Agentic Artificial Intelligence: Concepts, Architecture, Applications, Security, and Future Directions

Ronit Raj Sharma, Akshit Sharma, Dr. Vishal Shrivastava, Dr. Akhil Pandey
Computer Science & Engineering, Arya College of Engineering & I.T., Jaipur, Rajasthan, India

Received:Oct 19, 2026 Accepted:Nov 16, 2026 Published:Nov 17, 2026

***Abstract*—Conventional chatbots can answer a question but often cannot finish the task behind it, so people still move between order systems, policy documents and email by hand. Agentic artificial intelligence (Agentic AI) addresses this gap with software that plans steps toward a goal, uses tools to act, checks the results and adjusts. This review explains the core concepts and traces the progression from rule-based automation and chatbots to tool-using agents and multi-agent systems. It presents a layered reference architecture, describes the closed goal-to-action workflow, and organizes the enabling technologies, including language models, tool connections, memory, retrieval-augmented generation (RAG), and safety monitoring. The paper compares agentic systems with earlier approaches, classifies the main design patterns, surveys applications in customer support, sales, software engineering, office work and research, and summarizes the advantages. Because agents can take actions, security risks such as prompt injection, excessive permissions and unreviewed high-risk steps are examined together with human-approval checkpoints, and practical evaluation measures are described. Future directions are outlined. As a review, the paper reports no new experiments and draws only on published literature.**

***Index Terms*—Agentic AI, AI agents, large language models, tool use, multi-agent systems, human-in-the-loop, prompt injection, retrieval-augmented generation, agent evaluation.**

I. INTRODUCTION

Much routine knowledge work consists of small decisions spread across several applications. In an online store, for example, hundreds of support requests arrive each day. Staff must look up each order, find the return rule that applies, and write a reply, often repeating the same checks. Order details and policies usually sit in different tools, and a basic chatbot, which can produce a fluent reply, typically cannot carry the task through those tools to completion. The result is repetitive work, slower help for customers, and staff time spent on routine lookups instead of complex or personal cases.

Agentic AI responds to this gap. An agent is software that works toward a goal: it plans steps, takes an action with an approved tool, observes what happened and decides what to do next. The reasoning-and-acting pattern introduced by the ReAct method interleaves written reasoning with tool actions, so that each observation can change the next step [1]. Surveys of language-model-based autonomous agents describe such systems through modules such as memory, planning and action [2]. Large language models (LLMs) supply the language understanding and general reasoning ability that make the approach practical [3], and research has shown that models can learn when and how to call external tools [4]. Retrieval-augmented generation (RAG) grounds answers in trusted documents [5], and reflection lets an agent use feedback on earlier attempts [6]. Several specialist agents can also be coordinated by one controller [7]. Because an agent can act, and not only talk, it also introduces new risks. Hidden instructions in untrusted content can hijack its behavior [8], and test environments now exist to study such attacks and defenses [9]. Research prototypes therefore keep a person in the loop for risky steps [10], and agent benchmarks measure whether tasks are actually completed [11].

Existing literature covers individual aspects of this field: reasoning and acting [1], tool learning [4], multi-agent conversation [7], security [9] and evaluation [11]. This paper complements it with one integrated, practice-oriented view. Its contributions are: (i) a conceptual account of how AI systems evolved toward agents; (ii) a layered reference architecture and a goal-to-action workflow; (iii) a comparison and classification of approaches, with applications and advantages; and (iv) a consolidated discussion of security, human control, evaluation, challenges and future directions. The paper is a review and presents no new experiments or measurements.

II. EVOLUTION OF AI SYSTEMS TOWARD AGENTS

Automation has moved through successive stages, each giving software more freedom to decide how to reach a result. Fig. 1 summarizes the progression, and the subsections describe each stage.

A. Rule-Based Automation and Chatbots

The earliest business automation runs fixed scripts: when a trigger occurs, a predefined sequence executes. It is predictable and cheap to verify, but it cannot handle requests outside its rules. Chatbots built on language models added the ability to understand free-form requests and reply in natural language [3]. A basic chatbot, however, mainly produces text. It can describe a return policy, but it cannot check the order, apply the policy and complete the next step on its own.

B. Tool-Augmented Assistants

Connecting a model to tools removes part of this limit. Through APIs the assistant can search, read files or query a database, and retrieval over trusted documents supplies facts it would otherwise have to guess [4], [5]. In many such systems the sequence of calls is still fixed in advance by the developer, so the assistant helps with a step but does not decide the overall route.

