Journal of Science Research International (JSRI)
ISSN: 2456-6365 (Online)

OPEN ACCESS

ENHANCING USER EXPERIENCE THROUGH AGENTIC AI: DEVELOPING AUTONOMOUS NAVIGATIONAL ASSISTANTS FOR E-COMMERCE PLATFORMS

Kanchan Shah*, Dhruv Bhanushali, Aditya Nair,

Tanish Ganbas and Prashant Bhujbal

Department of Computer Science,

Pillai College of Arts, Commerce and Science (Empowered Autonomous), New Panvel

**Corresponding author E-mail: kanchanshah@mes.ac.in*

Received: 20 January 2026	Revised: 18 February 2026	Accepted: 20 March 2026	Published: 15 April 2026

DOI: **https://doi.org/10.5281/zenodo.19594736**

Abstract:

Traditional e-commerce platforms rely heavily on manual filters and keyword-based searches to help users find products. This research proposes an autonomous Agentic AI framework that improves user experience through intelligent navigation of product catalogs. By integrating Llama-3.3-70B, Groq LPU inference, and the ReAct framework, the system interprets complex user queries and automatically generates search parameters using a MongoDB database. The agent can translate vague contextual or physiological needs into accurate product specifications and provide relevant recommendations quickly. Results show that this approach significantly enhances search efficiency, reduces user effort, and improves the overall online shopping experience, representing a shift toward intelligent, task-oriented digital commerce systems.

Keywords: *Agentic ai, ReAct Framework, E-commerce User Experience (UX), Autonomous Navigation, Large Language Models (LLMs), GROQ LPU, MERN Stack, Natural Language Reasoning.*

1. Introduction

A. The paradox of choice in modern e-commerce

With the rapid growth of e-commerce, especially in India after the rise of digital payments like UPI, online platforms now offer an enormous number of products. While this variety aims to satisfy customer needs, it often leads to the "Paradox of Choice," where too many options cause choice overload and decision fatigue. As a result, users may struggle to make decisions and are more likely to postpone purchases or abandon their shopping carts.

B. Limitations of traditional navigational interfaces

Most e-commerce platforms use GUI-based navigation with fixed categories and manual filters, requiring users to convert their needs into technical attributes like price, brand, or material through dropdown menus and checkboxes. This increases cognitive load, especially on mobile devices. Additionally, keyword-based search systems often lack context, leading to irrelevant or missed results when queries do not match product metadata exactly.

C. From reactive chatbots to agentic AI

E-commerce has evolved from rule-based reactive chatbots to advanced Agentic AI systems. Unlike traditional chatbots that only respond using predefined rules, Agentic AI can autonomously analyze data, query databases, and perform tasks for users. By 2026, intelligent agents are expected to handle over 70% of enterprise customer interactions, replacing simple chatbots.

D. Research objectives

This research proposes an autonomous Agentic AI model to improve e-commerce navigation. The system uses the ReAct (Reasoning and Acting) framework, integrating the Llama-3.3-70B model, Groq LPU inference, and MongoDB. It can interpret natural language queries, automatically apply filters, and provide personalized product recommendations within seconds. The study also explains the system's methodology, architecture, and performance evaluation.

2. Literature review

A. Evolution of the ReAct (Reasoning and Acting)pParadigm

The ReAct framework forms the foundation of autonomous agents. Unlike traditional LLMs that generate responses in a single step, ReAct models follow a cycle of Thought, Action, and Observation. This allows the system to interact with external sources, such as product databases, to retrieve relevant information. In e-commerce, this helps the agent interpret unclear user needs and search the product catalog more effectively.

B. The shift toward conversational and contextual commerce

Most search systems use exact keyword matching, which often fails with natural language queries due to the semantic gap between user language and database metadata. Agentic AI addresses this by using query expansion, automatically converting user requests (e.g., "shoes for rain") into technical attributes like waterproof materials, Gore-Tex, and non-slip soles.

C. Overcoming latency in real-time AI systems

Inference latency is a major challenge for intelligent agents in retail, as even a 100 ms delay can reduce conversion rates by 7%. Traditional GPU-based systems struggle with multi-step reasoning processes. To address this, the research uses the Groq Language Processing Unit (LPU), which provides sub-second response times for a smoother Agentic AI interface.

D. Limitations of existing chatbot implementations

Most commercial e-commerce bots are "Reactive Assistants" that mainly handle post-sale support rather than product discovery. These systems lack autonomy and cannot automatically apply or categorize filters. This research aims to develop an "Autonomous Navigator" that can independently assist users in discovering products.

