Multi-Agent E-Commerce Shopping Assistant: Technical Design Document

Executive Summary

This document outlines the technical architecture and implementation strategy for a production-ready conversational e-commerce assistant. The system employs a **context-first multi-agent orchestration pattern** with specialized domain agents, hybrid embedding strategies, and universal conversation management to deliver seamless shopping experiences across text and image modalities.

Key Innovation: Universal Context Pipeline ensuring every conversation turn maintains and enriches conversational state, enabling natural pronoun resolution and contextual understanding across all agent types.

---

1. Data Engineering & Schema Design

1.1 Scalable Database Architecture

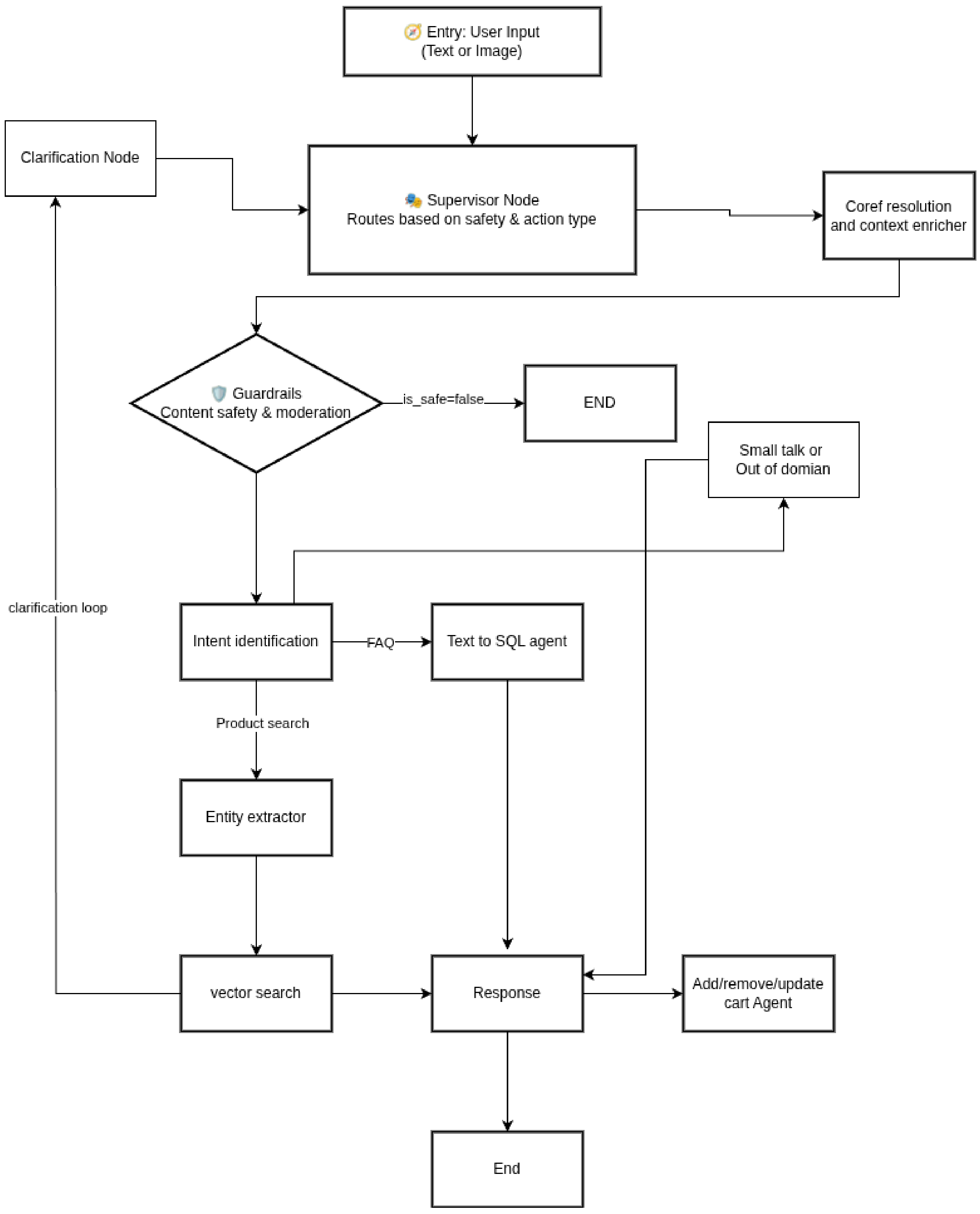
The foundation leverages a hierarchical product taxonomy designed for extensibility across diverse product categories:

