

000 AUTOAGENT: A FULLY-AUTOMATED AND ZERO-CODE 001 FRAMEWORK FOR LLM AGENTS

002
003
004
005 **Anonymous authors**
006 Paper under double-blind review
007
008



024 Figure 1: AutoAgent stands out as a new LLM Agent Framework that enables fully automated,
025 zero-code development for complex task automation. Ranking #1 among open-source solutions
026 on the GAIA benchmark, it delivers state-of-the-art RAG performance as a general AI assistant.
027 Its revolutionary approach democratizes AI development - allowing anyone, regardless of coding
028 experience, to create and customize their own agents, tools, and workflows with ease.

ABSTRACT

033 Large Language Model (LLM) Agents have demonstrated remarkable capabilities
034 in task automation and intelligent decision-making, driving the widespread
035 adoption of agent development frameworks such as LangChain and AutoGen. How-
036 ever, these frameworks predominantly serve developers with extensive technical
037 expertise—a significant limitation considering that only 0.03% of the global pop-
038 ulation possesses the necessary programming skills. This stark accessibility gap
039 raises a fundamental question: *Can we enable everyone, regardless of technical*
040 *background, to build their own LLM agents using natural language alone?* To
041 address this challenge, we introduce AutoAgent - a **Fully-Automated** and highly
042 **Self-Developing** framework that enables users to create and deploy LLM agents
043 through **Natural Language Alone**. Operating as an autonomous Agent Operating
044 System, AutoAgent comprises four key components: i) Agentic System Utilities,
045 ii) LLM-powered Actionable Engine, iii) Self-Managing File System, and iv) Self-
046 Play Agent Customization module. This lightweight yet powerful system enables
047 efficient and dynamic creation and modification of tools, agents, and workflows
048 without coding requirements or manual intervention. Beyond its code-free agent
049 development capabilities, AutoAgent also serves as a versatile multi-agent system
050 for **General AI Assistants**. Comprehensive evaluations on the GAIA benchmark
051 demonstrate AutoAgent’s effectiveness in generalist multi-agent tasks, surpassing
052 existing state-of-the-art methods. Furthermore, AutoAgent’s Retrieval-Augmented
053 Generation (RAG)-related capabilities have shown consistently superior performance
/a/ compared to many alternative LLM-based solutions. Code link: <https://anonymous.4open.science/r/AutoAgent-NIPS-D4AF/>.

054
055
056
057

1 INTRODUCTION

058
059
060
061
062
063
064
065
066
The emergence of Large Language Models (LLMs) has revolutionized AI agent development, enabling unprecedented breakthroughs in autonomous task execution and intelligent problem-solving. LLM-powered agents excel at understanding context, making informed decisions, and seamlessly integrating with various tools and APIs. Leading frameworks like LangChain [LangChain \(2023\)](#), AutoGPT [Significant-Gravitas \(2023\)](#), AutoGen [Wu et al. \(2023\)](#), CAMEL [Li et al. \(2023\)](#), and MetaGPT [Hong et al. \(2024a\)](#) have demonstrated remarkable success in automating increasingly complex workflows - from sophisticated web navigation to advanced data analysis and innovative creative content production. By leveraging advanced mechanisms such as role-playing, structured operating procedures, and dynamic agent coordination, these frameworks deliver exceptional problem-solving capabilities while significantly reducing human intervention.067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
Despite remarkable advancements in AI agent development, a significant barrier persists: the creation and optimization of LLM agent systems remains dependent on traditional programming expertise. Current frameworks primarily cater to technically proficient developers who can navigate complex codebases, understand API integrations, and implement sophisticated prompt engineering patterns. This reliance on coding skills creates a substantial accessibility gap, as only 0.03% of the global population possesses the necessary programming expertise to effectively build and customize these agents. Even with well-documented frameworks and development tools, the entry barrier remains dauntingly high for non-technical users. This limitation becomes particularly problematic given the universal need for personalized AI assistants in digital age. Everyone, from business professionals seeking workflow automation to educators designing interactive learning tools, requires customized LLM agents tailored to their specific needs. For instance, a researcher might need an agent specialized in literature review and data analysis, while a content creator might require an agent focused on creative writing and media management. The current paradigm of coding-dependent agent development not only severely restricts the user base but also creates a bottleneck in meeting the diverse and evolving demands for personalized AI assistance. This misalignment between universal needs and limited accessibility calls for a fundamental rethinking of how LLM agents are created and customized.082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
This stark contrast between universal needs and limited accessibility leads us to a fundamental research question: *Is it possible to democratize LLM agent development by enabling Natural Language-based Creation and Customization?* In this work, we aim to realize this vision by introducing AutoAgent, a novel framework that fundamentally reimagines agent development as a fully automated, language-driven process requiring zero programming expertise. To realize this vision, AutoAgent operates as an autonomous Agent Operating System with three key capabilities: 1) **Natural Language-Driven Multi-Agent Building** - automatically constructing and orchestrating collaborative agent systems purely through natural dialogue, eliminating the need for manual coding or technical configuration; 2) **Self-Managing Workflow Generation** - dynamically creating, optimizing and adapting agent workflows based on high-level task descriptions, even when users cannot fully specify implementation details; and 3) **Intelligent Resource Orchestration** - providing unified access to tools, APIs, and computational resources via natural language while automatically managing resource allocation and optimization. Through this innovative architecture, AutoAgent democratizes LLM agent development while maintaining enterprise-grade sophistication, transforming a traditionally complex engineering task into an intuitive conversation accessible to all users.097
098
099
100
101
102
103
104
105
106
107
To enable fully-automated and zero-code LLM agent development, AutoAgent introduces several synergistic technical innovations that form a complete framework: First, the **Agentic System Utilities** provides a foundational multi-agent architecture, where specialized web, code, and file agents collaborate seamlessly to handle diverse real-world tasks. At its core, the **LLM-powered Actionable Engine** serves as the system's brain, supporting flexible integration of any LLM provider through both direct and transformed tool-use paradigms for robust action generation. To address the critical challenge of information management, the **Self-Managing File System** enhances overall system capability by automatically converting diverse data formats into queryable vector databases, enabling efficient information access across all operations. Additionally, the **Self-Play Agent Customization** not only transforms natural language requirements into executable agents through structured XML schemas, but also automatically generates optimized workflows through iterative self-improvement, eliminating the need for manual agent programming or workflow design. Together, these innovations enable AutoAgent to democratize agent development while maintaining production-level robustness.

108 AutoAgent’s exceptional capabilities have been rigorously validated through comprehensive empirical
 109 evaluation. In standardized benchmarks, it secured a strong second place on the Generalist Agent
 110 Benchmark (GAIA), while significantly outperforming state-of-the-art RAG approaches on the
 111 Retrieval-Augmented Generation benchmark. Beyond these quantitative achievements, extensive
 112 case studies demonstrated AutoAgent’s robust self-development capabilities across diverse real-world
 113 scenarios, highlighting its practical value in automated agent development.

114

115

2 RELATED WORK AND PRELIMINARIES

117

118 LLM-empowered agents have revolutionized AI systems through tool invocation capabilities. Frameworks like LangChain [LangChain \(2023\)](#), AutoGPT [Significant-Gravitas \(2023\)](#), CAMEL [Li et al. \(2023\)](#),
 119 MetaGPT [Hong et al. \(2024a\)](#), and OpenAgent [Xie et al. \(2023\)](#) demonstrate diverse capabilities: CAMEL pioneered role-playing communication, AutoGen [Wu et al. \(2023\)](#) developed
 120 LLM combinations for problem-solving, MetaGPT [Hong et al. \(2024a\)](#) integrated Standardized
 121 Operating Procedures, and OpenAgent [Xie et al. \(2023\)](#) provided specialized web agents. However,
 122 these frameworks require coding skills and domain expertise, limiting accessibility to non-technical
 123 users. We propose a new paradigm that democratizes agent development through natural language
 124 interactions, making agent technology accessible regardless of technical background.

125

126

LLM-Empowered Agent. The task-solving process of Large Language Model (LLM) agents can be formalized as a Markov Decision Process (MDP), providing a comprehensive framework for modeling their interaction with the environment. Defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P(\cdot), \mathcal{E})$, the MDP captures the agent’s state space \mathcal{S} , action space \mathcal{A} , observation space \mathcal{O} , state transition function $P(\cdot)$, and the set of environments \mathcal{E} it can interact with. At each time step, the LLM agent observes the current state, selects an action based on its policy, interacts with the environment, and updates its state, often referred to as the agent’s “context”. The mapping from state to action can follow two primary paradigms: **Tool-Use** [Yao et al. \(2024\)](#), where the agent utilizes external capabilities, and **ReAct** [Yao et al. \(2023\)](#) (Non-tool-use), where the agent generates the next action solely based on its internal language model. This MDP formulation provides a powerful framework for understanding, analyzing, and designing LLM-empowered agents capable of tackling a wide range of complex, multi-step tasks.

127

128

Generalist Multi-Agent System. Multi-agent systems (MAS) overcome single agent limitations in complex problems by leveraging diverse capabilities of multiple specialized agents. Generalist Multi-Agent Systems employ agent teams coordinated by an Orchestrator to solve varied tasks through collaborative intelligence.

129

130

In a Generalist MAS, there are multiple agents, denoted as $\pi_0 : S_0 \rightarrow A_0, \pi_1 : S_1 \rightarrow A_1, \dots, \pi_n : S_n \rightarrow A_n$. Within each agent’s action set, there exists a special **transfer action** $\hat{A}_i \in A_i$, which enables the delegation of tasks to other agents. The key challenge in a MAS lies in designing an effective **Task Transfer Mechanism**, which organizes different agents through appropriate transfer actions. We define such an agent organization mechanism as the “MAS Design Pattern”. A common design is the **Orchestrator-Workers** paradigm [Fourney et al. \(2024\)](#); [Anthropic \(2024\)](#), where the **Orchestrator** comprehends the task and distributes subtasks to **Workers** via transfer actions. The Workers, acting as sub-agents, execute the subtasks and return the results to the Orchestrator through transfer actions.