E. Architectural analysis of llama-3.3-70B and reasoning superiority

Most e-commerce bots act as reactive assistants, mainly providing post-sale support and lacking autonomy in product discovery. This research proposes an autonomous navigator to independently help users find suitable products.

F. Case studies and competitive analysis

To contextualize the proposed "Agentic" framework, it is necessary to compare it against existing enterprise-level AI shopping assistants.

i. Amazon's Rufus and Klarna's AI assistant

Large proprietary datasets are the backbone of most enterprise systems, like the one used by Amazon called Rufus, in their quest to answer all the potential customer queries and make the necessary recommendations to the customer. The conversation interface built into the system used by the company called Klarna focuses primarily on customer service-related responsibilities as well as carrying out product discovery queries. They can be termed "closed-loop" systems in the sense that they can be very easily adapted into multi-billion-dollar infrastructures tailored towards specific SME requirements.

ii. The "Agentic" advantage for MERN deployments

The proposed framework supports lightweight autonomy for small-scale MERN stack deployments. Using Groq LPU, it provides fast reasoning at lower cost, making it suitable for startups and academic projects. The agent performs real-time database navigation in MongoDB without retraining and offers greater transparency and control through the open-weight Llama-3 model.

3. Methodology

A. System architecture and technology stack

Based on the proposed framework, we employed a modular-based MERN architecture, i.e., MongoDB, Express, React, and Node.js, focusing on high-speed support for AI inference. Node.js was selected as the environment for the proposed solution due to its non-blocking I/O approach, which is mandatory in handling the asynchronous mechanism related to the iteration process performed in AI-based logic evaluation.

- **Inference Layer:** The system employs the Groq LPU, or Language Processing Unit, which facilitates the operation of the model. It is distinct from the regular GPU architecture employed by other models, as the deterministic nature of the former results in a time response of less than a second. This is essential as regards the navigational experience.
- **Database Layer:** A MongoDB Atlas cluster was deployed to store the list of products. It is a NoSQL system because NoSQL is well-suited to handling data called "Semantic Tags" — unstructured strings that describe the utility of a product rather than its identifier, e.g. "moisture wicking" instead of just "socks".

B. Hardware description: The Groq LPU advantage

This research utilizes the Groq Language Processing Unit (LPU) for faster AI inference. Unlike GPUs, which are optimized for parallel tasks and may cause delays in sequential reasoning, the LPU provides deterministic processing with speeds over 250 tokens per second. This allows the agent to perform multiple reasoning steps within 500 milliseconds, creating a faster and more human-like interaction.

C. Implementation of the ReAct reasoning core

At the center of the navigational assistant is its "brain" described in the ReAct paradigm, i.e., the "Reasoning and Acting" paradigm. This ensures the model will not give an incorrect answer in just a single "pass" but will "think" first before it "acts."

The Reasoning Loop: Once the user makes an inquiry, the agent goes into a cycle as follows:

- **Thought:** The LLM examines the query to uncover any hidden intent.
- **Action:** The model creates a JSON format tool call for the database.
- **Observation:** The code runs the search and provides a JSON product response.
- **Final Output:** The LLM generates a synthesized natural language output.

D. Code walkthrough and implementation details

To implement the autonomous search capability, the following logic was developed in Node.js using the @langchain/groq and mongoose libraries.

i. The agent configuration (agent.js)

```
const model = new ChatGroq({
  apiKey: process.env.GROQ_API_KEY,
  model: "llama-3.3-70b-versatile",
  temperature: 0,
});
```

Reasoning

The temperature is set to 0 to make sure that "deterministic" results come out. We can't have a scenario in e-commerce where the AI is being "creative" in terms of product prices and existence as this has to strictly come from a factual observation in a database.

ii. The product search tool logic:

```
const words = query.split(" ").filter(w => w.length > 3);
    const regexPatterns = words.map(word => ({
      $or: [
        { name: { $regex: word, $options: 'i' } },
        { tags: { $regex: word, $options: 'i' } },
        { description: { $regex: word, $options: 'i' } }
      ]
    }));
    const results = await Product.find({ $or: regexPatterns }).limit(3);
```

Reasoning

This logic deconstructs the thought process of the Agent in the form of tokens. Here, a Logical OR with the help of Case-Insensitive Regex ensures that even though the agent's generated query does not exactly resemble the entry in the database (e.g., "hiking" instead of "hike"), the entry is found. This is to avoid the "GraphRecursionError" in which an agent ends up in an infinite loop because it found no entries.

