IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

Smart Deal Finder Bot Using Agentic AI: An Intelligent Conversational System for Automated E-Commerce Price Discovery and Comparison

Navnath Lahane[1], Aditya Patil[2], Madhav More[3], Prof. Anil Walke[4]
Department of Artificial Intelligence and Data Science[1-3]
ISBM College of Engineering, Pune, Maharashtra, India
navnath.lahane@isbmcoe.org, aditya.patil@isbmcoe.org, madhav.more@isbmcoe.org
anil.walke@isbmcoe.org

Abstract: *The rapid expansion of electronic commerce has created a highly fragmented marketplace in which consumers must navigate dozens of platforms to identify the most competitive prices for any given product. This manual comparison process is time-consuming, error-prone, and cognitively demanding. This paper presents the Smart Deal Finder Bot, an agentic artificial intelligence system designed to automate product price discovery, aggregation, and comparison across multiple e-commerce platforms through a conversational interface. Unlike conventional price-comparison websites that rely on static data snapshots, the proposed system employs an agentic AI architecture in which autonomous software agents continuously monitor, scrape, and evaluate live pricing data in real time. The bot integrates a natural language processing engine for intent recognition and named-entity extraction, a dynamic web-scraping and API module for data collection, a multi-criteria price comparison engine, and a personalised recommendation sub-system powered by a lightweight machine learning ranking model. Experimental evaluation conducted on a dataset of 500 product queries across five major Indian e-commerce platforms demonstrates that the system achieves an intent-classification accuracy of 93.6%, a named-entity recognition F1-score of 91.2%, and an average query response time of 2.4 seconds. The bot reduces the average consumer deal-finding time by approximately 78% compared to manual browsing. These results confirm that the proposed agentic AI paradigm offers a practical and scalable solution to the modern price-discovery problem.*

Keywords: Agentic AI; Natural Language Processing; Web Scraping; Price Comparison; E-Commerce Automation; Intent Recognition; Conversational Agents

I. INTRODUCTION

Electronic commerce has undergone transformational growth over the past decade. Industry analysts report that global e-commerce revenues surpassed USD 5.8 trillion in 2023 and are projected to exceed USD 8 trillion by 2027 [1]. In India alone, the digital retail sector is expected to reach USD 350 billion by 2030, driven by rising internet penetration, increasing smartphone adoption, and a growing middle-class consumer base [2].

Despite this growth, a fundamental inefficiency persists: the information asymmetry between buyers and sellers. A consumer wishing to purchase a laptop, a pair of headphones, or a home appliance must independently visit Amazon, Flipkart, Snapdeal, Croma, Reliance Digital, and several other platforms, compare specifications, evaluate reviews, and assess pricing before mak-ing a purchase decision. This fragmented experience imposes a significant cognitive and time burden on consumers.

Chatbot technology and conversational AI have demonstrated considerable promise in simplifying user interactions with digital services [3]. However, most existing shopping chatbots operate within a single platform's ecosystem, lack

DOI: 10.48175/IJARSCT-35387

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

true real-time price awareness, and cannot reason autonomously across external data sources. They respond to narrowly defined commands and fail to exhibit the goal-directed, adaptive behaviour required for comprehensive deal finding.

The emergence of agentic AI—a paradigm in which AI systems autonomously plan, execute multi-step tasks, and use exter-nal tools to accomplish user-defined objectives—offers a com-pelling solution to these limitations [4]. An agentic system does not merely respond to a query; it perceives the environ-ment, decomposes a goal into sub-tasks, selects appropriate tools, executes those tasks iteratively, and synthesises a coherent, actionable response.

This paper presents the Smart Deal Finder Bot, a fully re-alised agentic AI system that enables users to find the best avail-able price for any product through a natural-language conversa-tional interface. The key contributions of this work are:

1. A novel agentic AI architecture coordinating an NLP engine, a real-time web-scraping module, a multi-criteria price com-parison engine, and an ML-based ranking model within a unified conversational interface.
2. An original intent-recognition and entity-extraction pipeline fine-tuned for the Indian e-commerce domain.
3. Comprehensive experimental evaluation demonstrating accu-racy, latency, and user-satisfaction metrics superior to base-line approaches.
4. An extensible system design supporting future integration of voice interfaces, mobile clients, and advanced personalisation.

II. LITERATURE REVIEW

Price-comparison systems, conversational agents, and in-telligent scraping frameworks have each attracted substantial research attention. This section surveys the most relevant prior works and identifies the gaps that the proposed system addresses.