131

132

For tasks with deterministic steps, workflow-driven mechanisms have emerged as a particularly effective approach. Notable examples include GPTSwarm [Zhuge et al.](#), which models workflows as computational graphs for complex data operations, and specialized systems for mathematical reasoning [Wang et al. \(2023\)](#) and code generation [Wang et al. \(2024a\)](#). These advances have enabled strong capabilities across various applications, including GUI interaction [Hong et al. \(2024b\)](#); [Xu et al. \(2024\)](#), software development [Wang et al. \(2024b\)](#); [Yang et al. \(2024\)](#), web browsing [Drouin et al. \(2024\)](#); [Song et al. \(2024\)](#), and embodied tasks [Li et al. \(2024\)](#), demonstrating the versatility and potential of LLM-powered agent systems.

133

134

Workflow Design in Generalist MAS. While Generalist Multi-Agent Systems offer high flexibility, particularly for open-ended and complex tasks, they also come with higher costs, increased complexity, and potential for compounding errors [Anthropic \(2024\)](#). For tasks with deterministic processes and expert domain knowledge, fixed workflows can be a more stable and effective approach.

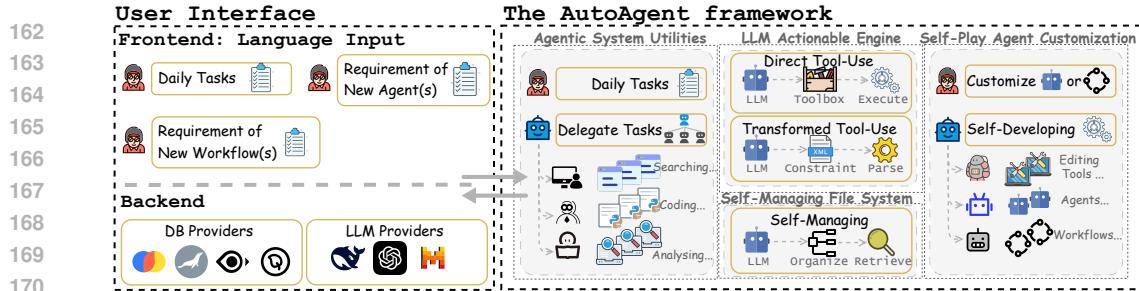


Figure 2: AutoAgent is a fully automated, language-driven generalist agent system. The core components that enable this include the Agentic System Utilities, the LLM-powered Actionable Engine, the Self-Managing File System, and the Self-Play Agent Customization module.

A workflow in a MAS is defined as $\mathcal{W} = w_{i,j} : \pi_i \xrightarrow{c_k} \pi_j$, where π_i and π_j represent agents, and c_k is the transfer condition. Effective workflow design involves defining conditional transfer equations between agents based on downstream tasks, ensuring smooth and coordinated transmission - the “Workflow Design Pattern”. Common patterns include **Routing** (directing tasks through a sequence), **Parallelization** (distributing subtasks concurrently), and **Evaluator-Optimizer** (using agents to assess and refine) [Anthropic \(2024\)](#). These can create stable and efficient task-solving in MAS.

Fully-Automated Generalist MAS. While the successful design and execution of both multi-agent systems and complex workflows typically require substantial expert knowledge and engineering expertise, the goal of AutoAgent is to fully automate this process. The key challenge lies in seamlessly bridging the gap from high-level user requirements to the practical implementation of effective MAS and workflow solutions - all through natural language interactions.

3 THE AUTOAGENT FRAMEWORK

AutoAgent is designed to be the automated operating system for LLM agents and general AI assistant. Inspired by modern computer operating systems, AutoAgent consists of key components that enable seamless natural language-driven agent development and task execution, as illustrated in Fig 2. Its **Agentic System Utilities** provide foundational building blocks for complex agent-driven tasks, while the **LLM-powered Actionable Engine** forms the central brain, understanding inputs and orchestrating multi-agent coordination. The **Self-Managing File System** manages structured storage and retrieval of user multi-modal data, and the **Self-Play Agent Customization** empowers users to generate specialized, tailored agents and workflows through natural language, without any coding requirements. Collectively, these robust capabilities make AutoAgent a versatile and powerful platform, powering a variety of autonomous agent-based solutions for diverse applications.

3.1 AGENTIC SYSTEM UTILITIES

The AutoAgent framework employs a modular, multi-agent architecture to address the key challenge of developing intelligent personal assistant agents capable of seamlessly integrating and coordinating diverse capabilities, from web browsing and information retrieval to data analysis and code execution. This design choice, which comprises specialized agents for web, coding, and file management tasks, as well as an orchestrator agent to decompose and delegate user requests, enables the agentic system utilities to serve as a versatile and extensible foundation that can adapt to a wide range of user requirements, facilitating the rapid development of tailored, agent-driven solutions. Detailed system prompts and tool definitions for **Agentic System Utilities** can be found in Appendix Sec A.

3.1.1 ORCHESTRATOR AGENT

The Orchestrator Agent is the primary interface for interacting with the user. It receives tasks from the user, comprehends the tasks, decomposes them into sub-tasks, and delegates these sub-tasks to appropriate sub-agents using the handoff tools [OpenAI \(2024\)](#). Once a sub-agent completes a sub-task, it returns the result to the Orchestrator also using the handoff tool. Based on the task completion status, the Orchestrator continues to assign the next sub-task to a suitable agent. This iterative process continues until the entire task is completed. The Orchestrator, designed with the handoff mechanism, is a simple yet effective solution, eliminating the need for complex prompts to handle task planning.

3.1.2 DEDICATED AGENT MODULE

The basic Agent system can be broadly categorized into three types: **web interaction**, **code execution**, and **file analysis**. We have constructed three functionally complementary dedicated agent modules: Web Agent, Coding Agent, and Local File Agent, each designed for these respective tasks.

216 **The Web Agent provides a flexible and extensible web-toolkit**, enabling the agent to perform a
 217 variety of web-based tasks, from general web searches to file downloads. Its core functionalities
 218 include web searching, page navigation, content browsing, and file downloading, which are abstracted
 219 into 10 high-level operational interfaces (e.g., `click`, `web_search`, `visit_url`, etc.). This
 220 module is built on BrowserGym [Drouin et al. \(2024\)](#), creating a browser environment that combines
 221 low-level code-driven actions to implement high-level tools, thereby enhancing the extensibility of
 222 tool definitions.

223 **The Coding Agent is a comprehensive solution for code-related tasks**, capable of handling a wide
 224 range of applications—from data processing and numerical computation to machine learning and
 225 system management. It offers diverse tools and supports execution in a secure, isolated interactive
 226 terminal environment. All code execution results are fed back to the agent via terminal output, with
 227 pagination support (e.g., using commands like `terminal_page_up`, `terminal_page_down`,
 228 and `terminal_page_to`), effectively bypassing the context length limitations of LLMs. To
 229 ensure security, all operations run within a Docker sandbox, with optional integration of third-party
 230 secure execution platforms like E2B [E2B \(2024\)](#).

231 **The Local File Agent is designed to uniformly process various local multimodal file types** (e.g.,
 232 text documents, audio, video, spreadsheets) and convert them into structured formats for analysis.
 233 This agent standardizes different file formats (e.g., `.pdf`, `.mp4`, `.csv`, `.docx`) and displays them
 234 in a terminal-like Markdown browser with pagination, enabling efficient handling of long texts or
 235 complex file structures for everyday tasks.

236 These three agent modules work collaboratively, forming light-weight yet complete basic Agent
 237 System, laying the foundation for future functional expansion.

238 3.2 LLM-POWERED ACTIONABLE ENGINE

240 As the CPU executes instructions, manages resources, and coordinates processes in an OS, the
 241 LLM-powered actionable engine can understand natural language, generate plans, and coordinate
 242 tasks across agents. This enables seamless human-agent collaboration and task completion.

243 We utilize LitELL M [BerriAI \(2024\)](#) to standardize LLM requests through an OpenAI-like interface,
 244 supporting 100+ models from various providers. For agent collaboration, the LLM receives all
 245 action-observation pairs up to time t as state s_t to determine the next action. These pairs serve as
 246 system RAM, facilitating efficient retrieval and enabling language-driven system coordination.

247 3.2.1 GENERATING ACTIONABLE REFLECTIONS

248 We generate reflections (*i.e.*, actions) based on LLM context, which can be broadly categorized into
 249 two distinct approaches that leverage the language model’s capabilities.

251 **Direct Tool-Use Paradigm.** This approach is suitable for commercial LLMs or LLM serving
 252 platforms that support tool-use. These LLMs can directly generate a parsed next-step tool to execute
 253 based on the provided tool set and the current state, reducing errors during the tool parsing phase.
 254 However, this method heavily relies on the optimization of the third-party platform’s capabilities.

255 **Transformed Tool-Use Paradigm.** This approach does not rely on the LLM’s inherent tool-use capa-
 256 bilities. Leveraging the superior code-generation abilities of modern LLMs, we transform the tool-use
 257 paradigm into a structured XML code generation task, *e.g.*, `<function=function_name>`
 258 `<parameter=parameter_1>value_1</parameter> ... </function>`. This
 259 structured output is then parsed to extract critical information like tool arguments and others. It
 260 improves the performance of commercial models with suboptimal tool-use capabilities and enables
 261 the integration of open-source LLMs into the system, providing greater flexibility and customization.

262 3.3 SELF-MANAGING FILE SYSTEM

263 The file system in AutoAgent is a vector database that LLM agents can retrieve and understand. In our
 264 design framework, users can upload text files in any format (e.g., `.pdf`, `.doc`, `.txt`) or compressed
 265 archives and folders containing any text files. The system tools in the file system automatically
 266 convert these files into a consistent text format and store them in a user-defined collection within
 267 the vector database (using the `save_raw_docs_to_vector_db` tool). This enables agents to
 268 self-manage their database memory and perform efficient and accurate retrieval and generation using
 269 tools like `query_db` and `answer_query`. The detailed definitions of the tools are presented in
 Tab 4.

270 3.4 SELF-PLAY AGENT CUSTOMIZATION
271

272 To allow users to customize tools and agents for specific scenarios or build their own multi-agent
273 systems and workflows, it is designed as a code-driven, controllable self-programming agent framework.
274 By implementing constraints, error-handling mechanisms, and customized workflows, it enables
275 controlled code generation, facilitating the creation of tools, agents, and workflows. The AutoAgent
276 supports two distinct modes: agent creation without workflow and agent creation with workflow.

