Asist Data Flow and Class Architecture

Data Flow Description

1. User Query Entry Point

- User submits query through Vue.js frontend
- Request hits /api/query or /api/query/stream endpoint
- User authentication verified via require_auth()
- User profile extracted including authorization level for ITAR compliance

2. Cache Check Layer

- Query normalized and checked against in-memory cache
- If cache hit: Return cached answer immediately (sub-second response)
- If cache miss: Proceed to orchestration pipeline
- Cache uses TTL (Time To Live) for freshness (default: 1 hour)

3. ITAR Compliance Gate

- Query analyzed for controlled information
- User authorization level checked against content requirements
- If unauthorized: Return denial message with recommendations
- If authorized: Continue to answer generation

4. State Orchestration Pipeline

INIT → INTENT → ENTITY → ANSWER → COMPLETE

Phase 1: Intent Analysis

- Query passed to IntentAgent
- Ollama LLM classifies intent (definition, distinction, authority, etc.)
- Confidence score calculated
- Historical patterns from HIL feedback applied

Phase 2: Entity Extraction (3-Phase Process)

- **Pattern Matching**: SAMM entity patterns checked (organizations, programs, authorities)
- **Knowledge Graph**: Local TTL knowledge graph queried
- **Database Integration**:
    - Cosmos Gremlin for graph relationships
    - Vector DB for document similarity search
    - Vector DB TTL for knowledge base search
- Entities enriched with context, definitions, relationships
- Overall confidence score calculated

Phase 3: Answer Generation

- Context compiled from all sources (entities, text sections, chat history, documents)
- Intent-specific system message created
- Ollama LLM generates answer with optimized prompts
- Multi-pass validation applied
- Quality enhancements: acronym expansion, section citations, terminology fixes
- Answer scored against quality metrics

5. Response Caching

- Generated answer cached with metadata
- LRU eviction if cache size limit reached
- Cache statistics updated (hits, misses, hit rate)

6. Response Delivery

- Standard mode: Complete JSON response with answer + metadata
- Streaming mode: Server-Sent Events with token-by-token delivery
- Frontend receives and renders response

Class Architecture

Core Classes

DatabaseManager

- Manages connections to 3 databases
- Methods: query_cosmos_graph(), query_vector_db(), query_vector_db_ttl()
- Handles embedding model for semantic search
- Implements connection retry and error handling

IntentAgent

- Classifies query intent using Ollama
- Maintains learned patterns from HIL feedback
- Methods: analyze_intent(), update_from_hil(), update_from_trigger()

IntegratedEntityAgent

- 3-phase entity extraction
- Database integration for enrichment
- Methods: extract_and_retrieve(), _extract_entities_enhanced(), _populate_enhanced_context()
- Maintains dynamic knowledge base and custom entities

EnhancedAnswerAgent

- Intent-optimized answer generation
- Quality scoring and enhancement
- Methods: generate_answer(), _generate_with_validation(), _enhance_answer_quality()
- Stores answer corrections and templates

SimpleStateOrchestrator

- Coordinates workflow between agents
- Manages state transitions
- Methods: process_query(), update_agents_from_hil(), update_agents_from_trigger()
- Implements LangGraph-style state management

Supporting Structures

AgentState (TypedDict)

- Shared state across workflow
- Fields: query, chat_history, intent_info, entity_info, answer, execution_steps

WorkflowStep (Enum)

