SHOPASSIST AI

August 6, 2025

Raja Ravi Sekar

Table of Contents

OBJECTIVE

The primary goal of **ShopAssist AI 2.0** is to **enhance online shopping experiences** by helping users find the best laptops based on their preferences.

SHOPASSIST AI 1.0

The ShopAssist AI 1.0 is the base for the ShopAssist AI 2.0.

On a high-level, the 1.0 captures the following

1. What is in ShopAssist AI 1.0

- Introduction
- System Design
- Implementation
- Evaluation

2. Implementation

The implementation is carried in 3 stages

- Intent clarity and confirmation
- Product mapping and information extraction
- Product recommendation

SHOPASSIST AI 2.0

1. Core Components

LLM Layer

- The LLM layer handles natural language understanding
- It determines user intent
- And it decides which function to call using Function Calling API

Function Layer

Functions exposed to the LLM

- extract_user_intent(user_input) → It returns structured profile dictionary
- map_laptop_features(description) → It classifies laptop specs
- compare_laptops(user_profile) → It returns top 3 laptops in JSON
- validate_recommendations(laptop_list) → It validate the recommendation quality

Data Layer laptops.csv has the Stores product details like Product Name, Price, Description, and Specs

Conversation & Dialogue Flow

- The Multi-turn conversation manager
- The Moderation & safety layer integrated
- A LLM orchestrates function calls and merges structured data into natural responses

2. The Function Calling flow

Flow of a typical conversation in ShopAssist AI 2.0

User → LLM (GPT-4 Function Calling) → extract_user_intent() → compare_laptops() → Top 3 laptops in INR → Conversational Response

User Input - I'm Raja, looking for a gaming laptop under Rs 125000 with high GPU performance and 16gb ram

Example

```
{
 "GPU Intensity": "High",
 "Display Quality": "High",
 "Portability": "Medium",
 "Multitasking": "High",
 "Processing Speed": "High",
 "Budget": 125000
}
```

LLM receives input

- The Uses Function Calling API to invoke extract_user_intent()
- It returns structured user profile dictionary

LLM calls compare_laptops()

- It passes the structured profile
- The backend filters and scores laptops
- It returns Top 3 laptops in JSON

LLM formats the output

- It converts JSON into a friendly chat response with key specs and follow-up question(s)

An optional Function Calls

- The validate_recommendations() to ensure the quality
- The map_laptop_features() for unseen product classifications

Implementation

Tools and Libraries

- Python (pandas, json, ast, openai)
- OpenAI API (GPT-4 with function-calling)
- CSV Dataset for laptops with pricing

3. Key Enhancements with ShopAssist AI 2.0 (Function Calling)

- The Intent Extraction - Structured via Function Calling
- The Product Matching - Automatic LLM-to-function routing
- A Recommendation output - Conversational, dynamic
- It's a Misunderstanding Handling - Auto-confirmation via structured dict
- An Extensibility - Easy to add new functions (e.g., vector search)