Categories → Subcategories → Product_Types → Products → Product_Attributes

Design Rationale: This normalized schema pattern ensures:

- **Scalability**: Easy addition of new product verticals beyond t-shirts and TVs
- **Consistency**: Standardized attribute management across categories
- **Performance**: Optimized for both transactional operations and analytical queries
- **Flexibility**: Category-specific attribute tables (tshirt_attributes, tv_attributes) allow domain-specific properties without schema bloat

Validation Dataset: Initial implementation with diverse products (t-shirts and TVs) provides sufficient data diversity for proof-of-concept while maintaining manageable complexity for iterative development.


🧭 Entry: User Input (Text or Image)
Clarification Node
🎭 Supervisor Node Routes based on safety & action type
Coref resolution and context enricher
🛡️ Guardrails Content safety & moderation
is_safe=false
END
Small talk or Out of domian
clarification loop
Intent identification
FAQ
Text to SQL agent
Product search
Entity extractor
vector search
Response
Add/remove/update cart Agent
End

1.2 Text2SQL Infrastructure

The SQL layer serves dual purposes:

- **Operational Queries**: Real-time inventory, pricing, and availability checks
- **Analytics Queries**: Business intelligence, trend analysis, and recommendation engine support
- **Context-Aware Queries**: Resolves conversational references in analytical queries

Key Enhancement: Text2SQL agent now processes context-enriched queries, enabling natural follow-up questions like "average price of them?" where "them" refers to previously discussed products.

---

2. Embedding Strategy & Multi-Modal Architecture

2.1 Design-Time Embedding Generation

Model Selection: LlamaIndex VDR-2B-Multi-V1 chosen for its multi-modal capabilities and production-ready performance characteristics.

Dual-Vector Strategy:

- **Content Embedding**: Direct image feature extraction capturing visual characteristics
- **Context Embedding**: GPT-4.1 generated metadata descriptions converted to embeddings

Technical Rationale: This dual approach captures both visual similarity and semantic understanding, enabling queries like "casual blue shirt" to match both visually similar items and semantically related products.

2.2 Runtime Hybrid Embedding Composition

Query-Time Fusion: 0.8 image embedding + 0.2 metadata embedding weighting optimizes for:

- **Visual Dominance**: Primary matching based on actual product appearance
- **Semantic Enhancement**: Contextual understanding from descriptive metadata
- **Flexibility**: Configurable weights allow optimization based on performance metrics

Performance Consideration: This approach avoids expensive real-time image processing while maintaining semantic richness through metadata augmentation.

2.3 Enhanced Result Processing

Quality Control:

- Similarity threshold filtering (>0.6) ensures high-relevance results
- Standardized metadata extraction for consistent cart operations
- Universal metadata structure across all search results

Clarification Strategy: When high-quality results exceed manageable quantities, the system triggers intelligent clarification rather than overwhelming users with options.

---

3. Agent Orchestration Architecture: Context-First Design

3.1 Universal Context Pipeline: Core Innovation

Architectural Principle: Every conversation turn flows through a universal context management pipeline, ensuring consistent state maintenance across all agent types.

```
graph LR
    A[User Input] --> B[Supervisor]
    B --> C[Intent Classifier]
    C --> D[Entity Extractor]
    D --> E[Context Stitcher]
    E --> F{Route by Intent}
    F -->|FAQ| G[SQL Agent]
    F -->|Product Search| H[Vector Search]
    F -->|Clarification| H
```

Key Benefits:

- **Universal State Management**: Every turn updates conversation context
- **Pronoun Resolution**: "them", "it", "those" resolved using conversation history
- **Context Continuity**: Seamless transitions between different agent types
- **Memory Persistence**: Rich conversational memory across multi-turn interactions

3.2 Supervisor Agent: Intelligent Traffic Controller

The Supervisor operates with simplified, safety-first routing logic:

Priority-Based Routing:

1. **Safety Validation**: Content moderation and toxicity detection
2. **Cart Operations**: E-commerce transaction management with result validation
3. **Social Interaction**: Small talk and relationship building
4. **Domain Validation**: Shopping vs. non-shopping classification
5. **Universal Pipeline**: All other queries routed to context processing

Key Change: Supervisor no longer routes directly to specialized agents (except terminals), instead channeling all queries through the universal context pipeline for consistent state management.