277 3.4.1 AGENT CREATION WITHOUT WORKFLOW
278

279 Building effective multi-agent systems often requires domain-specific expertise, such as in-depth
280 knowledge of financial regulations or healthcare protocols. However, this level of specialized know-
281 how may not always be available to users. For example, in the financial services, constructing
282 a multi-agent system to automate complex investment portfolio management would necessitate
283 expertise in areas like asset allocation, risk modeling, and regulatory compliance.

284 To address this challenge, our AutoAgent provides a powerful workflow-based mode allowing users to
285 generate sophisticated agent systems with minimal domain expertise. In this mode, the user provides
286 high-level descriptions of the desired agent(s), such as the agent’s name and a brief sentence-level
287 description for the expected agent functionalities. AutoAgent then uses this input to automatically
288 generate the appropriate agent(s) and the necessary workflow(s) to orchestrate their collaborative
289 efforts. This is all done based on the current state of the framework, including the available tools,
290 agents, and workflows. The key steps in this workflow-based agent generation approach are:

291 **•Analyze Requirements and Existing Components.** The process begins by carefully analyzing the
292 user’s requirements in the context of the existing tools and agents already available in the system.
293 This in-depth analysis is performed using the specialized profiling agent, which helps thoroughly
294 assess the current capabilities and resources that can be leveraged to fulfill the user’s needs.

295 **•Analyze Tools and Structure Agents.** Based on the comprehensive analysis performed, the system
296 then carefully determines the need for creating new tools, meticulously evaluates whether existing
297 tools can be effectively utilized, and subsequently structures the seamless collaboration between
298 multiple agents as appropriate. This ensures the optimal and comprehensive use of available resources,
299 ultimately leading to the efficient agent system design.

300 **•Generate Detailed XML Agent Specifications.** This step is designed to generate structured XML
301 code that accurately represents the detailed agent creation requirements. This comprehensive XML
302 representation captures the necessary information, including the agent’s functionality, dependencies,
303 and interactions, to enable the seamless and efficient subsequent process of agent generation.

304 **Optimized Tool Creation with Third-Party APIs.** The Tool Editor Agent can seamlessly integrate
305 various third-party APIs, such as LangChain, RapidAPI, and Hugging Face, to create powerful tools.
306 It expertly utilizes advanced retrieval techniques to search for and surface relevant API documentation,
307 including comprehensive details like names, descriptions, and invocation methods. The robust system
308 currently supports an extensive collection of 145 APIs from 8 diverse categories in RapidAPI,
309 LangChain [LangChain \(2023\)](#), and a wide range of models from 9 categories in Hugging Face. Future
310 plans include seamlessly integrating more cutting-edge platforms like Composio [Composio \(2024\)](#).

311 The agent also generates tool code based on its knowledge, automatically checking for syntax errors.
312 It designs test cases, runs the tool, and verifies functionality. If the tool fails, the agent automatically
313 debugs the code until successful. This targeted approach allows a more customized and adaptable
314 tool set, rather than a bloated, integrated system.

315 **Agent Creation and Execution.** When the user’s requirements involve multiple agents focused on
316 different tasks, the Agent Editor Agent automatically identifies this need and performs the necessary
317 multi-step agent creation operations. After all agents are successfully created, the system invokes the
318 `create_orchestrator_agent` tool to generate an orchestrator agent that connects the required
319 agents. This orchestrator adheres to the Orchestrator-Workers MAS design pattern, with a system
320 prompt that includes task descriptions, sub-task decomposition rules, and other scenario-specific
321 details. Detailed algorithms and system prompts are provided in Appendix Sec A.6.1.

322 3.4.2 AGENT CREATION WITH WORKFLOW
323

324 When users have specific requirements for a MAS’s workflow and domain knowledge, AutoAgent
325 allows a tailored approach. In this mode, users provide descriptions of the desired agent(s) and
326 specify the tasks they want the created agent(s) or workflows to accomplish. AutoAgent then uses this

324 information about the target tasks to generate not just the individual agent(s), but also the necessary
 325 workflow(s) to coordinate their collaborative efforts in achieving the specified objectives.
 326

327 Traditional graph-based methods often require strict adherence to graph theory principles [Zhuge et al. \(2024\)](#), [LangChain \(2024\)](#), [Hu et al. \(2024\)](#), [Zhang et al. \(2024\)](#), a task challenging for LLMs when
 328 generating workflows. To overcome these challenges, AutoAgent adopts an event-driven approach
 329 where we model each agent's task-solving as an event. By leveraging event listening and triggering
 330 mechanisms, AutoAgent enables seamless collaboration between agents, offering greater flexibility
 331 and adaptability compared to rigid graph structures.
 332

333 **Constructing New Workflows.** The process of creating a new workflow is itself a multi-agent
 334 workflow. The Workflow Form Agent analyzes the requirements and existing tools/agents to determine
 335 if new agents need to be created, which agents should form the workflow, and what the listening and
 336 triggering logic between events should be. It then generates structured XML code.
 337

338 During the parsing phase, a robust error detection mechanism verifies whether the generated workflow
 339 form complies with system constraints (e.g., constraints on the `on_start` event). If the constraints
 340 are not satisfied, detailed error messages are sent back to the Workflow Form Agent as feedback for
 341 regeneration. If the constraints are satisfied, the workflow form is passed to the Workflow Editor
 342 Agent, which creates new agents (if needed), constructs the new workflow, and executes it on the
 343 task. Detailed algorithms and system prompts are shown in Appendix Sec [A.6.2](#).
 344

4 EVALUATION

4.1 EVALUATION FOR A GENERALIST AGENT SYSTEM

Dataset and Evaluation Protocols.

345 The GAIA benchmark [Mialon et al. \(2024\)](#) evaluates General AI Assistants through 466 test and
 346 165 validation questions across 3 difficulty levels, assessing Reasoning, Multi-Modality Handling,
 347 Web Browsing, and Tool-Use Proficiency. We evaluated AutoAgent on GAIA's validation set using
 348 success rate as the metric, measuring task completion performance on human-like challenges.
 349

350 **Baseline Methods.** The baselines we selected are divided into two categories: Open-Source: FRIDAY [Wu et al. \(2024\)](#),
 351 Magentic-1 [Fourney et al. \(2024\)](#), Multi-Agent Experiment v0.1 (powered by AutoGen) [Microsoft \(2024\)](#), Hugging-
 352 Face Agents [HuggingFace \(2024\)](#), Langfun Agent [Google \(2024\)](#); Closed-Source: TapeAgent, AgentIM, Trase Agent [Trase \(2024\)](#), Omne, Barcelona¹, and the
 353 h2oGPTe Agent [H2O.ai \(2024\)](#). These diverse baselines represent the current state-of-the-art in open-source and proprietary
 354 multi-agent systems, providing a comprehensive landscape for evaluating the performance and capabilities of our proposed
 355 AutoAgent framework.
 356

357 Table 1: Performance comparison between the baseline models and our AutoAgent on the GAIA benchmark.
 358 The results we report are those published on the GAIA.

Agent Name	Avg.	L1	L2	L3
TapeAgent v0.1	33.94	47.17	34.88	3.85
FRIDAY	34.55	45.28	34.88	11.54
Magentic-1	36.97	54.72	33.72	11.54
AgentIM	37.58	50.94	36.05	15.38
Multi-Agent Exp v0.1	39.39	54.72	38.37	11.54
AgentIM v1.1	40.00	50.94	40.70	15.38
Trase Agent	40.00	47.17	40.70	23.08
HuggingFace Agents	44.24	58.49	43.02	19.23
Magentic-1 (01)	46.06	56.60	46.51	23.08
omne	46.06	60.38	44.19	23.08
Trase Agent v0.2	47.27	58.49	46.51	26.92
Barcelona v0.1	50.30	62.26	50.00	26.92
Langfun Agent v2.0	54.55	60.38	59.30	26.92
h2oGPTe Agent v1.6.8	63.64	67.92	67.44	42.31
AutoAgent	55.15	71.70	53.49	26.92

359 **Implementation Details.** To address tasks in the GAIA benchmark, we utilize a combination of the System Utilities of the Model and the
 360 Tool Editor Agent from the Agentic-SDK. The basic agents first attempt to complete the task while
 361 collecting relevant information and reflections. If successful, the result is directly returned. If not, the
 362 Tool Editor Agent creates new tools to continue the task. During validation, Claude-Sonnet-3.5 is
 363 used by default.
 364

365 **Evaluation Results and Analysis.** The results in Table 1 reveal the following key observations:
 366

- 367 • **Obs.1. Overall Superiority of AutoAgent:** Our method significantly outperforms all open-
 368 source agent systems and achieves performance close to the latest agent system, h2oGPTe Agent
 369 v1.6.8 (submitted on December 16, 2024), securing a stable position in the top 2 rankings. Notably,
 370

371 ¹TapeAgent, AgentIM, Omne, and Barcelona are anonymous.
 372

378 our approach demonstrates superior performance on Level 1 tasks compared to all state-of-the-art
 379 baselines, becoming the first method to achieve over 70% accuracy rate. This success is attributed to
 380 the well-designed System Utilities and the stable interaction of basic agents with the environment,
 381 enabling efficient solutions to everyday simple tasks.

382 • **Obs.2. Effectiveness of Key Components:** Specifically, our framework demonstrates significantly
 383 superior performance compared to Magentic-1 [Fourney et al. \(2024\)](#), a recent representative open-
 384 source MAS, and FRIDAY, a classic self-improved framework. While Magentic-1 leverages the
 385 powerful reasoning capabilities of ol-preview to design complex Orchestrator Agent (also the Coder
 386 Agent), our framework emphasizes the stability of interactions between sub-agents and their respective
 387 environments, as well as the precision of tool definitions. Under these conditions, the Orchestrator
 388 Agent achieves better results with simple prompts and handoff tools.

389 4.2 EVALUATION OF AUTOAGENT ON THE RETRIEVAL-AUGMENTED GENERATION TASK

390 **Benchmark Dataset and Evaluation Protocols.** To test the basic functionalities of the AutoAgent,
 391 we use the RAG task as the testing benchmark. MultiHop-RAG [Tang & Yang \(2024\)](#) is a dataset
 392 designed to evaluate RAG capabilities, requiring the RAG methods to gather information from
 393 multiple sources and generate responses, which aligns with the file functionality logic of AutoAgent.
 394 We evaluate using two metrics: **Accuracy (Acc)** measures response consistency with expected
 395 answers (e.g., “ChatGPT” or “OpenAI’s ChatGPT” are both correct for “Which AI tool reached
 396 100M daily users in March?”). **Error (Err)** counts confident but incorrect responses (e.g., answering
 397 “Bard” to the above query).

