

# ShopAssist AI 2.0

System Design & Implementation Report

**Submitted by:** Deven Kalra, ML 74

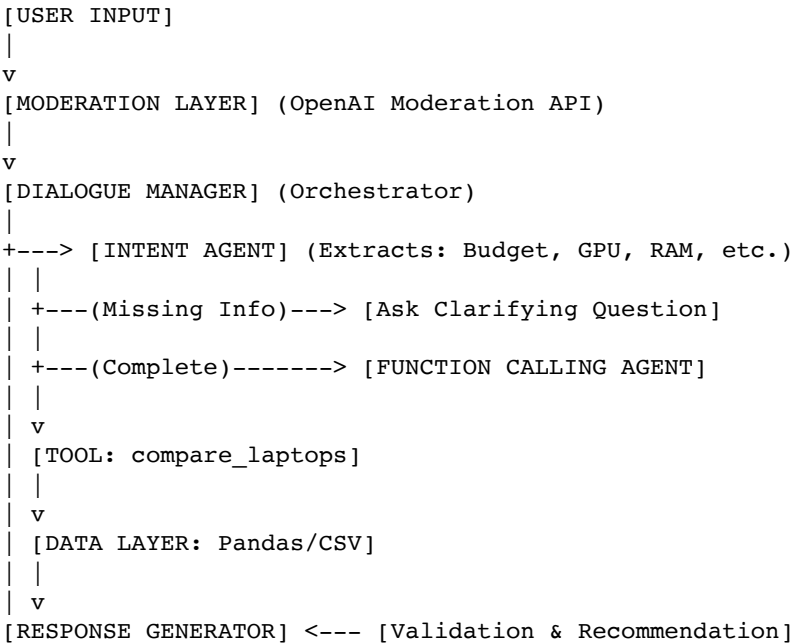
# Executive Summary

ShopAssist AI 2.0 is a specialized conversational commerce agent designed to assist users in selecting laptops based on their specific needs. Unlike standard chatbots, ShopAssist 2.0 employs a deterministic **Function Calling** architecture that retrieves real-time product data from a CSV database. This hybrid approach combines the natural language understanding of GPT-4 with the factual accuracy of a structured search engine, eliminating hallucinations regarding product specifications and prices.

## 1. System Architecture

The system utilizes a **Modular Agentic Architecture** orchestrated by a central Dialogue Management System. The conversation flow is linear but guarded by multiple validation layers to ensure safety and data integrity.

### 1.1 High-Level Data Flow



## 2. Module Documentation

### Dialogue Management System:

The central loop (dialogue\_mgmt\_system) that captures user input and routes it through the various layers. It maintains the conversation history and manages the exit conditions.

### Moderation Layer:

Implemented in `moderation_check`. Every user input is passed to the OpenAI Moderation API to detect harassment, violence, or self-harm before processing.

### **Intent & Slot Filling Layer:**

Implemented in `intent_confirmation_layer`. This agent acts as a gatekeeper. It analyzes the conversation to ensure 6 key 'slots' are filled: GPU Intensity, Display Quality, Portability, Multitasking, Processing Speed, and Budget.

### **Function Execution Engine:**

Implemented in `get_chat_completions_using_function_calling`. This specialized agent constructs a structured JSON payload to call the Python function `compare_laptops_with_user`.

### **Data Layer (Product Map):**

The `product_map_layer` preprocesses the raw laptop descriptions from the CSV, converting unstructured text into structured 'High/Medium/Low' tags to allow for mathematical comparison.

## **3. Technical Implementation**

### **3.1 Function Calling & JSON Handling**

The core innovation of ShopAssist 2.0 is the integration of OpenAI's Function Calling API. The system defines a strict schema for the `compare_laptops_with_user` tool. Crucially, the implementation handles the variability of LLM outputs by using robust parsing logic (`json.loads` and `ast.literal_eval`) to convert string responses into executable Python dictionaries.

### **3.2 Reliability (Tenacity)**

To mitigate API instability, the system wraps API calls with the `tenacity` library. The `@retry` decorator implements exponential backoff strategies, ensuring the system gracefully handles timeouts or rate limits without crashing.

## **4. Evaluation & Challenges**

The notebook includes a custom evaluation metric `evaluate_model_response` which compares the LLM's extracted profile against a ground-truth tag set.

- **Case Sensitivity:** The model often output 'High' while the logic expected 'high'. This was resolved by implementing string normalization (`.lower()`) in the evaluation layer.
- **JSON Parsing:** The LLM occasionally wrapped JSON in markdown blocks. The implementation uses a helper function `dictionary_present` to clean and extract the raw JSON string.

## **5. Conclusion**

ShopAssist AI 2.0 successfully demonstrates a robust implementation of Retrieval Augmented Generation (RAG) concepts using Function Calling. By grounding the LLM with a real-time CSV database and enforcing strict intent validation, the system provides accurate, safe, and context-aware product recommendations.