3.3 Specialized Agent Portfolio

3.3.1 Guardrails Agent: Safety & Content Moderation

Responsibility: Multi-layered content safety with graduated response mechanisms

- **Detection**: Toxicity, harassment, spam, manipulation attempts
- **Response Strategy**: Severity-based responses (warn → redirect → block)
- **Integration**: Routes safe content to context pipeline for processing

3.3.2 Small Talk Agent: Social Relationship Management

Responsibility: Human-like social interaction and engagement

- **Coverage**: Greetings, gratitude, casual conversation, rapport building
- **Transition Strategy**: Smooth pivoting from social interaction to shopping assistance
- **Terminal Behavior**: Completes interaction without context processing needs

3.3.3 Out-of-Domain Agent: Boundary Management

Responsibility: Graceful handling of non-shopping queries

- **Classification**: Weather, news, general knowledge, personal advice
- **Response Strategy**: Polite boundary-setting with redirection to shopping
- **Engagement Preservation**: Maintains positive user relationship despite scope limitations

3.3.4 Intent Classifier: Query Understanding Engine

Core Capability: Multi-class intent detection with confidence scoring

- **Classifications**: FAQ, product_search, clarification_response, modification, continuation
- **Context Awareness**: Leverages conversation history for better classification
- **Universal Entry**: All non-terminal queries pass through intent classification

3.3.5 Entity Extractor: Multimodal Information Extraction ⭐

Revolutionary Capability: GPT-4.1 Vision-powered multimodal entity extraction

Technical Implementation:

```
# Multimodal message structure
{
    "role": "user",
    "content": [
        {"type": "text", "text": entity_prompt},
        {
            "type": "image_url",
            "image_url": {"url": f"data:image/jpeg;base64,{image_base64}"}
        }
    ]
}
```

Capabilities:

- **Simultaneous Processing**: Text and image analysis in single API call
- **Visual Entity Extraction**: Product type, color, style, pattern from images
- **Text Entity Extraction**: Brand, size, price range, preferences from text
- **Fusion Logic**: Combines visual and textual entities intelligently
- **Base64 Encoding**: Secure image transmission to GPT-4.1

Example Flow:

```
Input: [Image of red Nike t-shirt] + "size medium"
Extracted: {
    "product_type": "t-shirt",
    "brand": "Nike",
    "color": "red",
    "size": "medium",
    "visual_tags": ["has logo", "athletic"]
}
```

3.3.6 Context Stitcher: Conversation Intelligence Engine ⭐

Core Innovation: Advanced coreference resolution and context enrichment

Technical Capabilities:

- **Reference Resolution**: Resolves pronouns using multi-turn conversation history
- **Context Enrichment**: Accumulates user preferences across conversation turns
- **Conflict Resolution**: Handles contradictory information intelligently
- **Intent-Aware Processing**: Different stitching strategies for different intent types

Advanced Example Flow:

```
Turn 1: "white Levi's t-shirt"
Context: {color: white, brand: Levi's, product_type: t-shirt}
```

```
Turn 2: "how many do you have?" (FAQ)
Resolved: "how many white Levi's t-shirts do you have?"
Context: Maintained + {query_type: inventory}

Turn 3: "average price of them?"
Resolved: "average price of white Levi's t-shirts?"
Context: Maintained + {query_type: pricing}

Turn 4: "add the cheapest one to cart"
Resolved: Uses previous search results + pricing context
Context: Updated with cart action
```

3.3.7 SQL Agent: Context-Aware Analytics Engine ⭐

Enhanced Capability: Analytics with conversational context resolution

Key Features:

- **Contextual Query Resolution**: Processes pronouns using conversation history
- **Memory Integration**: Updates conversation context even for SQL queries
- **Safety Validation**: Prevents SQL injection while enabling natural language queries
- **Business Intelligence**: Supports complex analytical queries with context

Technical Flow:

```
def resolve_contextual_references(user_input, memory):
    # Extract recent context from conversation history
    recent_context = extract_context_from_turns(memory.turn_history[-3:])

    # Resolve pronouns using context
    if "them" in user_input and "brand" in recent_context:
        resolved = user_input.replace("them", f"{recent_context['brand']}
{recent_context['product_type']}")

    return resolved
```

3.3.8 Vector Search Engine: Hybrid Similarity Search

Responsibility: Multi-modal product discovery with metadata standardization

- **Hybrid Embeddings**: Image + text fusion for comprehensive matching
- **Quality Filtering**: Similarity threshold management (>0.6)
- **Metadata Standardization**: Consistent product data structure for downstream agents