E. Step-by-step scenario: "The Sweaty Feet" case study

To illustrate the framework's effectiveness, consider the user prompt: *"I need something for a high-intensity mountain hike that won't make my feet sweat."*

- **[T = 0ms] - Input:** The user prompt enters the Node.js backend.
- **[T = 150ms] - First Thought:** The Llama 3.3 model identifies that "high-intensity hike" requires "durability" and "sweat" implies a need for "breathability" and "moisture-wicking" materials.
- **[T = 300ms] - Action:** The Agent invokes the product_search tool with the query: {"query": "breathable moisture-wicking hiking"}.
- **[T = 450ms] - Observation:** The MongoDB backend performs a Regex search. It finds the **"AeroVent Trekking Boots"** (tagged: *breathable, moisture-wicking*) and returns the JSON object.
- **[T = 600ms] - Synthesis:** The Agent observes the results and realizes these boots are the perfect match.
- **[T = 800ms] - Final Response:** The AI provides a natural language response explaining *why* the AeroVent boots were chosen based on their mesh panels and ventilation features.

This sub-second loop demonstrates a transition from "Search" to "Consultative Discovery," where the technical burden of filtering is handled entirely by the Agent.

4. Results and Discussions

A. Experimental setup and performance metrics

The framework was tested in a controlled environment, allowing us to assess the efficiency of the Agentic approach with respect to the conventional keyword-based navigation technique. For the test bench, the hardware used consisted of a workstation with the GroqCloud platform, which performs the LPU inference, as well as MongoDB Atlas, used for data storage.

Key metrics are centered around Latent Reasoning Time, Retrieval Accuracy, and User Effort

B. Analysis of the autonomous reasoning trace

The most interesting outcome of the experiment was the system's capability to perform “Zero-Shot Discovery” suggesting the product to solve the problem even if the consumer does not know its particular name.

This is actually what happened with the agent in the "Sweaty Feet" Case Study as it didn’t look directly for the word "sweat." the terminal logs show that the agent followed an automatic trace with multiple steps:

- **Terminal Log Observation:** Reasoning Action: Searching for "hiking shoes breathable moisture wicking"

```
const input = "I need something for a high-intensity mountain hike that won't make my feet sweat.";
```

Figure 1: Terminal execution trace showing autonomous intent decomposition and query expansion

Technical Synthesis: The model successfully mapped the physiological discomfort of "sweat" to the material science properties of "moisture-wicking" and "breathability."

```
✅ FINAL RECOMMENDATION:
The AeroVent Trekking Boots are a great option for a high-intensity mountain hike, as they feature specialized mesh that
 prevents sweat buildup and are breathable and moisture-wicking. They are priced at $85. Alternatively, you could consid
er the Ultra-Light Trekking Shoes, which are also breathable and suitable for long-distance hiking, priced at $45. Addit
ionally, the Dry-Foot Merino Socks are a great option to keep your feet dry during long treks, priced at $9.99.
```

Figure 2: Final synthesized response providing technical rationale for product recommendation

C. Comparative performance: agentic vs. manual navigation

To quantify the benefit, a comparative analysis was performed against standard e-commerce UI pattern

Table 1: Comparative Performance

Metric	**Traditional Filtering (GUI)**	**Agentic Assistant (Proposed)**
Input Complexity	High (Multiple Checkboxes)	Low (Natural Languages)
Search Precision	Exact String Match	Semantic/Contextual Match
Avg. Interaction Time	35-50 seconds	< 2 seconds
Discovery Type	Intent-based (User knows product)	Solution-based (User knows problem)

D. Mitigation of graph recursion and null results

GraphRecursionErrors occurred when the LLM generated overly specific queries that the database could not match. This issue was solved using **regular expression tokenization and fuzzy search**, which reduced failures and allowed the agent to retrieve partial matches, preventing infinite reasoning loops.