C. Agentic and Multi-Agent Systems

Agentic AI lets the model itself choose the route. Given a goal, it breaks the work into smaller steps, calls an approved tool, reads the result and repeats until the goal is met or it is

blocked [1]. When a task is too broad for one agent, a coordinator can assign sub-tasks to specialist agents that each hold only the tools they need [7]. These systems are the most capable stage in Fig. 1, but they also carry the most operational risk, which is why later sections treat control and evaluation in detail.

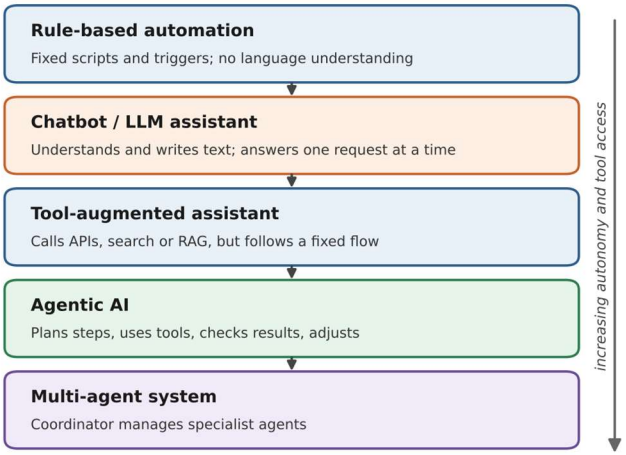

Fig. 1. Evolution from rule-based automation to agentic and multi-agent AI.

D. Human Control as a Design Principle

A recurring theme in current research is that autonomy should be bounded. Human-in-the-loop designs let people review plans, approve sensitive actions and take over when the agent is uncertain [10]. This view treats the agent as a capable assistant working under supervision, not as an unattended replacement for staff.

III. ARCHITECTURE OF AN AGENTIC AI SYSTEM

An agentic system is best understood as a stack of layers, each of which turns a user goal into something closer to a completed action. Fig. 2 shows the stack, and Table I lists the layers and their roles. The arrangement follows the component view common in agent surveys [2], extended with explicit safety and monitoring layers.

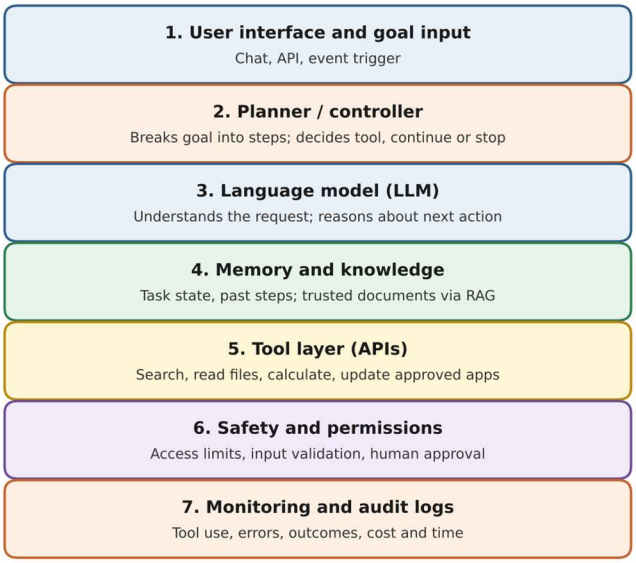

Fig. 2. Layered architecture of an agentic AI system.

A. Interface, Planner and Language Model

The interface receives the goal from a chat window, an API call or an event such as a new support ticket. The planner or controller decides when to use a tool, when to continue and when to stop. It relies on the LLM, which interprets the request, reasons about the situation and proposes the next action [1], [3].

B. Memory and Knowledge

Memory and state record the task, the steps already taken and useful results, so that the agent does not lose context between steps. Knowledge search lets the agent consult trusted documents, such as a return policy, through retrieval instead of relying on what the model happens to remember [5]. Reflection, in which the agent reviews feedback on an earlier attempt and stores a lesson, can be added at this layer [6].