- **Context Integration**: Uses stitched entities for enhanced search precision

3.3.9 Cart Agent: Context-Aware Transaction Management ⭐

Enhanced Capability: Intelligent cart operations with result integration

Key Features:

- **Result Integration**: Seamlessly adds products from search results or conversation history
- **Item Resolution**: Resolves "first one", "second one", "the blue one" using context
- **Session Persistence**: Maintains cart state across conversation flows
- **Flexible Action Detection**: Recognizes various cart command patterns

Technical Enhancement:

```
# Enhanced item selection
if "1st" in user_input or "first" in user_input:
   item_index = 0
elif "2nd" in user_input or "second" in user_input:
   item_index = 1
# Uses standardized metadata from vector search results
```

3.3.10 Clarification Checker: Intelligent Query Refinement

Responsibility: Smart result management and user guidance

- **Threshold Management**: Balances result quantity with user experience
- **Clarification Limits**: Maximum 3 clarifying questions per session to prevent loops
- **Context-Aware Questions**: Generates relevant clarifications based on search context

3.3.11 Response Generator: Natural Language Interface

Responsibility: Human-like response generation with rich product information

- **Metadata Integration**: Uses standardized product metadata for detailed responses
- **Context Awareness**: References conversation history in responses
- **Business Logic**: Incorporates pricing, availability, and recommendation logic

3.4 Reset Functionality: Session Management

Implementation: Comprehensive conversation state cleanup

- **Memory Clearing**: Conversation history and context reset
- **Cart Management**: Shopping cart state restoration

- **User Control**: Explicit user-initiated session restart capability

---

4. Tool Ecosystem & Integration Layer

4.1 Embedding Service Tools

- **Text Embedding API**: Consistent embedding generation for queries and metadata
- **Image Embedding API**: Visual feature extraction for product images
- **Hybrid Composition Engine**: Runtime embedding fusion with configurable weights

4.2 Database & Search Tools

- **Vector Database Interface**: High-performance similarity search with quality filtering
- **SQL Query Engine**: Structured data access for analytics and inventory
- **Text2SQL Converter**: Natural language to SQL with context resolution and safety validation

4.3 AI & Language Processing Tools

- **GPT-4.1 Vision Integration**: Advanced multimodal understanding and generation
- **Intent Classification**: Multi-class intent detection with confidence scoring
- **Entity Extraction**: Multimodal product attribute identification and normalization
- **Safety Classification**: Content moderation and appropriateness filtering
- **Context Resolution**: Pronoun and reference resolution using conversation history

4.4 Memory & State Management Tools

- **Universal Context Pipeline**: Multi-turn context preservation and enrichment
- **Session State**: Agent coordination and workflow management
- **Conversation Memory**: Rich turn-by-turn interaction history
- **User Profile**: Preference learning and personalization data

---

5. Enhanced Conversation Flow Examples

5.1 Context-Aware Multi-Modal Search

User: [Uploads image of blue denim jacket]
🖼️ Entity Extractor: {product_type: "jacket", color: "blue", material: "denim"}
🧵 Context Stitcher: Builds initial context
🔍 Vector Search: Hybrid embedding (0.8 image + 0.2 metadata)

🤖 System: "I found denim jackets similar to your image. Preferred price range?"

User: "under 3000"
🔍 Entity Extractor: {price_range: "under 3000"}
🧵 Context Stitcher: {product_type: "jacket", color: "blue", material: "denim", price_range: "under 3000"}
🔍 Vector Search: Filters results by price + similarity
🤖 System: "Here are 5 blue denim jackets under ₹3000..."

User: "add the second one"
🛒 Cart Agent: Resolves "second one" using search results + context
🤖 System: "✅ Added Levi's Denim Jacket (Blue) - ₹2,499 to cart!"

5.2 Context-Aware FAQ with SQL Analytics

User: "How many Wrangler t-shirts do you have?"
🎯 Intent Classifier: FAQ
🔍 Entity Extractor: {brand: "Wrangler", product_type: "t-shirt"}
🧵 Context Stitcher: Builds context
🗃️ SQL Agent: "How many Wrangler t-shirts do you have?" → "2 products"

User: "average price of them?"
🎯 Intent Classifier: FAQ
🔍 Entity Extractor: {pronoun: "them"}
🧵 Context Stitcher: "them" + context → "Wrangler t-shirts"
🗃️ SQL Agent: "average price of Wrangler t-shirts?" → "₹928.35"

5.3 Complex Multi-Turn Shopping Journey

User: [Image of pink t-shirt] "something similar but different brand"
🖼️ Entity Extractor: {product_type: "t-shirt", color: "pink", brand_preference: "different"}
🔍 Vector Search: Finds similar pink t-shirts, excludes detected brand
🤖 System: "Found pink t-shirts from other brands. Size preference?"