A. Price Comparison Systems

Narayanan et al. [5] proposed a rule-based price-comparison engine that aggregates data from five major e-commerce portals using structured API calls. Their system achieved high data ac-curacy but was limited to platforms with official APIs and could not handle dynamic JavaScript-rendered pages. Furthermore, the system had no conversational component, requiring users to interact through rigid web forms.

B. Conversational Commerce Agents

Patel and Sharma [6] designed a shopping assistant chatbot built on the Rasa open-source framework. The system effectively

III. PROBLEM STATEMENT

Let P = {p1, p2, . . . , pn} denote the set of all products available across a collection of e-commerce platforms S = {s1, s2, . . . , sm}. For each product pi on platform sj, let price(pi, sj), r(pi, sj) ∈ [0, 5], and d(pi, sj) denote its price, rating, and delivery time respectively.

A consumer issues a natural-language query q expressing a purchase intent. The problem is formally defined through two interlocking definitions.

Definition 1 (Query Understanding): Given an unstruc-tured natural-language string q, extract a structured intent tuple:

$I(q) = \langle \text{intent, product, brand, budget, platform} \rangle$ (1)

where each field may be null if not expressed in q.

Definition 2 (Optimal Deal): Given I(q) and real-time data D scraped from S, identify the listing $p^{\wedge}$ that maximises:

handled product inquiries on a single retail platform and demon-strated good intent recognition accuracy (88% on an in-domain test set). However, the agent was confined to one platform's

$p^{\wedge} = \arg\max_{p \in D} U(p) = w1\phi(\text{price}(p)) + w$

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

2 r(p)+w3ϕ−1(d(p)) (2)

inventory and could not perform cross-platform price discov-ery. Its NLP pipeline also struggled with code-mixed queries common among Indian users.

C. Web Scraping for E-Commerce Data

Kumar et al. [7] presented a distributed web-scraping archi-tecture using Scrapy and Apache Kafka to harvest pricing data from multiple portals in near-real-time. The system handled anti-scraping mechanisms using rotating proxies and browser emulation. While the data-collection layer was robust, the study did not address query understanding, user interaction, or result personalisation, limiting its practical deployment.

D. Agentic AI and Tool-Augmented LLMs

Wang et al. [4] introduced LLM-based autonomous agents equipped with external tools—web search, calculators, and code interpreters—to complete complex multi-step tasks. Their Re-Act framework demonstrated that language models guided by a Reason-then-Act loop could outperform static chain-of-thought prompting on information retrieval tasks. This work directly inspired the agentic design philosophy adopted in the present paper.

E. Recommendation Systems in E-Commerce

Gupta and Mishra [8] explored collaborative filtering and content-based hybrid recommendation models for product rank-ing. Their model achieved a mean average precision (MAP) of

0.79 but required rich historical user interaction data, which is unavailable for first-time users—a significant cold-start limita-tion.

F. Identified Research Gaps

A synthesis of the above works reveals four unaddressed chal-lenges: (i) no existing system combines real-time multi-platform data collection with a conversational interface and agentic task orchestration; (ii) prior NLP pipelines are not optimised for the Indian e-commerce context or code-mixed language; (iii) ex-isting scraping frameworks lack integration with downstream reasoning and comparison logic; and (iv) personalised recom-mendations are absent in price-comparison tools. The proposed Smart Deal Finder Bot is specifically designed to close all four gaps.

where $\phi(\cdot)$ is a monotone-decreasing normalisation of price, $\phi-1(\cdot)$ is a monotone-decreasing normalisation of delivery time, and $\sum_k w_k = 1$ with $w_k \geq 0$.

The core challenge is to solve Definitions 1 and 2 jointly, in

real time, within a conversational interface, while handling the noise, ambiguity, and dynamism inherent in natural language and live web data.

IV. PROPOSED SYSTEM

The Smart Deal Finder Bot is an agentic AI system that enables end-to-end automated price discovery through a multi-turn conversational interface.

A. Agentic AI Paradigm

The bot adopts an agentic architecture characterised by three core properties:

1. Goal-directedness: The system maintains a persistent repre-sentation of the user's objective across multiple conversation turns and works iteratively to fulfil it.

2. Tool use: The agent autonomously invokes external tools—scrapers, APIs, and database queries—as sub-routines neces-sary to accomplish the goal.