TABLE I
LAYERS OF THE ARCHITECTURE AND THEIR ROLES

Layer	Role
User interface and goal input	Receive the goal from chat, an API or an event trigger
Planner / controller	Break the goal into steps; decide whether to use a tool, continue or stop
Language model (LLM)	Understand the request and reason about the next action
Memory and knowledge	Keep task state and past steps; retrieve trusted documents (RAG)
Tool layer (APIs)	Search, read files, calculate and update approved applications
Safety and permissions	Limit access, validate inputs, request human approval
Monitoring and audit logs	Record tool use, errors, outcomes, time and cost

C. Tool Layer

APIs let the agent search, read files, calculate and update approved applications [4]. Each tool should be described clearly, take validated inputs and expose only the operations

the agent needs, so that the agent cannot reach beyond its intended role.

D. Safety, Monitoring and Integration

The safety layer checks permissions before every tool call, requests human approval for risky actions and keeps secret credentials away from the model. The monitoring layer logs tool use, errors and final outcomes so that people can review what the agent did [10]. Three design considerations cut across the layers. First, access should be limited: each agent receives only the permissions it needs. Second, risky and routine actions should be separated, so that payments, refunds or deletion always pass through approval. Third, every layer should be observable, so that a failure of the agent itself is noticed and not mistaken for success.

IV. WORKING PRINCIPLE

Fig. 3 shows the loop that an agent runs. An agent follows a goal and checks its progress after each step [1]. The main steps are as follows.

1) User goal. The user or an upstream system states what should be achieved, for example resolving a return request. A clear goal and clear limits make later steps easier to check.

2) Planning. The agent breaks the request into smaller steps, such as reading the order, reading the return policy and preparing a reply.

3) Permission check. Before acting, the system verifies that the agent may use the chosen tool and that the inputs are valid. Actions with financial or irreversible effect are flagged for human approval [10].

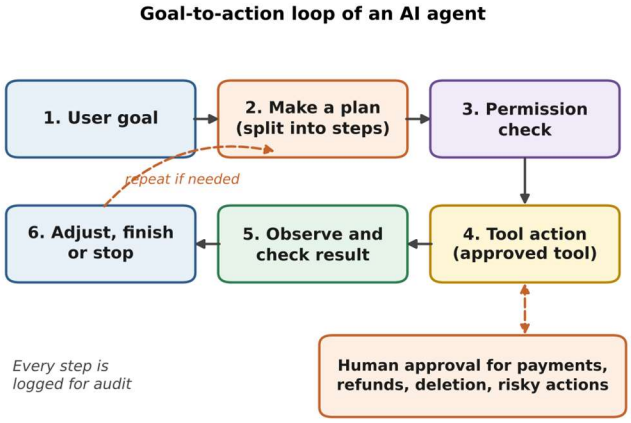

Fig. 3. Goal-to-action workflow of an agent as a loop with permission checks and human approval.

4) Tool action. The agent calls an approved tool, such as an order database, a policy search or a drafting function, and receives its output [4].

5) Observation and checking. The agent reads what happened. A missing order, an error message or a conflicting policy changes what it does next. This feedback loop is what separates agents from fixed scripts [1], [6].

6) Adjustment, finish or stop. The agent repeats the cycle if needed and stops when the goal is met or blocked. When blocked, it should explain the error and ask for help rather than guess.

In the online-store case, the agent reads the order and the return policy, prepares a reply using approved tools, and passes any refund to a staff member for approval. The result is less repetitive work, faster help, and staff control over important actions. Two practical rules keep the loop safe. Retries should be limited, so that a failing step cannot repeat indefinitely, and every step should be logged, so that a run can be reviewed later.

V. CORE ENABLING TECHNOLOGIES

Table II lists the main technologies and their functions. The language model is the reasoning core [3]. Tool connections through APIs allow the agent to search, read files and use applications, and models can be trained to decide when a call is useful [4]. Memory and state keep track of the task, past steps and useful results. The planner or controller decides when to use a tool, continue or stop [1].

Knowledge search with RAG combines retrieval from trusted documents with generation, which helps when facts are needed [5]. Reflection adds learning from feedback [6]. For larger problems, orchestration frameworks support conversations among several agents and allow human participation in the loop [7]. Finally, safety and monitoring components check permissions, log actions and request approval, and test environments make it possible to study attacks and defenses [9].

Choosing among these technologies is a matter of fit rather than fashion. A simple lookup-and-reply task may need only a model with one retrieval tool, whereas a long task across several systems may justify planning and several agents. Practitioners should start with the simplest design that works, add components only when a need appears, and weigh the cost of running and supervising the agent alongside its accuracy.

