2 Orchestration Layer (`core/`)

orchestrator.py

Responsibilities:

- Manage state
- Invoke intent classifier

- Route to appropriate pipeline
- Execute tool loop if required

Pseudo-flow:

```
def handle_request(query):
  intent = classify_intent(query)

  if intent == "product_search":
    return rag_pipeline.run(query)

  if intent == "recommendation":
    return recommender.run(query)

  if intent == "weather":
    return tool_executor.run(query)

  return fallback_handler.run(query)
```

4.3 RAG Module (`rag/`)

ingestion.py

- Load product catalog
- Normalize text

chunking.py

- RecursiveCharacterTextSplitter
- Configurable chunk size

embeddings.py

- OpenAI embedding wrapper

vector_store.py

- Chroma initialization
- Persistence handling

retriever.py

- Similarity search
- Metadata filtering

rag_pipeline.py

LCEL-based retrieval pipeline.

```
rag_chain = (
  {
    "context": retriever | format_docs,
    "question": RunnablePassthrough()
  }
  | prompt
  | llm
  | StrOutputParser()
)
```

4.4 Recommendation Engine (`recommendations/`)

recommender.py

- Applies business rules
- Calls retriever
- Applies ranking

ranking.py

- Score normalization
- Hybrid ranking

personalization.py

- User embedding generation
- Similarity scoring

4.5 Tool Layer (`tools/`)

Each tool is isolated.

Example:

weather.py

- OpenWeather API call

serper_search.py

- Serper API wrapper

tool_registry.py

```
TOOL_MAP = {
  "get_weather": get_weather,
  "web_search": web_search,
  "calculate_interest": calculate_interest
}
```

This avoids hardcoding in orchestrator.

4.6 LLM Layer (`llm/`)

openai_client.py

- Centralized OpenAI config
- Temperature control
- Model selection

prompts.py

- All prompt templates
- Strict grounding rules

lcel_pipelines.py

- RAG pipelines

- Tool pipelines
- Hybrid pipelines

tool_executor.py

- Automatic tool loop
- Safety validation

4.7 Monitoring Layer

- Structured logging
- Tool invocation logs
- Retrieval score logging
- Token usage tracking
- Latency measurement

Enterprise systems require observability.

---

5. Deployment Architecture

Load Balancer

↓

FastAPI Service (Docker)

↓

ChromaDB (Persistent Volume)

↓

External APIs:

- OpenAI
- Serper
- OpenWeather

---

6. CI/CD Strategy

- Unit tests for tools
- Integration tests for RAG
- LLM response validation
- Retrieval quality evaluation
- Dockerized builds
- Environment-based config

---

7. Scaling Considerations

- Stateless API servers
- Shared vector database
- Caching frequent queries
- Tool result caching
- Rate limiting
- Retry logic for APIs

---

8. Security Considerations

- API key management via environment variables
- Tool execution whitelisting
- Input validation
- Prompt injection guardrails
- Metadata filtering enforcement

---

9. Extension Roadmap

Future expansion:

- LangGraph stateful orchestration
- Multi-agent architecture
- Cross-encoder reranking
- Evaluation framework
- Feedback-driven learning loop

---

10. Final Summary

This folder structure and architecture:

- Enforces separation of concerns

- Enables modular scalability
- Supports tool calling and RAG
- Uses LCEL pipelines
- Supports hybrid retrieval
- Is deployable in enterprise environments
- Allows observability and governance