3. Adaptive reasoning: The agent monitors the outcomes of each tool call and adjusts its plan accordingly; if a scraper fails on one platform, the agent retries or substitutes an alter-native data source.

B. System Overview

The bot operates as follows. A user types a product query in natural language through the chat interface. The NLP en-gine interprets the query and populates I(q) as in Equation (1). The agent dispatches parallel scraping or API requests to the identified platforms. Raw data is normalised and stored in the database. The price comparison engine ranks results

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

using Equa-tion (2). Finally, the bot returns a concise, human-readable response listing the top deals along with direct purchase links.

C. Key Differentiators

The system differs from static price-comparison websites in three principal ways: (i) all data is fetched live at query time; (ii) the conversational interface allows iterative refinement across turns; and (iii) the agentic orchestration layer handles failures and retries transparently.

V. SYSTEM ARCHITECTURE

The system comprises six tightly integrated modules arranged in a layered architecture.

A. Chatbot Interface

The front-end is built with React.js and provides a responsive chat window accessible via web browsers and PWA wrappers on mobile devices. WebSocket connections (via Socket.IO) ensure low-latency, bidirectional communication with the Flask-based backend. The interface renders text, product cards with images, and sortable comparison tables.

B. NLP Engine

The NLP Engine is the cognitive core of the system. It operates in a two-stage pipeline: (a) a fine-tuned BERT-based classifier assigns the query to one of six intent classes (price search, product compare, deal alert, review inquiry, specification query, general chitchat); and (b) a Conditional Random Field (CRF) sequence labeller extracts entities of types PRODUCT, BRAND, PRICE RANGE, PLATFORM, and CAT-EGORY.

C. Web Scraping and API Module

This module handles live data acquisition through two com-plementary strategies. Where official APIs are available (e.g., Flipkart Affiliate API), structured JSON responses are parsed directly. For other platforms, a Scrapy-based spider farm renders pages using Playwright to handle JavaScript-heavy storefronts. Rotating residential proxies and request-header randomisation mitigate IP-blocking countermeasures.

D. Price Comparison Engine

The comparison engine receives normalised product listings from the scraping module, computes utility score U (p) for each listing, applies hard filters from user constraints (e.g., maxi-mum budget), and returns the top-k results (default k = 5) in descending order of utility.

E. Database Layer

A dual-database architecture is employed. MongoDB stores unstructured and semi-structured documents such as product descriptions, user session logs, and scraped payloads. MySQL stores structured relational data including user profiles, saved searches, price history, and alert configurations. Redis serves as an in-memory cache for frequently queried products, with a TTL of 15 minutes to balance freshness and performance.

F. Admin Dashboard

The administration panel, built with React.js and Chart.js, provides system operators with real-time visibility into scraping job status, query throughput, NLP model performance metrics, and error logs. Administrators can configure scraping schedules, manage platform connectors, and retrain NLP models through this interface.

VI. METHODOLOGY

A. End-to-End Workflow

The processing pipeline follows seven sequential steps, for-malised in Algorithm 1.

Algorithm 1 Smart Deal Finder Bot Core Pipeline

Require: User natural-language query q

Ensure: Ranked product deal list R

1: I(q) ← NLP Engine(q)

2: if I.intent ∈/ {price search, compare, deal alert} then

3: return GeneralResponse(q)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

4: end if
5: platforms ← ResolvePlatforms(I)
6: Draw ← ParallelFetch(platforms, I.product)
7: Dn ← NormaliseAndClean(Draw)
8: Df ← ApplyConstraints(Dn, I.budget)
9: for each listing p ∈ Df do
10: U (p) ← w1ϕ(price(p)) + w2 r(p) + w3ϕ−1(d(p))
11: end for
12: R ← TopK(Df , k = 5)
13: return FormatResponse(R)

B. Data Collection Protocol

Data collection is performed in parallel using Python's asyncio library. Each scraping job runs with a 10-second timeout. Failed jobs trigger automatic retries with exponential backoff (initial delay 1 s, maximum 3 retries). Successful re-sponses are placed on a shared message queue consumed by the normalisation pipeline.

C. Price Normalisation

Raw prices from different platforms may differ in format (e.g., "Rs. 12,499", "INR 12499.00"). A regular-expression-based parser strips currency symbols, commas, and whitespace, converting all values to a uniform float representation in Indian Rupees.