398 **Baseline Methods.** The baselines represent a diverse range of LLM-based RAG techniques. The
 399 chunk methods, such as NaiveRAG [Mao et al. \(2020\)](#) and HyDE [Gao et al. \(2022\)](#), utilize the original
 400 text segmentation. The graph methods, including MiniRAG [Fan et al. \(2025\)](#) and LightRAG [Guo et al. \(2024\)](#),
 401 manage files as sophisticated graphs. In contrast, Langchain’s Agentic RAG [LangChain \(2023\)](#) innovatively
 402 accesses files through intelligent software agents. These baselines cover a wide
 403 array of strategies for leveraging large language models to retrieve and generate robust responses.

404 **Implementation Details.** We used gpt-4o-mini [OpenAI \(2023\)](#) as the LLM and text-embedding-
 405 3-small for embeddings. We followed MultiHopRAG [Tang & Yang \(2024\)](#) for text chunking, with
 406 256-token chunks and top-6 retrieval. This leverages the gpt-4o-mini’s language abilities while
 407 text-embedding-3-small provides retrieval, with MultiHopRAG’s chunking managing information
 408 effectively.

409 **Evaluation Results and Analysis.** We summarize the key observations from Table 2.

410 • **Superior Performance of AutoAgent.** The results clearly demonstrate the superior performance
 411 of our proposed AutoAgent model compared to other baselines on the Multihop-RAG
 412 task. By leveraging a more flexible and adaptive agent-based framework, AutoAgent is able to dynamically
 413 orchestrate the retrieval and reasoning process, outperforming even other baselines.

414 • **AutoAgent vs. LangChain.** Our method significantly outperforms LangChain, which is also an
 415 agentic RAG. This is due to AutoAgent’s more flexible framework, where agents do not need to rely
 416 on predefined workflows and tools to execute file search tasks. The proposed model can orchestrate
 417 workflows on the fly during the search process, leading to more efficient and accurate results.

418 4.3 AUTOAGENT’S PERFORMANCE ON OPEN-ENDED TASKS

419 This section thoroughly explores the capabilities of the AutoAgent framework in generating agents
 420 and workflows based on even vague, natural language inputs across various scenarios. To illustrate
 421 the breadth of AutoAgent’s abilities, we will examine its performance on tasks of varying difficulty -
 422 from the creation of a single agent to the orchestration of multiple, coordinated agents. (We put the
 423 results of single agent in Appendix A.7.1)

424 **Task with Multi-Agents.** To further validate AutoAgent’s capability to generate agents and integrate
 425 third-party tools, we tasked it with creating a Financial Agent based on the following requirements:

426 I want to create ‘Financial Agent’ that can help me to do two kinds of
 427 tasks:

Table 2: Evaluation of AutoAgent for RAG.

Method	acc	err	acc	err
Chunk-Based	NaiveRAG		HyDE	
	53.36%	12.28%	56.59%	16.55%
Graph-Based	MiniRAG		LightRAG	
	57.81%	34.78%	58.18%	35.40%
Agent-Based	Langchain		AutoAgent	
	62.83%	20.50%	73.51%	14.20%

Table 3: Comparison between single LLMs and the AI-generated Majority Voting workflow.

Models	gpt-4o 0806	claude-3.5-sonnet 1022	deepseek-v3	Majority Voting Workflow (3 models)
pass@1	66.4	66.4	74.2	75.6

1. Manage the private financial docs. I have a folder that contain the financial docs in my local machine, and I want to help me to manage them.
 2. Search the financial information online. You may help me to:
 - get balance sheets for a given ticker over a given period.
 - get cash flow statements for a given ticker over a given period.
 - get income statements for a given ticker over a given period.

Building a Comprehensive Financial Agent. The Agent Form Agent created two agents: the **Document Manager Agent** and **Market Research Agent**, with XML structure shown in List 18. The Tool Editor Agent created tools: `get_balance_sheet`, `get_cash_flow`, `get_income_statement`, and `analyze_financial_data`. The Agent Editor Agent then composed these agents and established a Financial Analysis Orchestrator for coordination.

The Financial Analysis Orchestrator leveraged new tools and existing capabilities to conduct research on local documents and external data sources. This process produced a comprehensive research report, shown in List 19 in the Appendix. The agent trajectory is detailed in Tab 6 in the Appendix. Despite encountering a `SyntaxError` during initial creation, the Agent Editor successfully self-debugged and completed the task, demonstrating the robustness of AutoAgent system.

Workflow Generation. Scaling Test-Time Compute has been validated as a superior approach for solving reasoning problems. However, manually constructing workflows poses a high barrier to entry. We aim to explore whether AutoAgent’s automatic creation of agents and workflows can bridge the gap between the idea of Test-Time Compute and the implementation of workflows. Taking the majority voting method with multiple models as an example:

I want to create a workflow that can help me to solving the math problem. The workflow should:

1. Parallelize solving the math problem with the same ‘Math Solver Agent’ using different language models (‘gpt-4o’, ‘claude-3-5-sonnet’, ‘deepseek-chat’)
2. Aggregate the results from the ‘Math Solver Agent’ and return the final result using majority voting.

Potential Test-Time Scaling Law. Upon receiving the requirements, the Workflow Form Agent generated an XML-formatted workflow table (List 20). This table includes two new agents: **Math Solver Agent** and **Vote Aggregator Agent**. After validation, the Agent Editor Agent created agents. The Workflow Editor Agent then constructed a new workflow based on the form and conducted tests. To validate the workflow’s practicality, we performed comparative experiments on the MATH-500 dataset Lightman et al. (2024) using 3 LLMs (gpt-4o-20240806, claude-3.5-sonnet-20241022, deepseek-v3) and a Majority Voting workflow. As shown in Tab 3, the generated workflow performs significantly better than state-of-the-art baselines. We selected cases from deepseek-v3 (Tab 5) where AutoAgent’s workflow effectively corrected errors through multi-model collaboration, demonstrating its potential to establish scaling laws in LLMs.

5 CONCLUSION

The AutoAgent framework marks a significant advancement in democratizing LLM-powered agent technology, making it accessible to the non-programming majority. By bridging high-level natural language requirements with the practical implementation of multi-agent systems and workflows, MetaChain empowers users to create, customize, and deploy agents, tools, and workflows without requiring substantial technical expertise. Its modular architecture, versatile Agentic System Utilities, and LLM-powered Actionable Engine work together to enable seamless automation of agent development and task execution. Unique features such as the Self-Organizing File System and Self-Play Agent Customization further enhance AutoAgent’s capabilities, allowing for dynamic agent evolution and task-specific optimization. Extensive evaluations demonstrate AutoAgent’s superior performance, highlighting its transformative potential in making LLM capabilities accessible to a broad user base.

486 REFERENCES
487

488 Anthropic. Building effective agents. <https://www.anthropic.com/research/building-effective-agents>, 2024. URL <https://www.anthropic.com/research/building-effective-agents>.

489

490

491 BerriAI. Litellm: Proxy server (llm gateway) to call 100+ llm apis in openai format. <https://github.com/BerriAI/litellm>, 2024. URL <https://github.com/BerriAI/litellm>.

492

493

494 Composio. Composio: Production ready toolset for ai agents. <https://github.com/ComposioHQ/composio>, 2024. URL <https://github.com/ComposioHQ/composio>.

495

496

497

498 Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom
499 Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable
500 are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico
501 Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp
502 (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of
503 *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL
504 <https://proceedings.mlr.press/v235/drouin24a.html>.

505

506 E2B. Secure open source cloud runtime for ai apps & ai agents. <https://github.com/e2b-dev/e2b>, 2024. URL <https://github.com/e2b-dev/e2b>.

507

508 Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. Minirag: Towards extremely simple
509 retrieval-augmented generation. *arXiv preprint arXiv:2501.06713*, 2025.

510

511 Adam Journey, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu,
512 Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang,
513 Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema
514 Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks. *CoRR*,
515 abs/2411.04468, 2024.

516

517 Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without
518 relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.

519

520 Google. Langfun: Oo for llms. <https://github.com/google/langfun>, 2024. URL
521 <https://github.com/google/langfun>.

522

523 Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-
524 augmented generation. 2024.

525

526 H2O.ai. Autonomous agentic ai: execute multi-step workflows autonomously. <https://h2o.ai/platform/enterprise-h2ogpte/#AgenticAI>, 2024. URL <https://h2o.ai/platform/enterprise-h2ogpte/#AgenticAI>.

527

528 Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao
529 Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng
530 Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent
531 collaborative framework. In *The Twelfth International Conference on Learning Representations*,
2024a. URL <https://openreview.net/forum?id=VtmBAGCN7o>.

532

533 Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan
534 Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for GUI agents.
In *CVPR*, pp. 14281–14290. IEEE, 2024b.

535

536 Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *CoRR*, abs/2408.08435,
537 2024.

538 HuggingFace. Transformers documentation: Agents and tools. <https://huggingface.co/docs/transformers/agents>, 2024. URL <https://huggingface.co/docs/transformers/agents>.

539

540 LangChain. Langchain: Build context-aware reasoning applications. <https://github.com/langchain-ai/langchain>, 2023. URL <https://github.com/langchain-ai/langchain>.

541

542

543 LangChain. Langgraph: sbuild resilient language agents as graphs. <https://github.com/langchain-ai/langgraph>, 2024. URL <https://github.com/langchain-ai/langgraph>.

544

545

546

547 Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbulin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In *NeurIPS*, 2023.

548

549 Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. In *NeurIPS 2024*, 2024.

550

551

552

553 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*. OpenReview.net, 2024.

554

555

556 Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Generation-augmented retrieval for open-domain question answering. *arXiv preprint arXiv:2009.08553*, 2020.

557

558

559 Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *ICLR*. OpenReview.net, 2024.

560

561

562 Microsoft. Multi-agent experiment v0.1 msr ai frontiers (autogen team members). https://aka.ms/gaia_multiagent_v01_march_1st, 2024. URL https://aka.ms/gaia_multiagent_v01_march_1st.

563

564

565 OpenAI. Gpt-4 technical report, 2023.

566

567 OpenAI. Educational framework exploring ergonomic, lightweight multi-agent orchestration. <https://github.com/openai/swarm>, 2024. URL <https://github.com/openai/swarm>.

