Shop Assist 2.0 Function Call API Enhancement

Submitted By

Pradeep Harry Michael

Contents

Problem Statement

- 🧠 **Problem Statement: Enhancing ShopAssist AI with Function Calling for Improved Efficiency and User Experience**

The current version of ShopAssist AI lacks the architectural flexibility and conversational fluidity required to meet modern chatbot standards. With the introduction of the Function Calling API, there is an opportunity to significantly upgrade the system by streamlining its architecture and improving its interaction capabilities. The challenge lies in transforming ShopAssist into **ShopAssist 2.0**, a more efficient and intuitive chatbot that leverages function calling to deliver dynamic, context-aware responses.

Objective: Upgrade ShopAssist AI using the Function Calling API

Goals:

- Improve chatbot efficiency
- Enhance user experience
- Simplify architecture
- Deliver structured, actionable insights

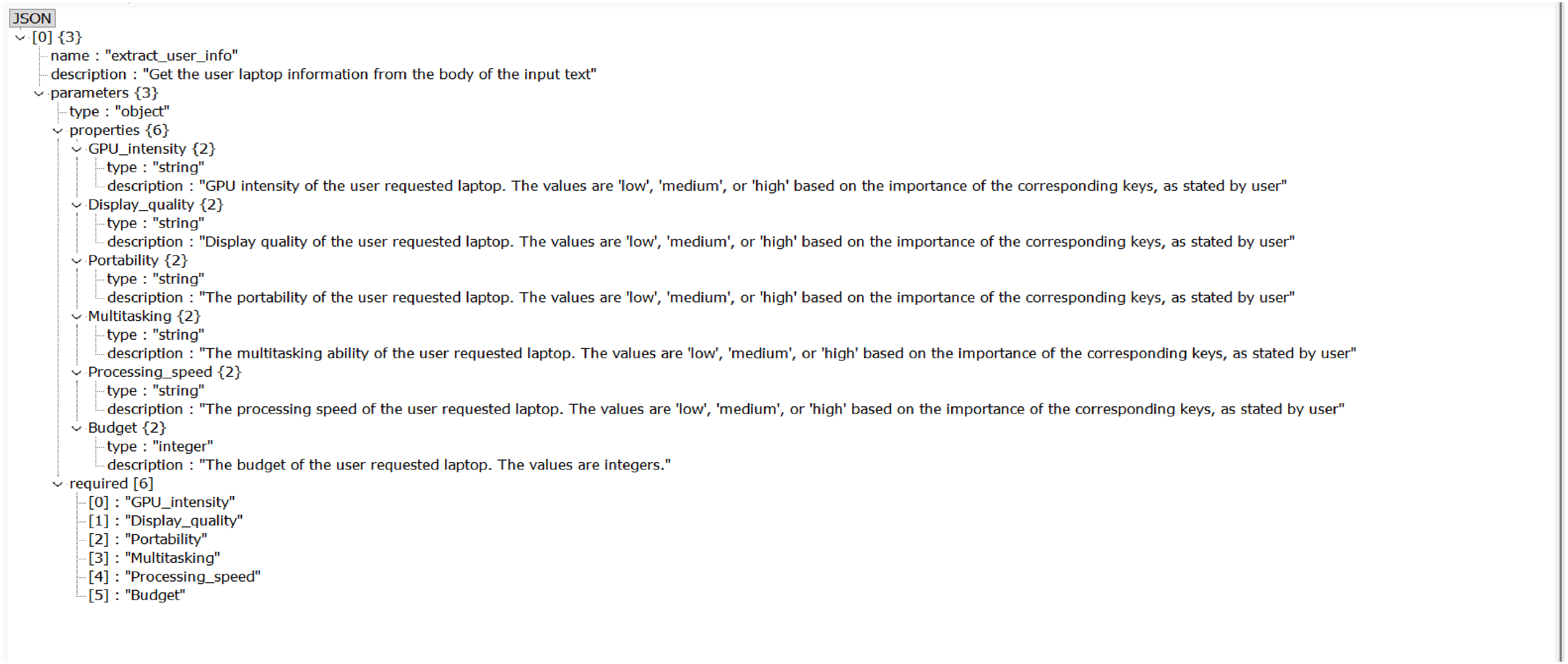
(38577, 54)