D. Session Management

User sessions are maintained server-side with a unique ses-sion ID stored in an HTTP-only cookie. The session object records conversation history, the most recent I(q), and active price-alert subscriptions. Multi-turn reasoning is enabled by appending previous turns to the NLP engine's context window on each new message.

VII. ALGORITHMS USED

A. Intent Classification Using BERT

The intent classifier is implemented as a fine-tuned bert-base-uncased model with a fully-connected classification head. Given a tokenised input x = [x1, . . . , xT], the model produces contextual embedding h[CLS] for the classification token. The intent probability vector is:

p = softmax(Wc h[CLS] + bc) (3)

where Wc ∈ R|C|×H , H = 768, and |C| = 6. The model is trained with cross-entropy loss on 18,000 augmented e-commerce queries.

B. Named-Entity Recognition Using BiLSTM-CRF

The NER sub-system employs a BiLSTM-CRF architec-ture. Token embeddings combine pre-trained word vectors with character-level CNN representations. The BiLSTM produces:

The CRF layer predicts the globally optimal tag sequence via the Viterbi algorithm:

ŷ = arg max Score(x, y) (5)
y

C. Price Comparison and Ranking

The utility function in Equation (2) uses min-max normalisa-tion:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

C. NLP Model Training

The BERT intent classifier was fine-tuned on a custom dataset of 12,000 queries augmented with synonym substitution and back-translation to 18,000 samples. Training was con-ducted for five epochs with the AdamW optimiser (learning rate 2 × 10−5, batch size 32) on a single NVIDIA RTX 3060 GPU. The BiLSTM-CRF NER model was trained on 8,500 manually annotated sentences for 30 epochs.

$\phi(\text{price}(p)) = 1 - \text{price}(p) - \text{pricemin}$

$\text{pricemax} - \text{pricemin}$

$\phi - 1(d(p)) = 1 - d(p) - dmin$

$dmax - dmin$

D. Web Scraping Implementation

Scrapy spiders are implemented for five platforms: Amazon India, Flipkart, Snapdeal, Croma, and Reliance Digital. Play-wright is used for JavaScript rendering on Flipkart and Croma, which employ React-based storefronts. A middleware layer in-

The weights [w1, w2, w3] = [0.5, 0.3, 0.2] are initialised from domain knowledge and refined via logistic regression on anonymised click data, giving greatest influence to price while still accounting for quality and delivery speed.

D. Budget-Aware Filtering

Before ranking, a hard-constraint filter removes all listings where price(p) > Bmax, where Bmax is the budget ceiling extracted by the NER module. If no budget is stated, all listings proceed to ranking.

VIII. IMPLEMENTATION

A. Technology Stack

Table 1 summarises the technologies employed.

Layer	Technology	Purpose
Frontend	React.js, Socket.IO	Chat UI, real-time comms
Backend	Python 3.11, Flask	REST API, ses-sion management
NLP Engine	HuggingFace Transformers, spaCy	Intent clf., NER
Scraping	Scrapy, Play-wright, asyn-cio	Dynamic data fetch
Primary DB	MongoDB 6.0	Product docs, logs
Relational DB	MySQL 8.0	User data,price history
Cache	Redis 7.0	Price cache (15 min TTL)
ML Train-ing	PyTorch, scikit-learn	Model fine-tuning
Deployment	Docker, Nginx	Containerised de-ployment

Table 1: Technology Stack of the Smart Deal Finder Bot

DOI: 10.48175/IJARSCT-35387

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

jects random user-agent strings and introduces request delays sampled uniformly from [0.5, 2.0] seconds to mimic human browsing behaviour.

E. Database Schema

MongoDB stores product documents using the following abbreviated schema:

```
{
"product_id": "uuid-string", "name": "string",
"brand": "string",
"platform": "string",
"price": 12499.0,
"rating": 4.3,
"delivery_days": 2,
"url": "https://...", "timestamp": "2024-06-01T10:00:00Z"

}
```

Price history is retained for 30 days to support trend visualisation and deal-alert evaluation.

IX. RESULTS AND ANALYSIS

A. Experimental Setup

Evaluation was performed on a test set of 500 product queries assembled from three sources: (i) 200 queries manually authored to cover diverse product categories; (ii) 200 queries sampled from a public e-commerce chatbot benchmark; and (iii) 100 ambiguous or multi-constraint queries designed to stress-test the NLP pipeline. Queries were executed against the live system during a two-week evaluation window to ensure realistic data conditions.

B. NLP Performance

Table 2 presents the NLP engine's evaluation metrics.