568

569

570 Significant-Gravitas. Autogpt. <https://github.com/Significant-Gravitas/AutoGPT>, 2023.

571

572 Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. Beyond browsing: Api-based web agents. *arXiv preprint arXiv:2410.16464*, 2024. URL <https://arxiv.org/abs/2410.16464>.

573

574

575 Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *CoRR*, abs/2401.15391, 2024.

576

577 Trase. Meet trase systems, the ai agent platform. <https://www.trasesystems.com/>, 2024. URL <https://www.trasesystems.com/>. Accessed: 2025-01-15.

578

579

580 Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves LLM search for code generation. *CoRR*, abs/2409.03733, 2024a.

581

582

583 Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2024b. URL <https://arxiv.org/abs/2407.16741>.

584

585

586

587

588 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net, 2023.

589

590

591

592 Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155, 2023.

593

594 Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao
 595 Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement.
 596 *arXiv preprint arXiv:2402.07456*, 2024.

597

598 Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning
 599 Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong,
 600 and Tao Yu. Openagents: An open platform for language agents in the wild, 2023.

601 Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu,
 602 and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. 2024.
 603 URL <https://arxiv.org/abs/2412.04454>.

604

605 John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan,
 606 and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering.
 607 In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL
 608 <https://arxiv.org/abs/2405.15793>.

609 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao.
 610 React: Synergizing reasoning and acting in language models. In *ICLR*. OpenReview.net, 2023.

611 Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for
 612 tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024.

613

614 Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen
 615 Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin
 616 Wu. Aflow: Automating agentic workflow generation. *CoRR*, abs/2410.10762, 2024.

617 Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbulin, and Jürgen
 618 Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International
 619 Conference on Machine Learning*.

620 Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbulin, and Jürgen
 621 Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *ICML*. OpenReview.net,
 622 2024.

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

A APPENDIX

In the supplementary materials, we provide a detailed technical description of the 'Agentic System Utilities' implementation within our AutoAgent framework.

A.1 SYSTEM-LEVEL TOOLS

To empower our diverse array of system-level agents, we have carefully curated and predefined seven distinct categories of powerful tools. These tools span a wide range of functionalities, including coding, web browsing, file management, creating new tools, agents, and workflows, as well as natural language question answering for documents. The detailed names and comprehensive descriptions of these versatile tools are presented in Table 4.

Table 4: List of detailed information of system-level tools.

Tool Name	Category	Description
run_python	Coding	Run a python script.
execute_command	Coding	Execute a command in the system shell. Use this function when there is a need to run a system command, and execute programs.
gen_code_tree_structure	Coding	Generate a tree structure of the code in the specified directory. Use this function when you need to know the overview of the codebase and want to generate a tree structure of the codebase.
create_directory	Coding	Create a directory if it does not exist. Use this function when there is a need to create a new directory.
list_files	Coding	List all files and directories under the given path if it is a directory. Use this function when there is a need to list the contents of a directory.
write_file	Coding	Write content to a file. Use this function when there is a need to write content to an existing file.
create_file	Coding	Create a file with the given path and content. Use this function when there is a need to create a new file with initial content.
read_file	Coding	Read the contents of a file and return it as a string. Use this function when there is a need to check an existing file.
terminal_page_to	Coding	Move the viewport to the specified page index. The index starts from 1. Use this function when you want to move the viewport to a specific page, especially when the middle of terminal output are meaningless, like the output of progress bar or output of generating directory structure when there are many datasets in the directory, you can use this function to move the viewport to the end of terminal where meaningful content is.

702	terminal_page_down	Coding	Scroll the viewport DOWN one page-length in the current terminal. Use this function when the terminal is too long and you want to scroll down to see the next content.
703			
704			
705			
706	terminal_page_up	Coding	Scroll the viewport UP one page-length in the current terminal. Use this function when the terminal is too long and you want to scroll up to see the previous content.
707			
708			
709			
710	input_text	Web	Types the given text value into the specified field.
711			
712			
713	click	Web	Clicks the mouse on the target with the given element bid.
714			
715	page_down	Web	Scrolls the entire browser viewport one page DOWN towards the end.
716			
717	page_up	Web	Scrolls the entire browser viewport one page UP towards the beginning.
718			
719	history_back	Web	Navigates back one page in the browser's history. This is equivalent to clicking the browser back button.
720			
721			
722			
723	history_forward	Web	Navigates forward one page in the browser's history. This is equivalent to clicking the browser forward button.
724			
725			
726	visit_url	Web	Navigate directly to a provided URL using the browser's address bar. Prefer this tool over other navigation techniques in cases where the user provides a fully-qualified URL (e.g., choose it over clicking links, or inputting queries into search boxes).
727			
728			
729			
730			
731			
732	web_search	Web	Performs a web search on ' https://www.google.com.sg/?hl=en&gl=US ' with the given query.
733			
734			
735	sleep	Web	Wait a short period of time. Call this function if the page has not yet fully loaded, or if it is determined that a small delay would increase the task's chances of success.
736			
737			
738			
739	get_page_markdown	Web	Get the markdown content of the current page. Use this tool if you need to watch the Youtube video, Wikipedia page, or other pages that contain media content. Note that this tool can only be used after you have visited a valid page.
740			
741			
742			
743			
744	open_local_file	File	Open a local file at a path in the text-based browser and return current viewport content.
745			
746			
747	page_up_markdown	File	Scroll the viewport UP one page-length in the current file and return the new viewport content.
748			
749			
750	page_down_markdown	File	Scroll the viewport DOWN one page-length in the current file and return the new viewport content.
751			
752			
753	find_next	File	Scroll the viewport to next occurrence of the search string.
754			
755			

756	visual_question_answering	File	This tool is used to answer questions about attached images or videos.
757			
758	find_on_page_ctrl_f	File	Scroll the viewport to the first occurrence of the search string. This is equivalent to Ctrl+F.
759			
760	list_tools	Tools Edit	List all plugin tools in the MetaChain.
761			
762	create_tool	Tools Edit	Create a plugin tool.
763			
764	delete_tool	Tools Edit	Delete a plugin tool.
765			
766	run_tool	Tools Edit	Run a tool with the given code.
767			
768	search_trending_models_on_huggingface	Tools Edit	Search trending models on Hugging Face. Use this tool when you want to create a tool that uses Hugging Face models, only support the following tags: ['audio-text-to-text', 'text-to-image', 'image-to-image', 'image-to-video', 'text-to-video', 'text-to-speech', 'text-to-audio', 'automatic-speech-recognition', 'audio-to-audio'].
769			
770			
771			
772			
773			
774	get_hf_model_tools_doc	Tools Edit	Get the detailed information of a model on Hugging Face, such as the detailed usage of the model containing the model's README.md. You should use this tool after you have used 'search_trending_models_on_huggingface' to find the model you want to use.
775			
776			
777			
778			
779			
780	get_api_plugin_tools_doc	Tools Edit	Retrieve satisfied tool documents based on the query text.
781			
782	list_agents	Agents Edit	List all plugin agents in the MetaChain.
783			
784	read_agents	Agents Edit	Get detailed information of plugin agents in the MetaChain.
785			
786			
787	delete_agent	Agents Edit	Delete a plugin agent.
788			
789	run_agent	Agents Edit	Run a plugin agent.
790			
791	create_agent	Agents Edit	Use this tool to create a new agent or modify an existing agent.
792			
793			
794	create_orchestrator_agent	Agents Edit	Use this tool to create a orchestrator agent for the given sub-agents. You MUST use this tool when you need to create TWO or MORE agents and regard them as a whole to complete a task.
795			
796			
797			
798	create_workflow	Workflows Edit	Create a workflow.
799			
800	list_workflows	Workflows Edit	List all workflows in the MetaChain.
801			
802	run_workflow	Workflows Edit	Run a workflow.
803			
804			
805			
806			
807			
808			
809			

810	save_raw_docs_to_vector	RAG	Save the raw documents to the vector database. The documents could be: - ANY text document with the extension of pdf, docx, txt, etcs. - A zip file containing multiple text documents - a directory containing multiple text documents All documents will be converted to raw text format and saved to the vector database in the chunks of 4096 tokens.
818	query_db	RAG	Retrieve information from the database. Use this function when you need to search for information in the database.
821	modify_query	RAG	Modify the query based on what you know. Use this function when you need to modify the query to search for more relevant information.
825	answer_query	RAG	Answer the user query based on the supporting documents.
827	can_answer	RAG	Check if you have enough information to answer the user query.

830 A.2 WEB AGENT

832 The specific tools and system prompt for implementing the Web Agent are as follows:

834 Listing 1: Tools of **Web Agent**

835 [click, page_down, page_up, history_back, history_forward,
 836 web_search, input_text, sleep, visit_url, get_page_markdown,
 837 transfer_back_to_orchestrate_agent]

839 Listing 2: System Prompt of **Web Agent**

840 Review the current state of the page and all other information to
 841 ↳ find the best possible next action to accomplish your goal. Your
 842 ↳ answer will be interpreted and executed by a program, make sure to
 843 ↳ follow the formatting instructions.
 844 Note that if you want to analyze the YouTube video, Wikipedia page,
 845 ↳ or other pages that contain media content, or you just want to
 846 ↳ analyze the text content of the page in a more detailed way, you
 847 ↳ should use '**get_page_markdown**' tool to convert the page information
 848 ↳ to markdown text. And when browsing the web, if you have downloaded
 849 ↳ any files, the path of the downloaded files will be
 850 ↳ '**/workplace/downloads**', and you CANNOT open the downloaded files
 851 ↳ directly, you should transfer back to the '**System Orchestrate**
 852 ↳ **Agent**', and let '**System Orchestrate Agent**' to transfer to '**Local**
 853 ↳ **File Agent**' to open the downloaded files.
 854 When you think you have completed the task the '**System Orchestrate**
 855 ↳ **Agent**' asked you to do, you should use
 856 ↳ '**transfer_back_to_orchestrate_agent**' to transfer the conversation
 857 ↳ back to the '**System Orchestrate Agent**'. And you should not stop to
 858 ↳ try to solve the user's request by transferring to '**System**
 859 ↳ **Orchestrate Agent**' only until the task is completed.