Enhancements : Function Description

Function Name	Purpose
extract_user_info	Extracts 6 key user preferences from conversation Parses user input for: GPU intensity Display quality Portability Multitasking Processing speed Budget
confirm_intent	Validates completeness and format of user inputs Ensures all 6 fields are present and valid Prevents incomplete or ambiguous recommendations
classify_laptop_features	Converts raw laptop descriptions into structured attributes (low/med/high) Translates product specs into standardized values Enables accurate matching with user preferences

(38577, 54)

Enhancements : extract_user_info Function Schema

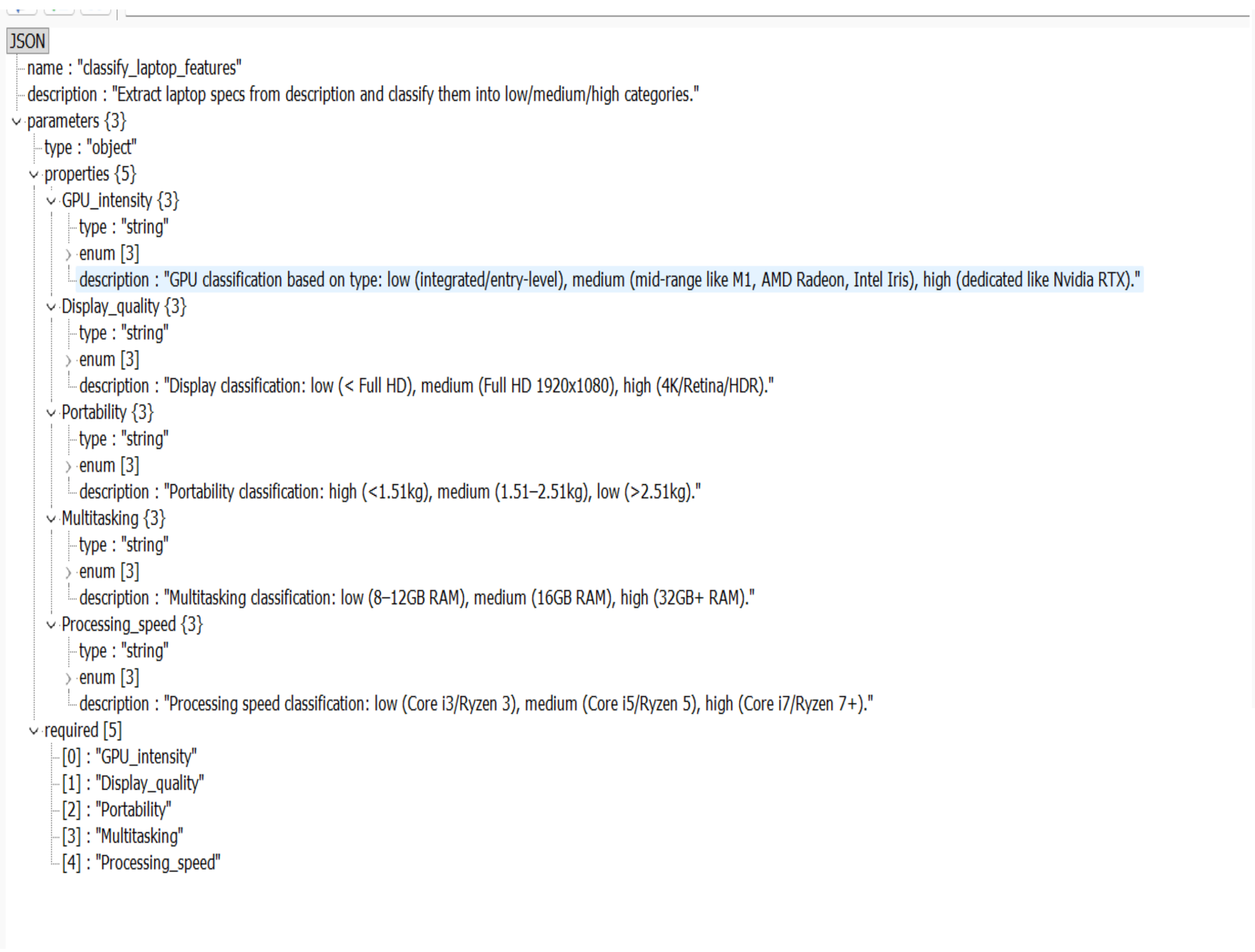


```
JSON
[0] {3}
  name : "extract_user_info"
  description : "Get the user laptop information from the body of the input text"
  parameters {3}
    type : "object"
    properties {6}
      GPU_intensity {2}
        type : "string"
        description : "GPU intensity of the user requested laptop. The values are 'low', 'medium', or 'high' based on the importance of the corresponding keys, as stated by user"
      Display_quality {2}
        type : "string"
        description : "Display quality of the user requested laptop. The values are 'low', 'medium', or 'high' based on the importance of the corresponding keys, as stated by user"
      Portability {2}
        type : "string"
        description : "The portability of the user requested laptop. The values are 'low', 'medium', or 'high' based on the importance of the corresponding keys, as stated by user"
      Multitasking {2}
        type : "string"
        description : "The multitasking ability of the user requested laptop. The values are 'low', 'medium', or 'high' based on the importance of the corresponding keys, as stated by user"
      Processing_speed {2}
        type : "string"
        description : "The processing speed of the user requested laptop. The values are 'low', 'medium', or 'high' based on the importance of the corresponding keys, as stated by user"
      Budget {2}
        type : "integer"
        description : "The budget of the user requested laptop. The values are integers."
    required [6]
      [0] : "GPU_intensity"
      [1] : "Display_quality"
      [2] : "Portability"
      [3] : "Multitasking"
      [4] : "Processing_speed"
      [5] : "Budget"
```