User: "medium"
🔍 Entity Extractor: {size: "medium"}
🧵 Context Stitcher: {product_type: "t-shirt", color: "pink", size: "medium", brand_preference: "different"}
🔍 Vector Search: Refined results
🤖 System: "Here are medium pink t-shirts from different brands..."

User: "how many Nike ones?"
🎯 Intent Classifier: FAQ

🔍 Entity Extractor: {brand: "Nike"}
🧵 Context Stitcher: Combines with existing context
🗃️ SQL Agent: "how many Nike medium pink t-shirts?" → "3 products"

User: "add the cheapest Nike one"
🛒 Cart Agent: Resolves using SQL results + search context + pricing
🤖 System: "✅ Added Nike Pink T-shirt (Medium) - ₹396.75 to cart!"

---

6. Technical Performance Considerations

6.1 Latency Optimization

- **Caching Strategy**: Pre-computed embeddings and frequent query results
- **Async Processing**: Non-blocking agent coordination
- **Pipeline Optimization**: Efficient context processing with minimal overhead
- **Circuit Breakers**: Fallback mechanisms for service dependencies

6.2 Quality Assurance

- **Universal Context Management**: Consistent state across all interaction types
- **Threshold Management**: Dynamic similarity thresholds based on query complexity
- **Clarification Limits**: Maximum 3 clarifying questions per session to prevent loops
- **Confidence Scoring**: Multi-dimensional confidence assessment across agent decisions

6.3 Monitoring & Observability

- **Context Pipeline Metrics**: Context resolution accuracy, entity extraction quality
- **Conversation Analytics**: Intent classification accuracy, clarification success rates
- **Business Metrics**: Search-to-cart conversion, session engagement depth
- **Technical Metrics**: Response latency, agent routing accuracy, error rates

---

7. Production Readiness Assessment

7.1 Current Implementation Status

✅ **Universal Context Pipeline**: Complete implementation across all agent types
✅ **Multimodal Entity Extraction**: GPT-4.1 Vision integration functional
✅ **Context-Aware SQL Agent**: Pronoun resolution in analytical queries
✅ **Enhanced Cart Management**: Integration with search results and context

✅ **Intelligent Clarification**: Smart threshold management
✅ **Safety and Domain Boundaries**: Comprehensive content moderation

7.2 Architectural Innovations Delivered

- **Context-First Design**: Revolutionary approach to conversational AI state management
- **Universal Pipeline**: Consistent processing regardless of query type
- **Multimodal Intelligence**: Seamless text and image understanding
- **Contextual Analytics**: Natural language SQL queries with pronoun resolution
- **Intelligent Cart Operations**: Context-aware e-commerce transactions

7.3 Scalability Roadmap

Next Phase: Retrieve & Rerank Strategy

- **Stage 1**: High-recall retrieval (100-500 candidates)
- **Stage 2**: Multi-signal reranking (semantic + business + personalization)
- **Stage 3**: Presentation optimization (diversity + business rules)

7.4 Business Impact Potential

This architecture delivers:

- **Enhanced User Engagement**: Natural conversation flows with perfect memory
- **Improved Conversion**: Intelligent clarification and seamless cart management
- **Scalable Growth**: Extensible schema and universal agent framework
- **Operational Efficiency**: Automated customer service with intelligent context routing

Competitive Advantage: The universal context pipeline represents a significant advancement in conversational AI architecture, enabling natural multi-turn interactions that competitors using traditional chatbot frameworks cannot match.

---

8. Conclusion

The implemented multi-agent e-commerce assistant demonstrates production viability with several architectural innovations:

1. **Universal Context Pipeline**: Every conversation turn enriches conversational state
2. **Multimodal Intelligence**: GPT-4.1 Vision enables true image understanding
3. **Context-Aware Analytics**: SQL queries understand conversational references
4. **Intelligent Cart Management**: Seamless integration between search and transactions

This context-first approach fundamentally transforms conversational commerce, enabling natural, memory-rich interactions that scale across complex multi-turn shopping journeys. The system balances immediate functionality with long-term architectural flexibility, positioning the platform for sustained competitive advantage in conversational e-commerce.