859 A.3 LOCAL FILE AGENT

860 The Local File Agent is equipped with a tailored set of tools and system prompts to enable it to
 861 efficiently manage and interact with files and directories. This specialized toolkit includes:

863 Listing 3: Tools of **Local File Agent**

```
864 | [open_local_file, page_up_markdown, page_down_markdown,  
865 | find_on_page_ctrl_f, find_next, visual_question_answering,  
866 | transfer_back_to_orchestrate_agent]
```

Listing 4: System Prompt of Local File Agent

870 You are a file surfer agent that can handle local files.
871
872 You can only access the files in the folder '**/workplace**' and when you
873 → want to open a file, you should use absolute path from root like
874 → '**/workplace/...**'.
875
876 Note that '**open_local_file**' can read a file as markdown text and ask
877 → questions about it. And '**open_local_file**' can handle the following
878 → file extensions: `[".html", ".htm", ".xlsx", ".pptx", ".wav",
879 → ".mp3", ".flac", ".pdf", ".docx"]`, and all other types of text
880 → files.
881
882 But IT DOES NOT HANDLE IMAGES, you should use
883 → '**visual_question_answering**' to see the image.
884
885 If the converted markdown text has more than 1 page, you can use
886 → '**page_up**', '**page_down**', '**find_on_page_ctrl_f**', '**find_next**' to
887 → navigate through the pages.
888
889 When you think you have completed the task the '**System Orchestrate**
890 → **Agent**' asked you to do, you should use
891 → '**transfer_back_to_orchestrate_agent**' to transfer the conversation
892 → back to the '**System Orchestrate Agent**'. And you should not stop to
893 → try to solve the user's request by transferring to '**System**
894 → **Orchestrate Agent**' only until the task is completed.
895
896 If you are unable to open the file, you can transfer the conversation
897 → back to the '**System Orchestrate Agent**', and let the '**Coding Agent**'
898 → try to solve the problem by coding.

A.4 CODING AGENT

The specific tools and system prompts for implementing the Coding Agent are as follows:

Listing 5: Tools of **Coding Agent**

```
901 [gen_code_tree_structure, execute_command, read_file, create_file,
902 write_file, list_files, create_directory, run_python,
903 terminal_page_up, terminal_page_down, terminal_page_to,
904 transfer_back_to_orchestrate_agent]
```

Listing 6: System Prompt of **Coding Agent**

You are a helpful programming assistant that can write and execute code. You are working in the folder: `'/workplace'`, and you can only access the files in this folder.

Your can leverage your capabilities by using the specific functions listed below:

1. Creating project structures based on the user requirement using function `'create_directory'`.
2. Writing clean, efficient, and well-documented code using function `'create_file'` and `'write_file'`.
3. You must run python scripts using function `'run_python'` rather than using the `'execute_command'` function.
4. Exam the project to re-use the existing code snippets as much as possible, you may need to use

```

918 functions like 'list_files', 'read_file' and 'write_file'.
919 5. Writing the code into the file when creating new files, do not
920 → create empty files.
921 6. Before you write code into the existing files, you should first
922 → read the file content using function 'read_file' and reserve the
923 → original content as much as possible.
924 7. Decide whether the task requires execution and debugging before
925 → moving to the next or not.
926 8. Generate the commands to run and test the current task, and the
927 → dependencies list for this task.
928 9. You only write Python scripts, don't write Jupiter notebooks which
929 → require interactive execution.
930 10. Note that every path you read, write, or search should be the
931 → absolute path (starting with "/").
932 11. If you should use programming other than Python, you should use
933 → the 'write_file' function to write the code into a file, and then
934 → use the 'execute_command' function to run the code.
935 12. If the terminal output is too long, you should use
936 → 'terminal_page_up' to move the viewport up, 'terminal_page_down' to
937 → move the viewport down, 'terminal_page_to' to move the viewport to
938 → the specific page of terminal where the meaningful content is.
939 Note that you can use this agent to make complex computation, write a
940 → api request, and anything else that can be done by writing code.
941 When you think you have completed the task the 'System Orchestrate
942 → Agent' asked you to do, you should use
943 → 'transfer_back_to_orchestrate_agent' to transfer the conversation
944 → back to the 'System Orchestrate Agent'. And you should not stop to
945 → try to solve the user's request by transferring to 'System
946 → Orchestrate Agent' only until the task is completed.
947
948
949 A.5 ORCHESTRATOR AGENT
950
951 The specific tools and system prompt for implementing the Orchestrator Agent are as follows:
952
953 Listing 7: Tools of Orchestrator Agent
954
955 [transfer_to_local_file_agent, transfer_to_web_agent,
956 transfer_to_coding_agent]
957
958
959 Listing 8: System Prompt of Orchestrator Agent
960
961 You are a helpful assistant that can help the user with their request.
962 Based on the state of solving user's task, your responsibility is to
963 → determine which agent is best suited to handle the user's request
964 → under the current context, and transfer the conversation to that
965 → agent. And you should not stop to try to solve the user's request
966 → by transferring to another agent only until the task is completed.
967
968 There are three agents you can transfer to:
969 1. use 'transfer_to_local_file_agent' to transfer to 'Local File
970 → Agent', it can help you to open any type of local files and browse
971 → the content of them.
972 2. use 'transfer_to_web_agent' to transfer to 'Web Agent', it can
973 → help you to open any website and browse any content on it.
974 3. use 'transfer_to_coding_agent' to transfer to 'Coding Agent', it
975 → can help you to write code to solve the user's request, especially
976 → some complex tasks.
977

```

972 A.6 DETAILED IMPLEMENTATION OF “SELF-PLAY AGENT CUSTOMIZATION” IN AUTOAGENT
973974 A.6.1 AGENT CREATION WITHOUT WORKFLOW
975976 The following details demonstrate the specific process of Agent Creation without Workflow (Alg 1),
977 as well as the tools and system prompts used in the implementation of Agent Profiling Agent, Tool
978 Editor Agent, and Agent Editor Agent.979 **Algorithm 1** Controllable Workflow of Creating Tools and Agents
980

```

981 1: Input: requirements  $\mathcal{R}$ , existing tool set  $\mathcal{A}$ , existing agent set  $\pi$ , task  $\mathcal{T}$  (optional), Maximum
982   iterations of attempts  $M$ .
983 2: Output: the response of requirements for creating tools and agents to solve the task.
984 3:  $\text{AgentProfile} = \text{agent\_profile\_agent}(\mathcal{R}|\mathcal{A}, \pi)$ 
985 4:  $\text{ParsingResults}_0 = \text{form\_parsing\_function}(\text{AgentProfile}|\mathcal{R}, \mathcal{A}, \pi)$ 
986 5: for  $i = 1$  to  $M - 1$  do
987   6:   if  $\text{ParsingResults}_{i-1}$  is ‘Success’ then
988     7:     break
989   8:   else
990     9:      $\text{AgentProfile} = \text{agent\_profile\_agent}(\mathcal{R}, \text{ParsingResults}_{i-1}|\mathcal{A}, \pi)$ 
991     10:     $\text{ParsingResults}_i = \text{form\_parsing\_function}(\text{AgentProfile}|\mathcal{R}, \mathcal{A}, \pi)$ 
992   11:   end if
993   12: end for
994 13: if There are new tools need to be created in  $\text{AgentProfile}$  then
995   14:   // Automatically create tools and test them.
996   15:    $\text{ToolsResults}_0 = \text{tool\_editor\_agent}(\text{AgentProfile}|\mathcal{R}, \mathcal{A}, \pi)$ 
997   16:   for  $i = 1$  to  $M - 1$  do
998     17:     if  $\text{ToolsResults}_{i-1}$  is ‘Success’ then
999       18:         break
1000     19:     else
1001       20:        $\text{ToolsResults}_i = \text{tool\_editor\_agent}(\text{AgentProfile}, \text{ToolsResults}_{i-1}|\mathcal{R}, \mathcal{A}, \pi)$ 
1002     21:     end if
1003   22:   end for
1004 23: end if
1005 24: // Automatically create agents and run them on the given task.
1006 25:  $\text{AgentsResults}_0 = \text{agent\_editor\_agent}(\text{AgentProfile}, \mathcal{T}|\mathcal{R}, \mathcal{A}, \pi)$ 
1007 26: for  $i = 1$  to  $M - 1$  do
1008   27:   if  $\text{AgentsResults}_{i-1}$  is ‘Success’ then
1009     28:       break
1010   29:   else
1011     30:      $\text{AgentsResults}_i = \text{agent\_editor\_agent}(\text{AgentProfile}, \mathcal{T}, \text{AgentsResults}_{i-1}|\mathcal{R}, \mathcal{A}, \pi)$ 
1012   31:   end if
1013 32: end for

```

1012 **Listing 9: System Prompt of Agent Profiling Agent**1013 You are an agent specialized in creating agent forms for the
1014 ↪ MetaChain framework.1015 Your task is to analyze user requests and generate structured
1016 ↪ creation forms for either single or multi-agent systems.

1018 KEY COMPONENTS OF THE FORM:

1. **<agents>** – Root element containing all agent definitions
2. **<system_input>** – Defines what the system receives
 - Must describe the overall input that the system accepts
 - For single agent: Same as `agent_input`
 - For multi-agent: Should encompass all possible inputs that will
↪ be routed to different agents
3. **<system_output>** – Specifies system response format

```

1026
1027     - Must contain exactly ONE key-description pair
1028     - <key>: Single identifier for the system's output
1029     - <description>: Explanation of the output
1030     - For single agent: Same as agent_output
1031     - For multi-agent: Should represent the unified output format from
1032       ↳ all agents
1033
1034 4. <agent> - Individual agent definition
1035     - name: Agent's identifier
1036     - description: Agent's purpose and capabilities
1037     - instructions: Agent's behavioral guidelines
1038       * To reference global variables, use format syntax: {variable_key}
1039       * Example: "Help the user {user_name} with his/her request"
1040       * All referenced keys must exist in global_variables
1041     - tools: Available tools (existing/new)
1042     - agent_input:
1043       * Must contain exactly ONE key-description pair
1044       * <key>: Identifier for the input this agent accepts
1045       * <description>: Detailed explanation of the input format
1046     - agent_output:
1047       * Must contain exactly ONE key-description pair
1048       * <key>: Identifier for what this agent produces
1049       * <description>: Detailed explanation of the output format
1050
1051 5. <global_variables> - Shared variables across agents (optional)
1052     - Used for constants or shared values accessible by all agents
1053     - Variables defined here can be referenced in instructions using
1054       ↳ {key}
1055     - Example:
1056       ````xml
1057       <global_variables>
1058         <variable>
1059           <key>user_name</key>
1060           <description>The name of the user</description>
1061           <value>John Doe</value>
1062         </variable>
1063       </global_variables>
1064       ````

1065     - Usage in instructions: "You are a personal assistant for
1066       ↳ {user_name}."
1067
1068 IMPORTANT RULES:
1069
1070     - For single agent systems:
1071       * system_input/output must match agent_input/output exactly
1072
1073     - For multi-agent systems:
1074       * system_input should describe the complete input space
1075       * Each agent_input should specify which subset of the system_input
1076         ↳ it handles
1077       * system_output should represent the unified response format
1078
1079 Existing tools you can use is:
1080 ...
1081
1082 Existing agents you can use is:
1083 ...
1084
1085 EXAMPLE 1 - SINGLE AGENT:
1086
1087 User: I want to build an agent that can answer the user's question
1088   ↳ about the OpenAI products. The document of the OpenAI products is
1089   ↳ available at '/workspace/docs/openai_products/'.
1090 The agent should be able to:
1091 1. query and answer the user's question about the OpenAI products
1092   ↳ based on the document.