TABLE II
CORE TECHNOLOGIES AND THEIR FUNCTIONS

Technology	Function in agentic AI
Language model (LLM)	Understands the request and chooses next steps
Tool connections (APIs)	Let the agent search, read files and use apps
Memory and state	Track the task, past steps and useful results
Planner or controller	Decide when to use a tool, continue or stop
Knowledge search (RAG)	Retrieve trusted documents when facts are needed
Reflection	Use feedback on earlier attempts to improve the next one
Multi-agent frameworks	Coordinate specialist agents and human participation
Safety and monitoring	Check permissions, log actions and request approval

VI. COMPARISON WITH EARLIER APPROACHES

Table III compares three approaches on six aspects. The comparison is qualitative, because the real benefit depends on the task, the quality of the tools and the cost of errors. Agentic AI is not always the right choice. When a task is simple, fixed and well understood, a rule-based script is cheaper and easier

to verify. The practical rule is to use an agent when the work needs decisions and tools, not for every simple task.

TABLE III
RULE-BASED AUTOMATION, BASIC CHATBOTS AND AGENTIC AI

Aspect	Rule-based automation	Basic chatbot	Agentic AI
Main output	Fixed action sequence	Text answers	Completed task, with actions taken
Goal handling	Follows predefined rules	Answers one message at a time	Works toward a goal in several steps
Tool use	Hard-coded integrations	Usually none	Chooses among approved tools
Adapts to results	No	Limited	Yes, through a feedback loop
Human oversight	Built into the design	Person reads the output	Approval for risky actions; logs
Main limitation	Breaks on unexpected input	Cannot finish tasks	May fail, err or cost too much

VII. TYPES OF AGENTIC AI APPROACHES

Agentic systems can be grouped along three axes, summarized in Fig. 4.

A. By Structure

A single agent handles the whole task with one set of tools. It is the simplest design, the easiest to debug and the recommended starting point. A multi-agent system uses a coordinator that gives clear tasks to specialist agents, each with only the tools it needs [7]. This splits the work and controls access, although it adds communication overhead and more places where errors can arise. For scale, queues and worker processes allow many requests to be handled during busy periods.

B. By Control Pattern

In a reason-act loop the agent alternates between reasoning and tool actions, adapting after each observation [1]. In plan-then-execute designs the agent writes a complete plan first and then carries it out, which makes the route easy to review but less flexible when surprises occur. Reflection-based designs add a step in which the agent critiques its previous attempt and uses that feedback in the next one [6].

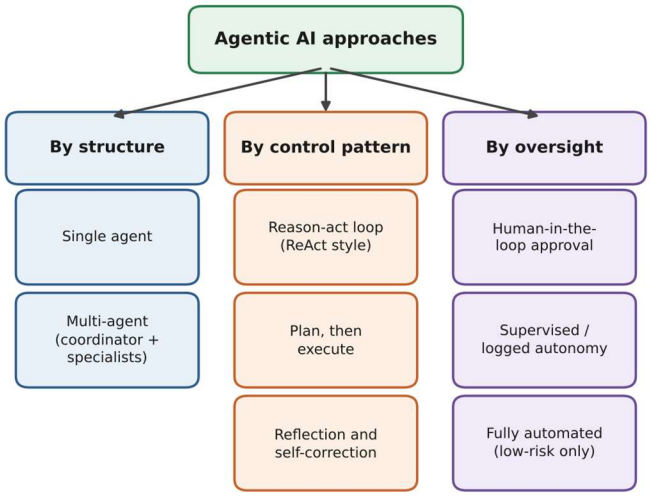

Fig. 4. Classification of agentic AI approaches by structure, control pattern and oversight level.

C. By Oversight Level

Human-in-the-loop designs ask a person to approve plans or sensitive actions [10]. Supervised designs allow the agent to act but keep complete logs for later review. Fully automated operation is suited only to low-risk tasks in which errors are cheap to reverse.

D. Choosing Among Approaches

No category is best in every setting. A sensible path is to start with one agent and add more only when useful. The control pattern should follow the task: a reason-act loop for uncertain environments, plan-then-execute for predictable workflows, and reflection where quality can be improved by a second attempt. The oversight level should follow the cost of a mistake: the higher the cost, the closer the human should be to the decision.