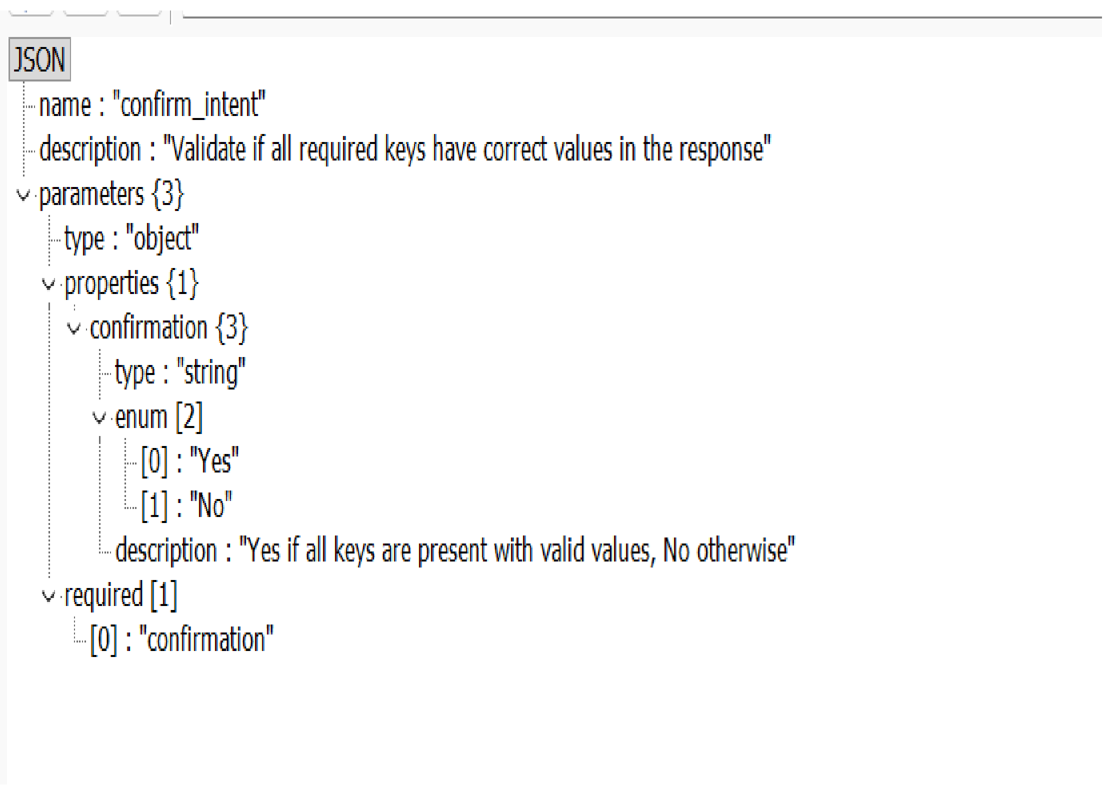
(38577, 54)