Task	Prec.	Rec.	F1
Intent Classification	94.1%	93.1%	93.6%
NER — PRODUCT	93.4%	90.7%	92.0%
NER — BRAND	92.8%	89.5%	91.1%
NER — PRICE RANGE	95.2%	94.0%	94.6%
NER — PLATFORM	97.1%	96.4%	96.7%
NER Macro Avg.	92.1%	90.4%	91.2%

Table 2: NLP Engine Evaluation Results

B. Backend Architecture

The Flask backend exposes a RESTful API with the follow-ing primary endpoints: POST /api/chat for processing user messages, GET /api/history for fetching session conver-sation history, POST /api/alert for registering price-drop notifications, and GET /api/admin/stats for admin dash-board telemetry. The scraping worker pool is managed by Celery with a Redis broker, enabling asynchronous task execution and automatic retry logic.

The intent classifier achieves an overall F1-score of 93.6%.

The most common misclassifications occur between prod-uct compare and price search intents due to lexical overlap.

C. System Response Time

Table 3 reports end-to-end response times by query type. The cache hit rate observed during evaluation was 42%, reflecting that a substantial proportion of popular product queries benefit from the Redis caching layer.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

Table 3: Average End-to-End Response Time (seconds)

Query Type	Cache Hit	Cache Miss	Avg.
Simple product	0.8	2.1	1.6
Multi-platform compare	1.1	3.4	2.7
Budget-constrained	0.9	2.6	1.9
Ambiguous / multi-turn	1.3	3.8	2.8
Overall	1.0	3.0	2.4

D. Price Coverage and Accuracy

Across all 500 queries, the system successfully retrieved at least one valid product listing for 487 queries (97.4% coverage). Price accuracy was validated by manually cross-checking 100 randomly selected listings against the corresponding platform pages. The system reported prices matching live page prices in 96 of 100 cases (96% accuracy), with four discrepancies attributable to flash-sale price changes occurring within the 15-minute cache window.

E. User Time Saving Study

A user study with 30 participants compared the time required to find the best deal for a given product manually versus using the bot. Manual browsing required an average of 14.3 minutes per query, while the bot reduced this to 3.1 minutes—a reduction of approximately 78.3%. Participants rated the bot's helpfulness at 4.2 out of 5 on average.

F. Ranking Quality

Ranking quality was assessed using Normalised Discounted Cumulative Gain (NDCG@5):

B. Limitations

1. Anti-scraping countermeasures: CAPTCHA challenges, JavaScript obfuscation, and IP rate-limiting periodically disrupt the scraping module.
2. API rate limits: Official APIs impose daily request quotas (e.g., Flipkart Affiliate API: 1,000 calls/day) that constrain throughput during peak usage.
3. Data staleness in cache: The 15-minute Redis TTL means prices from cache may be slightly outdated during highly volatile sale events.
4. Language support: Fully multilingual NLP support for regional Indian languages is not yet implemented.
5. No product verification: The bot relies on platform-reported data and cannot independently verify authenticity or listing accuracy.

XI. FUTURE SCOPE

The current implementation establishes a strong foundation for several planned extensions.

Voice Assistant Integration: Embedding the bot within a voice assistant platform (e.g., Google Assistant) would al-low hands-free product queries, requiring speech-to-text pre-processing and text-to-speech response generation.

Native Mobile Application: A dedicated Android and iOS application with push-notification support would improve the us-ability of the price-alert feature and enable camera-based product search via QR/barcode scanning.

Advanced Personalised Recommendation: Incorporating collaborative filtering on anonymised user interaction logs would allow the system to surface deals aligned with a user's purchase history, even when the query is ambiguous.

Multilingual NLP: Fine-tuning multilingual transformer models (e.g., XLM-R) on corpora covering major Indian languages—Tamil, Telugu, and Marathi—would significantly expand the system's accessibility across diverse demographics.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

where ri is the relevance score of the i-th result. The system achieved an NDCG@5 of 0.847, indicating high-quality ranking with the most relevant deals appearing near the top of results.

X. ADVANTAGES AND LIMITATIONS

A. Advantages

1. Real-time price discovery: Unlike static sites that update nightly, the bot fetches live prices at query time, ensuring access to current deals including flash sales.
2. Conversational accessibility: The natural-language inter-face lowers the barrier to entry; users need not learn platform-specific search syntax.
3. Multi-criteria optimisation: The utility function balances price, rating, and delivery time, avoiding the simplistic "low-est price wins" heuristic.
4. Agentic resilience: Automatic retries and fallback mecha-nisms ensure graceful degradation when individual scrapers fail.
5. Extensibility: The modular architecture allows new platform connectors to be added with minimal code changes.
6. Price alert subscriptions: Users can register for notifications when a product's price drops below a specified threshold.