VIII. APPLICATIONS

Table IV lists representative applications. In customer support, an agent checks orders, finds policies and drafts replies, while a staff member approves refunds. In sales and customer relationship management (CRM), agents sort new leads and prepare follow-up messages; a simple flow runs from a new enquiry, through a check of CRM details, to a drafted follow-up. In software teams, agents run tests, inspect errors and suggest fixes. In office work they summarize documents and prepare task updates, and in research tasks they collect sources and organize findings for review.

Across all five areas the pattern is the same. The agent handles lookup, drafting and organization across several tools, while a person owns the final decision where the stakes are meaningful. This division is what allows staff to spend more time on complex or personal work.

TABLE IV
REPRESENTATIVE APPLICATIONS

Domain	Typical agent tasks	Illustrative human checkpoint
Customer support	Check orders, find policies, draft replies	Staff approve refunds and other risky actions
Sales and CRM	Sort new leads, check CRM details, draft follow-ups	Review messages before they are sent
Software teams	Run tests, inspect errors, suggest fixes	Developer reviews proposed changes
Office work	Summarize documents, prepare task updates	Owner confirms the updates
Research tasks	Collect sources, organize findings	Researcher verifies sources and conclusions

IX. ADVANTAGES

When designed and supervised well, agents offer the benefits in Table V. They are potential gains that depend on correct deployment, not guaranteed outcomes, and this paper does not quantify them.

TABLE V
ADVANTAGES OF AGENTIC AI

Advantage	Description
Saves time	Takes care of repeat tasks across several tools
Completes workflows	Moves from a request to useful actions
Adapts to changes	Uses tool feedback to choose the next step
Helps teams	People spend more time on complex or personal work
Scalability	Queues, workers and added agents handle more jobs
Traceability	Logs let people review what the agent did

These advantages depend on one another and on trust. Time savings appear only when staff can rely on the drafts and actions the agent produces, and that reliance in turn requires logs, limits and evidence of reliability. The next section addresses the limits of agents, because they may fail, make mistakes or cost too much.

X. CHALLENGES, SECURITY AND HUMAN CONTROL

Agents can take actions, so access rules and human review are essential. Table VI summarizes the main obstacles. The first is access: each agent should receive only the permissions it needs, and secret credentials must never be exposed to the model. The second is the safety of tool calls, where inputs should be validated before a tool is used. The third is prompt injection, in which hidden instructions in untrusted content, such as an email or a web page, try to steer the agent [8]. The AgentDojo environment was built to evaluate such attacks and defenses for agents that use tools [9], which reflects that the problem is active and unsolved. The fourth control is the approval checkpoint: a human should approve payments, refunds or deletion [10]. The fifth is the audit log, which records tool use, errors and the final outcome.

Beyond security, agents may fail, make mistakes or cost too much. Errors can compound across steps, repeated failed steps can waste time and money, and a fluent but wrong draft can mislead staff who trust it. Multi-agent designs add coordination overhead. Organizations also need staff who understand both the business process and the agent, so that supervision is meaningful. In practice the control sequence is: user goal, agent plan, permission check, tool action, and human approval where needed.

TABLE VI
MAJOR CHALLENGES AND CONTROLS

Challenge	Impact and control
Excessive permissions	Agent may reach beyond its role; give only the permissions needed
Unsafe tool calls	Bad inputs or exposed credentials; validate inputs, hide secrets
Prompt injection	Hidden instructions in untrusted content can steer the agent [8]; do not follow them
Unreviewed risky actions	Mistaken payments, refunds or deletion; add approval checkpoints
Errors and bad loops	Failed steps repeat and waste time and cost; limit retries
Weak traceability	Hard to explain what happened; keep audit logs
Cost and complexity	Agents may cost too much for simple tasks; use them only where decisions and tools are needed

XI. EVALUATION AND PERFORMANCE

Evaluating an agent means judging the finished task and the path taken, not only the quality of a single reply. Benchmarks such as AgentBench assess LLMs acting as agents across several interactive environments [11], and AgentDojo adds security-oriented tests [9]. Table VII lists the things to measure in practice.