Enhancements : confirm_intent and classify_laptop_features

Function Schema



```
JSON
name : "classify_laptop_features"
description : "Extract laptop specs from description and classify them into low/medium/high categories."
parameters {3}
  type : "object"
  properties {5}
    GPU_intensity {3}
      type : "string"
      enum [3]
      description : "GPU classification based on type: low (integrated/entry-level), medium (mid-range like M1, AMD Radeon, Intel Iris), high (dedicated like Nvidia RTX)."
    Display_quality {3}
      type : "string"
      enum [3]
      description : "Display classification: low (< Full HD), medium (Full HD 1920x1080), high (4K/Retina/HDR)."
    Portability {3}
      type : "string"
      enum [3]
      description : "Portability classification: high (<1.51kg), medium (1.51–2.51kg), low (>2.51kg)."
    Multitasking {3}
      type : "string"
      enum [3]
      description : "Multitasking classification: low (8–12GB RAM), medium (16GB RAM), high (32GB+ RAM)."
    Processing_speed {3}
      type : "string"
      enum [3]
      description : "Processing speed classification: low (Core i3/Ryzen 3), medium (Core i5/Ryzen 5), high (Core i7/Ryzen 7+)."
  required [5]
    [0] : "GPU_intensity"
    [1] : "Display_quality"
    [2] : "Portability"
    [3] : "Multitasking"
    [4] : "Processing_speed"
```



```
JSON
name : "confirm_intent"
description : "Validate if all required keys have correct values in the response"
parameters {3}
  type : "object"
  properties {1}
    confirmation {3}
      type : "string"
      enum [2]
        [0] : "Yes"
        [1] : "No"
      description : "Yes if all keys are present with valid values, No otherwise"
  required [1]
    [0] : "confirmation"
```

(38577, 54)

Enhancements : Updated Conversation Flow

Step 1: User shares needs

Step 2: extract_user_info captures preferences

Step 3: confirm_intent validates input

Step 4: Laptop descriptions processed via classify_laptop_features

Step 5: Chatbot recommends best-fit laptops

(38577, 54)

Enhancements : Updated Method with Function Call Integration

Function	Purpose	Function API Used
initialize_conversation()	Sets up expert assistant prompt & starts structured dialogue	
get_chat_model_completions(messages)	Gets assistant's next message via Chat API	
moderation_check(user_input)	Ensures input/output safety via moderation	
get_user_requirement_string(response)	Converts response to readable preference string	
get_chat_completions_func_calling(input)	Extracts structured preferences using function calling	extract_user_info
intent_confirmation_layer(response)	Validates 6 user preferences (5 keys + budget ≥ 25000)	confirm_intent
compare_laptops_with_user(profile)	Scores laptops from CSV based on user match	
recommendation_validation(recommendation)	Filters laptops with score > 2	
initialize_conv_reco(products)	Starts product-focused conversation	
product_map_layer(description)	Classifies laptop specs into 5 categories	classify_laptop_features
dialogue_mgmt_system()	Orchestrates moderation, flow, matching, and dialogue	

(38577, 54)

Summary

⚙️ **Why Function Calling API Is Essential for Smarter Chatbots**

🔍 **Problem with Traditional Chatbots**

- Rely on rigid, rule-based parsing
- Struggle with extracting structured data from natural language
- Require manual validation and formatting
- Limited scalability and adaptability

Benefit of Function Calling API	**Impact**
🧠 **Structured Data Extraction**	Automatically converts user input into clean JSON format
✅ **Intent Validation**	Ensures all required fields are present and correctly formatted
🔄 **Dynamic Flow Control**	Enables modular, context-aware conversation management
📦 **Simplified Architecture**	Reduces need for custom parsing layers and manual logic
📈 **Scalability & Maintainability**	Easier to extend, debug, and maintain over time

💡 **Result:** Function Calling transforms ShopAssist into a **more intelligent, responsive, and efficient assistant**, capable of delivering personalized recommendations with precision.