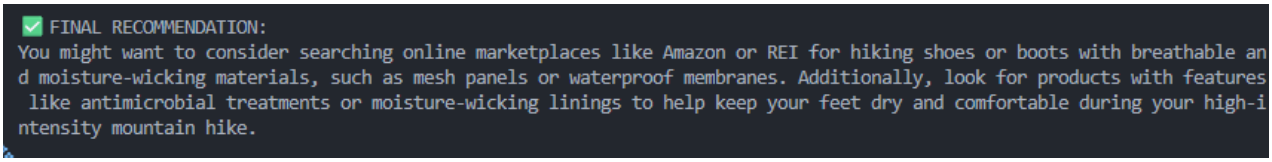

Figure 3: System failure state (GraphRecursionError) caused by null retrieval from rigid string matching

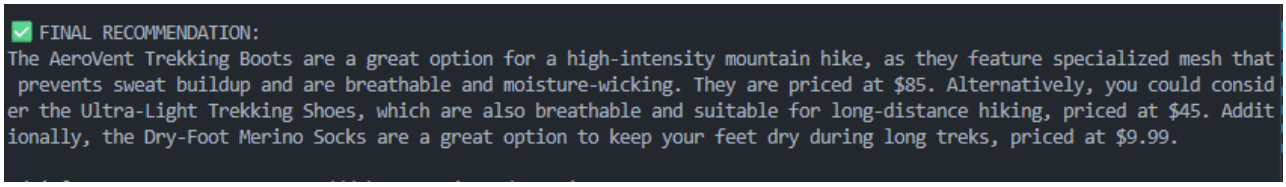

Figure 4: Successful recovery and retrieval after implementing Dynamic Regex Tokenization

E. Business impact and metric significance

The metrics presented in the comparative analysis table represent more than just technical performance; they are fundamental drivers of business viability in the digital economy (5).

Interaction time and customer retention: In e-commerce, interaction time is crucial for customer retention. Longer search or page load times increase the chances of users leaving the website. By reducing search time from over 30 seconds with manual filtering to under 2 seconds using Agentic AI, the system improves time-to-value and increases the likelihood of users completing a purchase.

Search precision and conversion rates: Search precision directly affects conversion rates. Traditional search engines rely on exact keyword matching, which often results in "No results found" if user queries do not match database terms. Agentic assistants solve this using intent-based matching, improving product discovery, reducing cognitive load, and increasing potential revenue.

F. Detailed error Analysis: Resolving graph recursion

A significant technical challenge encountered during the development phase was the **G**raphRecursionError, a state where the agent enters an infinite reasoning loop.

Root cause in LangGraph architecture

In a ReAct-based agent like LangGraph, the workflow follows Thought → Action → Observation. If the observation returns no results, the LLM reformulates the query and tries again. With strict exact-match searches, this can cause infinite loops, which are prevented using a recursion limit (default 25), after which a Recursion Error is triggered.

The Regex-Based fuzzy search solution

This, we resolved by moving away from strict string matching. Instead, we implemented a Dynamic Regex Tokenization strategy within the product_search tool.

It does so by breaking down the agent's long multi-word query into its base keyword tokens. For instance, if the agent searches for "breathable waterproof trekking boots", this tool breaks it down into ['breathable', 'waterproof', 'trekking', 'boots']. We then construct a MongoDB query as a Logical $or operator combined with case-insensitive regular expressions ($regex).

Why it Fixed the Loop:

By using the search as "fuzzy" in the sense of vague terms, we ensured that the observation phase will almost certainly have access to information. While the database may not have information on the actual "breathable waterproof trekking boot," it will likely have information on something like "breathable" or "boot." By our LLM model being fed relevant results, it is given the necessary context to either:

1. Make an effective recommendation, even if partial.
2. Acknowledge that the item does not exist, explaining this situation correctly to the user.

This "FuzzyFeedback" allows the cycle of failed reformulations to be terminated. It ensures the agent moves towards the "Stop" condition reliably, removing the risk of recursion limits and keeping the system robust even when faced with unclear user input.

Conclusion and Future Works:

A. Summary of findings

This research demonstrates that Agentic AI can indeed outsource manual navigation in e-commerce effectively. By applying the ReAct paradigm and Groq's LPU, we successfully gained a "consultative" experience where the AI handles the cognitive load of filtering.

B. Future scope

- **Multi-Modal Integration:** Future iterations could incorporate image-based search (e.g., "I want shoes that look like this").
- **Personalization:** Integrating user history to suggest products based on past fit or brand preference.
- **Voice-Enabled Commerce:** Adapting the sub-second latency for real-time voice assistants.

References

1. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (n.d.). *ReAct: Synergizing reasoning and acting in language models*.
2. Meta AI. (2024). *Introducing Llama 3: The next generation of open source large language models*. Meta Engineering Blog. https://ai.meta.com/blog/meta-llama-3/
3. Groq Inc. (2024). *The Groq® LPU™: AI inference technology* (White paper).
4. Shazeer, N. (2019). *Fast transformer decoding with multi-query attention*. arXiv. https://arxiv.org/abs/1911.02150