Success rate asks whether the task was done and how good the result is. Tool errors ask whether actions were safe, correct and permitted, which is a question of reliability. Time and cost ask whether the agent is worth the effort, which is a question of efficiency. A practical checklist follows from these measures. Test full tasks, checking whether the agent actually finishes the goal. Check tool use to make sure actions are correct and permitted. Catch bad loops by limiting retries and stopping repeated failed steps. Review quality by tracking time, cost, errors and human feedback. Because results depend on the task, the agent and its tools, this paper reports no performance figures.

TABLE VII
WHAT TO MEASURE WHEN EVALUATING AN AGENT

Measure	Key question	Practical check
Success rate	Task done? (task quality)	Test full tasks against the stated goal
Tool errors	Safe actions? (reliability)	Check that calls are correct and permitted
Time and cost	Worth the time? (efficiency)	Track time, cost, errors and human feedback
Attack resistance	Resists hidden instructions?	Test with prompt-injection scenarios [9]

XII. FUTURE DIRECTIONS

Several directions deserve attention. Better human-agent interfaces could let people inspect plans, edit them and approve actions with less effort [10]. Stronger defenses against prompt injection and standard security tests are needed as agents gain access to more tools [8], [9]. Richer benchmarks that measure complete workflows, cost and safety would give

organizations firmer evidence for deployment decisions [11]. Multi-agent frameworks may mature into more reliable ways to divide work [7], although the advice to keep designs simple is likely to remain. Improvements in reflection and learning from feedback could reduce repeated mistakes [6], and better retrieval could keep agents grounded in trusted knowledge [5]. Table VIII lists these directions.

TABLE VIII
FUTURE RESEARCH DIRECTIONS

Direction	Research focus
Human-in-the-loop interfaces	Plan review, editing and approval with low effort [10]
Security and robustness	Defenses against prompt injection; safe tool use [8], [9]
Evaluation	Benchmarks for full workflows, cost and safety [11]
Multi-agent coordination	Reliable task splitting and access control [7]
Learning from feedback	Reflection and self-correction across attempts [6]
Grounded knowledge	Retrieval over trusted, up-to-date documents [5]
Tool learning	Models that learn when and how to call tools [4]

XIII. CONCLUSION

Agentic AI plans, uses tools and checks results, which lets it reduce routine work across connected systems. This review presented the evolution from chatbots to agents, a layered architecture, the goal-to-action workflow, the enabling technologies, a comparison with earlier approaches, a classification of design patterns, and applications in five areas. One agent or several agents can handle different tasks, but their value depends on putting safety first: protecting accounts, limiting permissions and reviewing high-risk actions. Success should be measured through completion, accuracy, time, cost and errors. In short, useful agents combine clear goals, safe tools, feedback and human control. Open problems in security, evaluation and human-centered design remain active research areas, and progress there will determine how widely agents can be trusted in practice.

REFERENCES

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing reasoning and acting in language models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2023. [Online]. Available: https://arxiv.org/abs/2210.03629

[2] L. Wang et al., “A survey on large language model based autonomous agents,” Front. Comput. Sci., vol. 18, no. 6, Art. no. 186345, 2024.

[3] T. B. Brown et al., “Language models are few-shot learners,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 1877–1901.

[4] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, et al., “Toolformer: Language models can teach themselves to use tools,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, 2023. [Online]. Available: https://arxiv.org/abs/2302.04761

[5] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.

[6] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, 2023.

[7] Q. Wu, G. Bansal, J. Zhang, Y. Wu, et al., “AutoGen: Enabling next-gen LLM applications via multi-agent conversations,” in Conf. Lang. Model. (COLM), 2024. [Online]. Available: https://arxiv.org/abs/2308.08155

[8] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection,” in Proc. 16th ACM Workshop Artif. Intell. Secur. (AISec), 2023, pp. 79–90.

[9] E. Debenedetti, J. Zhang, M. Balunović, et al., “AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents,” in Adv. Neural Inf. Process. Syst. (NeurIPS), 2024. [Online]. Available: https://arxiv.org/abs/2406.13352

[10] H. Mozannar, G. Bansal, C. Tan, A. Fourney, et al., “Magentic-UI: Towards human-in-the-loop agentic systems,” Microsoft Research, Tech. Rep. MSR-TR-2025-40, Jul. 2025. [Online]. Available: https://arxiv.org/abs/2507.22358

[11] X. Liu, H. Yu, H. Zhang, Y. Xu, et al., “AgentBench: Evaluating LLMs as agents,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2024. [Online]. Available: https://arxiv.org/abs/2308.03688