Benefits of This System Design

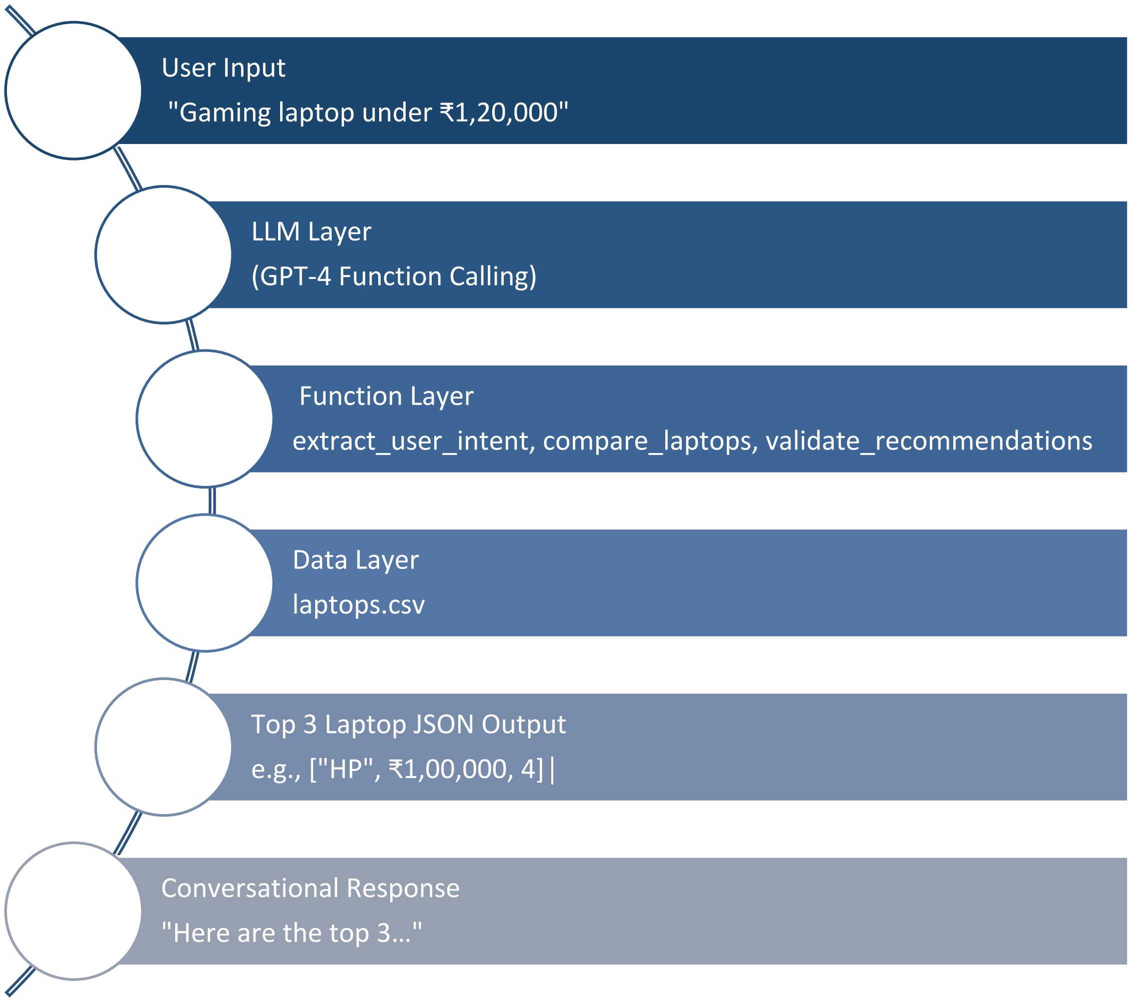
- The Structured Outputs → help in less parsing errors
- A Dynamic & Modular → Will help in easy to add more product domains
- The Function Calling → The LLM can trigger exact actions
- An Improved UX → Its reliable, human-like conversation
- The Future-proof → Its ready for embeddings & retrieval-augmented search

Define Functions for ShopAssist AI 2.0 We will expose 3 main functions to the LLM as below

- extract_user_intent → The extracts structured requirements
- compare_laptops → The filters and scores laptops to find top 3 matches
- validate_recommendations → It ensures recommendation quality

Define Function Schema for OpenAI

- We now register these Python functions with OpenAI using Function Calling API.



4. The Next Step - User experience

Full Function-Calling Loop will accept user input and send it to OpenAI LLM with Function Calling API enabled. If LLM decides to call a function

- It parse the function name & arguments
- Execute the corresponding Python function locally
- It send the function's return value back to the LLM
- The LLM formats a conversational recommendation
- Enable local function execution when LLM requests it (parse message["function_call"])
- Build the multi-turn dialogue loop to fully automate recommendations

Enhancements for Better UX

- Add moderation_check for user safety
- Integrate validate_recommendations to the filter weak matches
- Enable multi-turn refinement like "I also want an OLED display, can you re-check?" Calls compare_laptops again

Challenges Faced

- Inconsistent LLM Outputs in v1.0
  - It is solved with Function Calling API for structured JSON results
- Budget and Currency Handling
- Misclassification of Laptop Features
  - We have added rule-based scoring with stepwise evaluation
- Maintaining Multi-Turn Context
  - Implemented conversation memory using the dialogue manager

Lessons Learned

- Function Calling Enhances Reliability
  - Structured JSON prevents LLM misinterpretation
  - Easier to integrate with downstream logic
- Localization Matters
  - Displaying recommendations in INR (₹) improves usability for Indian users
- Hybrid Approach Works Best
  - Combining LLM reasoning with rule-based scoring gives accurate recommendations
- Modular Design is Scalable
  - Functions can be extended to other product categories like smart phones

-- The End --