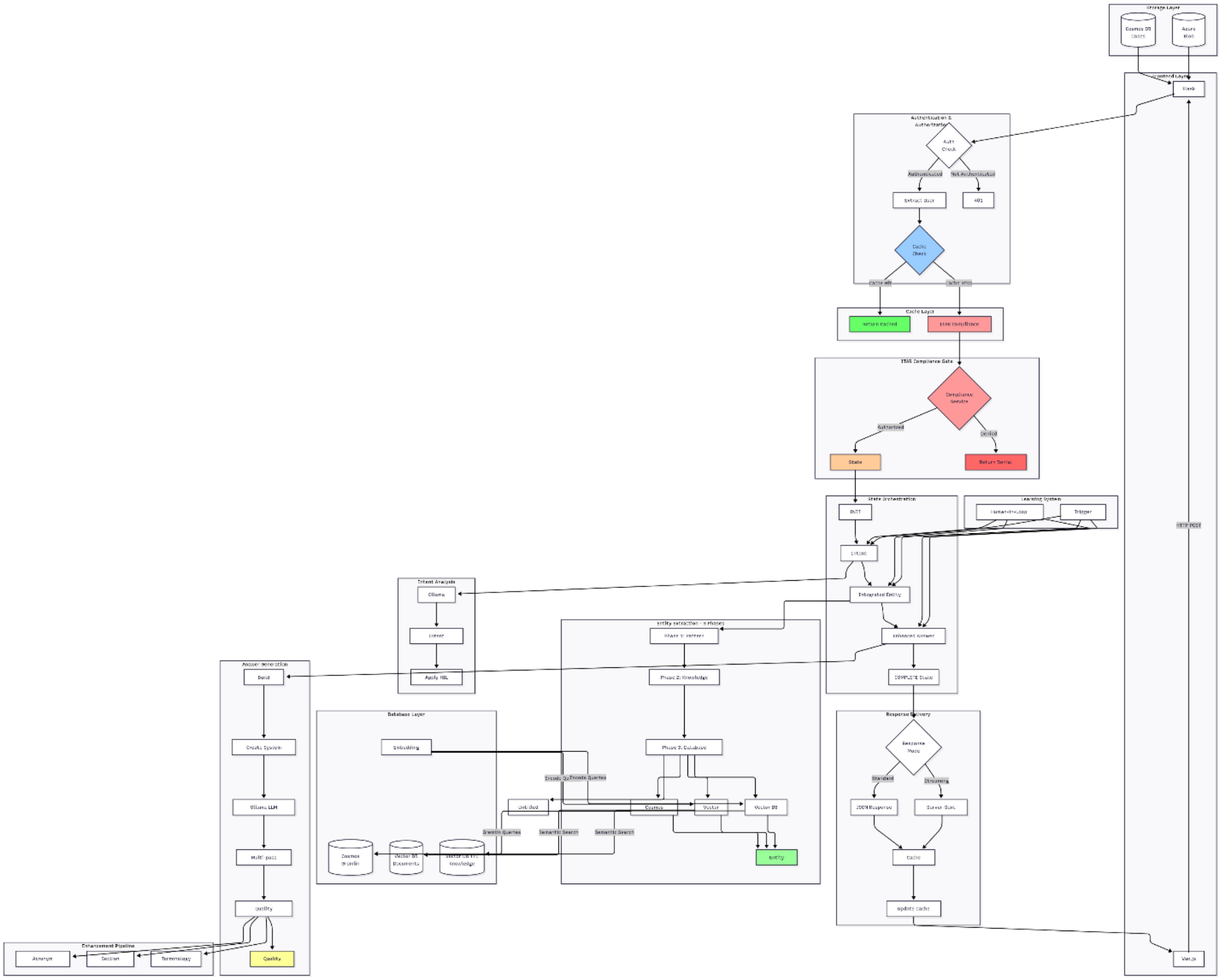
- Defines workflow phases: INIT, INTENT, ENTITY, ANSWER, COMPLETE, ERROR

SimpleKnowledgeGraph

- Parses TTL/RDF knowledge data
- Methods: find_entity(), get_relationships()

Key Data Flows

1. **Query Path**: User → Auth → Cache → ITAR → Orchestrator → Agents → LLM → Cache → User
2. **Entity Enrichment**: Query → Pattern Match → KG → Databases → Enriched Context
3. **Learning Path**: User Feedback → HIL Update → Agent State → Improved Future Responses
4. **Streaming Path**: Query → Orchestrator → SSE Stream → Token-by-token → Frontend


Storage Layer
Cosmos DB
cases
Azure
Blob
Frontend Layer
Flask
HTTP POST
Vue.js
Authentication & Authorization
Auth
Check
Authenticated
Not Authenticated
Extract User
401
Cache
Check
Cache Hit
Cache Miss
Cache Layer
Return Cached
ITAR Compliance
ITAR Compliance Gate
Compliance
Service
Authorized
Denied
State
Return Denial
State Orchestration
INIT
Intent
Integrated Entity
Enhanced Answer
COMPLETE State
Learning System
Human-in-Loop
Trigger
Intent Analysis
Ollama
Intent
Apply HIL
Entity Extraction - 3 Phases
Phase 1: Pattern
Phase 2: Knowledge
Phase 3: Database
Cosmos
Vector
Vector DB
Entity
Encode Queries
Gremlin Queries
Semantic Search
Semantic Search
Database Layer
Embedding
Untitled
Cosmos
Gremlin
Vector DB
Documents
Response Delivery
Response
Mode
Standard
Streaming
JSON Response
Server-Sent
Cache
Update Cache
Answer Generation
Build
Create System
Ollama LLM
Multi-pass
Quality
Quality
Enhancement Pipeline
Acronym
Section
Terminology