Reinforcement Learning for Agentic Planning: Replacing rule-based agent orchestration with a reinforcement learning policy trained to minimise response time and maximise user sat-isfaction would enable more adaptive multi-step task execution. Browser Extension: A lightweight browser extension over-laying bot-retrieved deal information directly on platform prod-uct pages would allow users to benefit from cross-platform com-
parison without leaving their current shopping session.

XII. CONCLUSION

This paper has presented the Smart Deal Finder Bot, a novel agentic AI system that addresses the pervasive problem of frag-mented price discovery in India's multi-platform e-commerce ecosystem. By unifying a fine-tuned NLP engine, a parallel web-scraping and API module, a multi-criteria utility-based price comparison engine, and a conversational front-end within a single agentic architecture, the system delivers a practical, scalable, and user-friendly solution to automated deal finding.

Experimental results confirm the system's effectiveness: intent-classification F1-score of 93.6%; NER macro F1-score of 91.2%; average end-to-end response time of 2.4 seconds; price retrieval accuracy of 96%; and ranking quality of NDCG@5 = 0.847. Most significantly, the bot reduces the average consumer deal-finding time by 78.3% compared to manual browsing, trans-lating directly into tangible time and monetary savings for end users.

The agentic paradigm adopted in this work—characterised by goal-directedness, tool use, and adaptive reasoning—proves well-suited to the dynamic, noisy environment of live e-commerce data. The Smart Deal Finder Bot represents a mean-ingful step toward intelligent, autonomous consumer-AI collab-oration in the digital marketplace, and its extensible architecture positions it well for the enhancements outlined in the future scope.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to Prof. Anil Walke for his expert guidance, consistent encouragement, and insightful feedback throughout this project. The authors also thank the Department of Artificial Intelligence and Data Science at ISBM College of Engineering, Pune, for providing computa-tional resources and a stimulating research environment.

REFERENCES

[1] Statista Research Department, "E-commerce worldwide — statistics & facts," Statista, Hamburg, Germany, Tech. Rep., 2023. [Online]. Available: https://www.statista. com/topics/871/online-shopping/

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, May 2026 Impact Factor: 8.2

[2] NASSCOM, "India Internet Report 2023: The Next Decade of Growth," NASSCOM Publications, New Delhi, India, 2023.
[3] S. Acharya and R. Verma, "A survey of conversational AI in e-commerce: Chatbots, virtual assistants and beyond," Journal of Intelligent Information Systems, vol. 57, no. 3, pp. 421–448, 2021.
[4] L. Wang et al., "A survey on large language model based au-tonomous agents," Frontiers of Computer Science, vol. 18, no. 6, p. 186345, 2023.
[5] R. Narayanan, K. Subramanian, and V. S. Iyer, "Cross-platform price comparison using API aggregation: Archi-tecture and evaluation," in Proc. IEEE International Con-ference on Data Engineering Workshops (ICDEW), Dallas, TX, USA, 2020, pp. 128–135.
[6] D. Patel and A. Sharma, "A Rasa-based shopping chatbot for single-platform retail: Design, training, and evaluation," in Proc. International Conference on Computational Intel-ligence and Data Science (ICCIDS), Indore, India, 2021, pp. 312–319.
[7] A. Kumar, P. Mehta, and S. Gupta, "Distributed web scraping for real-time e-commerce price monitoring us-ing Scrapy and Apache Kafka," International Journal of Advanced Computer Science and Applications, vol. 13, no. 4, pp. 552–561, 2022.
[8] N. Gupta and R. Mishra, "Hybrid recommendation sys-tem for e-commerce using collaborative filtering and content-based methods," Expert Systems with Applications, vol. 192, pp. 116–128, 2022.
[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for lan-guage understanding," in Proc. 2019 Conference of the North American Chapter of the Association for Computa-tional Linguistics (NAACL), Minneapolis, MN, USA, 2019, pp. 4171–4186.
[10] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Condi-tional random fields: Probabilistic models for segmenting and labeling sequence data," in Proc. 18th International Conference on Machine Learning (ICML), Williamstown, MA, USA, 2001, pp. 282–289.