```

```

1080 2. send email to the user if the sending email is required in the
1081   ↪ user's request.
1082
1083 The form should be:
1084 <agents>
1085   <system_input>
1086     Questions from the user about the OpenAI products. The document
1087     ↪ of the OpenAI products is available at
1088     ↪ `/workspace/docs/openai_products/`.
1089   </system_input>
1090   <system_output>
1091     <key>answer</key>
1092     <description>The answer to the user's question.</description>
1093   </system_output>
1094   <agent>
1095     <name>Helper Center Agent</name>
1096     <description>The helper center agent is an agent that serves as
1097     ↪ a helper center agent for a specific user to answer the
1098     ↪ user's question about the OpenAI products.</description>
1099     <instructions>You are a helper center agent that can be used to
1100     ↪ help the user with their request.</instructions>
1101     <tools category="existing">
1102       <tool>
1103         <name>save_raw_docs_to_vector_db</name>
1104         <description>Save the raw documents to the vector
1105         ↪ database. The documents could be:
1106         - ANY text document with the extension of pdf, docx, txt,
1107         ↪ etcs.
1108         - A zip file containing multiple text documents
1109         - a directory containing multiple text documents
1110         All documents will be converted to raw text format and
1111         ↪ saved to the vector database in the chunks of 4096
1112         ↪ tokens.</description>
1113       </tool>
1114       <tool>
1115         <name>query_db</name>
1116         <description>Query the vector database to find the answer
1117         ↪ to the user's question.</description>
1118       </tool>
1119       <tool>
1120         <name>modify_query</name>
1121         <description>Modify the user's question to a more
1122         ↪ specific question.</description>
1123       </tool>
1124       <tool>
1125         <name>answer_query</name>
1126         <description>Answer the user's question based on the
1127         ↪ answer from the vector database.</description>
1128       </tool>
1129       <tool>
1130         <name>can_answer</name>
1131         <description>Check if the user's question can be answered
1132         ↪ by the vector database.</description>
1133     </tools>
1134     <tools category="new">
1135       <tool>
1136         <name>send_email</name>
1137         <description>Send an email to the user.</description>
1138       </tool>
1139     </tools>
1140     <agent_input>
1141       <key>user_question</key>
1142       <description>The question from the user about the OpenAI
1143       ↪ products.</description>

```

```

1134     </agent_input>
1135     <agent_output>
1136         <key>answer</key>
1137         <description>The answer to the user's question.</description>
1138     </agent_output>
1139   </agent>
1140 </agents>

1141 EXAMPLE 2 - MULTI-AGENT:
1142
1143 User: I want to build a multi-agent system that can handle two types
1144 ↪ of requests for the specific user:
1145 1. Purchase a product or service
1146 2. Refund a product or service
1147 The specific user worked for is named John Doe.
1148
1149 The form should be:
1150 <agents>
1151   <system_input>
1152     The user request from the specific user about the product or
1153     ↪ service, mainly categorized into 2 types:
1154     - Purchase a product or service
1155     - Refund a product or service
1156   </system_input>
1157   <system_output>
1158     <key>response</key>
1159     <description>The response of the agent to the user's
1160     ↪ request.</description>
1161   </system_output>
1162   <global_variables>
1163     <variable>
1164       <key>user_name</key>
1165       <description>The name of the user.</description>
1166       <value>John Doe</value>
1167     </variable>
1168   </global_variables>
1169   <agent>
1170     <name>Personal Sales Agent</name>
1171     <description>The personal sales agent is an agent that serves
1172     ↪ as a personal sales agent for a specific user.</description>
1173     <instructions>You are a personal sales agent that can be used
1174     ↪ to help the user {user_name} with their
1175     ↪ request.</instructions>
1176     <tools category="new">
1177       <tool>
1178         <name>recommend_product</name>
1179         <description>Recommend a product to the
1180         ↪ user.</description>
1181       </tool>
1182       <tool>
1183         <name>recommend_service</name>
1184         <description>Recommend a service to the
1185         ↪ user.</description>
1186       </tool>
1187       <tool>
1188         <name>conduct_sales</name>
1189         <description>Conduct sales with the user.</description>
1190       </tool>
1191     </tools>
1192     <agent_input>
1193       <key>user_request</key>
1194       <description>Request from the specific user for purchasing a
1195       ↪ product or service.</description>
1196     </agent_input>
1197     <agent_output>

```

```

1188     <key>response</key>
1189     <description>The response of the agent to the user's
1190     ↪ request.</description>
1191   </agent_output>
1192 </agent>
1193 <agent>
1194   <name>Personal Refunds Agent</name>
1195   <description>The personal refunds agent is an agent that serves
1196   ↪ as a personal refunds agent for a specific user.</description>
1197   <instructions>Help the user {user_name} with a refund. If the
1198   ↪ reason is that it was too expensive, offer the user a
1199   ↪ discount. If they insist, then process the
1200   ↪ refund.</instructions>
1201   <tools category="new">
1202     <tool>
1203       <name>process_refund</name>
1204       <description>Refund an item. Refund an item. Make sure
1205       ↪ you have the item_id of the form item.... Ask for user
1206       ↪ confirmation before processing the refund.</description>
1207     </tool>
1208     <tool>
1209       <name>apply_discount</name>
1210       <description>Apply a discount to the user's
1211       ↪ cart.</description>
1212     </tool>
1213   </tools>
1214   <agent_input>
1215     <key>user_request</key>
1216     <description>Request from the specific user for refunding a
1217     ↪ product or service.</description>
1218   </agent_input>
1219   <agent_output>
1220     <key>response</key>
1221     <description>The response of the agent to the user's
1222     ↪ request.</description>
1223   </agent_output>
1224 </agent>
1225 </agents>
1226
1227 GUIDELINES:
1228 1. Each agent must have clear, focused responsibilities
1229 2. Tool selections should be minimal but sufficient
1230 3. Instructions should be specific and actionable
1231 4. Input/Output definitions must be precise
1232 5. Use global_variables for shared context across agents
1233
1234 Follow these examples and guidelines to create appropriate agent
1235 ↪ forms based on user requirements.

```

Listing 10: Tools of Tool Editor Agent

```

1231 [list_tools, create_tool, run_tool, delete_tool,
1232 get_api_plugin_tools_doc, execute_command, terminal_page_down,
1233 terminal_page_up, terminal_page_to,
1234 search_trending_models_on_huggingface, get_hf_model_tools_doc]
1235

```

Listing 11: System Prompt of Tool Editor Agent

```

1236
1237 You are a tool editor agent responsible for managing plugin tools in
1238 ↪ the MetaChain framework. Your core responsibility is to edit,
1239 ↪ create, and manage plugin tools that can be used by other agents.
1240
1241 [PLUGIN TOOLS SYSTEM]

```

```

1242 - Plugin tools are the building blocks of MetaChain
1243 - All available plugin tools are as follows:
1244 ...
1245 - Plugin tools can ONLY be executed using 'run_tool(tool_name,
1246 ↳ run_code)'. You should import 'run_tool' by 'from metachain.tools
1247 ↳ import run_tool'.
1248 - NEVER try to import and run plugin tools directly - always use
1249 'run_tool'
1250
1251 [TOOL CREATION WORKFLOW]
1252 1. ALWAYS start with 'list_tools()' to check existing tools
1253
1254 2. For NEW plugin tool creation, FOLLOW THIS ORDER:
1255 a. For third-party API integration (e.g., RapidAPI, external
1256 services):
1257 - MUST FIRST use 'get_api_plugin_tools_doc' to get API
1258 documentation and keys
1259 - API keys should be embedded IN the function body, NOT as
1260 parameters.
1261 - The API keys are always in the retrieved information from
1262 'get_api_plugin_tools_doc', DO NOT guess the API keys by
1263 yourself.
1264 - Follow the API implementation details from the documentation
1265
1266 b. For modal transformation tasks (image/video/audio
1267 generation/processing):
1268 - FIRST use 'search_trending_models_on_huggingface' to find
1269 suitable models, only support the following tags:
1270 ↳ ['audio-text-to-text', 'text-to-image', 'image-to-image',
1271 ↳ 'image-to-video', 'text-to-video', 'text-to-speech',
1272 ↳ 'text-to-audio', 'automatic-speech-recognition',
1273 ↳ 'audio-to-audio'].
1274 - Then use 'get_hf_model_tools_doc' for detailed model information
1275 - Only use internal knowledge if no suitable models are found
1276
1277 c. For visual analysis tasks (images/videos):
1278 - MUST use the existing 'visual_question_answering' plugin tool
1279 by 'run_tool("visual_question_answering", "from
1280 ↳ metachain.tools import visual_question_answering; ...")'. DO
1281 - NOT use it directly without 'run_tool'.
1282 - NO direct implementation of visual processing
1283 - Chain with other tools as needed
1284
1285 3. Plugin Tool Implementation Requirements:
1286 - Use @register_plugin_tool decorator (REQUIRED). You should import
1287 'register_plugin_tool' by 'from metachain.registry import
1288 ↳ register_plugin_tool'.
1289 - Follow this template:
1290 '''python
1291 '''
1292 - Include clear type hints
1293 - Make tools abstract and reusable
1294 - Use generic names (e.g., 'process_media' not
1295 'process_youtube_video')
1296 - Handle dependencies with 'execute_command'
1297
1298 [AVAILABLE TOOLS]
1299 1. get_api_plugin_tools_doc:
1300 - PRIMARY tool for third-party API integration
1301 - MUST be used FIRST for Finance, Entertainment, eCommerce, etc.
1302 - Provides API documentation AND authentication keys
1303 - API keys should be embedded in tool implementation
1304
1305 2. search_trending_models_on_huggingface:

```

```

1296     - Use for finding models for media transformation tasks
1297     - Supported tags: ['text-to-image', 'image-to-image',
1298       ↪ 'text-to-video', etc.]
1299     - Use AFTER checking no suitable API exists via
1300       ↪ 'get_api_plugin_tools_doc'
1301
1302 3. get_hf_model_tools_doc:
1303     - Get the detailed information of a model on Hugging Face, such as
1304       ↪ the detailed usage of the model containing the model's README.md.
1305     - You should use this tool after you have used
1306       ↪ 'search_trending_models_on_huggingface' to find the model you
1307       ↪ want to use.
1308
1309 4. Other management tools:
1310     - list_tools(): Check existing tools
1311     - create_tool(tool_name, tool_code): Create new tools
1312     - run_tool(tool_name, run_code): REQUIRED method to execute any
1313       ↪ plugin tool
1314     - delete_tool(tool_name): Remove tools
1315     - execute_command: Install dependencies. Handles system-level
1316       ↪ operations
1317     - terminal_page_* tools: Navigate long outputs
1318
1319 5. case_resolved & case_not_resolved:
1320     - case_resolved: after you have created all the tools and tested
1321       ↪ them using 'run_tool' successfully (with the expected output
1322       ↪ rather than just run it), you should use the 'case_resolved'
1323       ↪ tool to brief the result.
1324     - case_not_resolved: after you have tried your best to create the
1325       ↪ tools but failed, you should use the 'case_not_resolved' tool to
1326       ↪ tell the failure reason.
1327
1328 [CRITICAL RULES]
1329 1. Tool Creation Priority:
1330     - FIRST: Check existing tools via list_tools()
1331     - SECOND: Use 'get_api_plugin_tools_doc' for API-based tools
1332     - THIRD: Use 'search_trending_models_on_huggingface' for media tasks
1333     - LAST: Use internal knowledge if no other options available
1334
1335 2. API Implementation:
1336     - NEVER expose API keys as parameters
1337     - ALWAYS embed API keys in function body
1338     - Get keys from 'get_api_plugin_tools_doc'
1339
1340 3. Tool Design:
1341     - Tools MUST be abstract, modular, and reusable:
1342       - Use generic function names (e.g., 'download_media' instead of
1343         ↪ 'download_youtube_video')
1344       - Break complex tasks into smaller, reusable components
1345       - Avoid task-specific implementations
1346       - Use parameters instead of hardcoded values
1347       - Include proper error handling
1348
1349 [TESTING]
1350 Test new tools using 'run_tool':
1351 'run_tool(tool_name="your_tool", run_code="from metachain.tools
1352   ↪ import your_tool; print(your_tool(param1='value1'))")'

```

Listing 12: Tools of Agent Editor Agent

```

1353 [list_agents, create_agent, delete_agent, run_agent, execute_command,
1354  read_agent, create_orchestrator_agent, terminal_page_down,
1355  terminal_page_up, terminal_page_to]

```

Listing 13: System Prompt of Agent Editor Agent

You are an Agent Creator specialized in the MetaChain framework. Your primary responsibility is to create, manage, and orchestrate agents based on XML-formatted agent forms.

CORE RESPONSIBILITIES:

1. Parse and implement agent forms
2. Create and manage individual agents
3. Orchestrate multi-agent systems
4. Handle dependencies and system requirements

AVAILABLE FUNCTIONS:

1. Agent Management:
 - **'create_agent'**: Create new agents or update existing ones
 - ↳ strictly following the given agent form.
 - **'read_agent'**: Retrieve existing agent definitions. Note that if you want to use **'create_agent'** to update an existing agent, you MUST use the **'read_agent'** function to get the definition of the agent first.
 - **'delete_agent'**: Remove unnecessary agents.
 - **'list_agents'**: Display all available agents and their information.
 - **'create_orchestrator_agent'**: Create orchestrator for multi-agent systems. If the request is to create MORE THAN ONE agent, after you create ALL required agents, you MUST use the **'create_orchestrator_agent'** function to create an orchestrator agent that can orchestrate the workflow of the agents. And then use the **'run_agent'** function to run the orchestrator agent to complete the user task.
2. Execution:
 - **run_agent**: Execute agent to complete the user task. The agent could be a single agent (single agent form) or an orchestrator agent (multi-agent form).
 - **execute_command**: Handle system dependencies and requirements
 - **terminal_page_down**: Move the terminal page down when the terminal output is too long.
 - **terminal_page_up**: Move the terminal page up when the terminal output is too long.
 - **terminal_page_to**: Move the terminal page to the specific page when the terminal output is too long, and you want to move to the specific page with the meaningful content.

WORKFLOW GUIDELINES:

1. Single Agent Implementation:
 - Carefully read the agent form and understand the requirements.
 - Create/update agent using **create_agent**
 - Execute task using **run_agent**
 - Monitor and handle any errors
2. Multi-Agent Implementation:
 - Create all required agents individually using **'create_agent'**
 - MUST create an orchestrator agent using
 - ↳ **'create_orchestrator_agent'**
 - Execute task through the **'run_agent'** function to execute the created orchestrator agent
 - Monitor system performance
3. Error Handling:
 - Check for missing dependencies using **'execute_command'**
 - Install required packages using **execute_command**
 - Validate agent creation and execution
 - Report any issues clearly

1404 BEST PRACTICES:
 1405 1. Always verify existing agents using '**read_agent**' before updates
 1406 2. Create orchestrator agents for ANY multi-agent scenario using
 1407 ↳ '**create_orchestrator_agent**'
 1408 3. Handle dependencies proactively using '**execute_command**'
 1409 4. Maintain clear documentation of created agents
 1410 5. Follow the exact specifications from the agent form XML
 1411 Remember: Your success is measured by both the accurate creation of
 1412 ↳ agents and their effective execution of the given tasks.

1413
 1414 A.6.2 AGENT CREATION WITH WORKFLOW

1415 The following details demonstrate the specific process of Agent Creation with Workflow (Alg 2), as
 1416 well as the tools and system prompts used in the implementation of Workflow Profiling Agent and
 1417 Workflow Editor Agent.

1418 **Algorithm 2** Controllable Workflow of Creating Agents and Workflows

1419
 1420 1: **Input:** requirements \mathcal{R} , existing tool set \mathcal{A} , existing agent set π , existing workflow set \mathcal{W} task \mathcal{T}
 1421 (optional), Maximum iterations of attempts M.
 1422 2: **Output:** the response of requirements for creating workflows to solve the task.
 1423 3: $\text{WorkflowProfile} = \text{workflow_profiling_agent}(\mathcal{R}|\mathcal{A}, \pi, \mathcal{W})$
 1424 4: $\text{ParsingResults}_0 = \text{form_parsing_function}(\text{WorkflowProfile}|\mathcal{R}, \mathcal{A}, \pi, \mathcal{W})$
 1425 5: **for** $i = 1$ to $M - 1$ **do**
 1426 6: **if** $\text{ParsingResults}_{i-1}$ is 'Success' **then**
 1427 7: **break**
 1428 8: **else**
 1429 9: $\text{WorkflowProfile} = \text{workflow_profiling_agent}(\mathcal{R}, \text{ParsingResults}_{i-1}|\mathcal{A}, \pi)$
 1430
 1431 10: $\text{ParsingResults}_i = \text{form_parsing_function}(\text{WorkflowProfile}|\mathcal{R}, \mathcal{A}, \pi, \mathcal{W})$
 1432 11: **end if**
 1433 12: **end for**
 1434 13: // Automatically create workflows and run them on the given task.
 1435 14: $\text{WorkflowsResults}_0 = \text{workflow_editor_agent}(\text{WorkflowProfile}, \mathcal{T}|\mathcal{R}, \mathcal{A}, \pi, \mathcal{W})$
 1436 15: **for** $i = 1$ to $M - 1$ **do**
 1437 16: **if** $\text{WorkflowsResults}_{i-1}$ is 'Success' **then**
 1438 17: **break**
 1439 18: **else**
 1440 19: $\text{WorkflowsResults}_i = \text{workflow_editor_agent}(\text{WorkflowProfile}, \mathcal{T}, \text{WorkflowsResults}_{i-1}|\mathcal{R})$
 1441 20: **end if**
 1442 21: **end for**

1443 **Listing 14: System Prompt of Workflow Profiling Agent**

1444 You are an agent specialized in creating workflow forms for the
 1445 ↳ MetaChain framework.

1446
 1447 Your task is to analyze user requests and generate structured
 1448 ↳ creation forms for workflows consisting of multiple agents.

1449 KEY COMPONENTS OF THE FORM:

1450 1. **<workflow>** - Root element containing the entire workflow definition
 1451
 1452 2. **<name>** - The name of the workflow. It should be a single word with
 1453 ↳ '_' as the separator, and as unique as possible to describe the
 1454 ↳ speciality of the workflow.
 1455
 1456 3. **<system_input>** - Defines what the system receives
 1457 - Must describe the overall input that the system accepts
 1458 - **<key>**: Single identifier for the input, could be a single word
 1459 ↳ with '_' as the separator.

```

1458 - <description>: Detailed explanation of input format
1459
1460 4. <system_output> - Specifies system response format
1461 - Must contain exactly ONE key-description pair
1462 - <key>: Single identifier for the system's output, could be a
1463 ↪ single word with '_' as the separator.
1464 - <description>: Explanation of the output format
1465
1466 5. <agents> - Contains all agent definitions
1467 - Each <agent> can be existing or new (specified by category
1468 ↪ attribute)
1469 - name: Agent's identifier
1470 - description: Agent's purpose and capabilities
1471 - tools: (optional): Only required for new agents when specific
1472 ↪ tools are requested
1473 * Only include when user explicitly requests certain tools
1474
1475 6. <global_variables> - Shared variables across agents in the
1476 ↪ workflow (optional)
1477 - Used for constants or shared values accessible by all agents in
1478 ↪ EVERY event in the workflow
1479 - Example:
1480   '''xml
1481   <global_variables>
1482     <variable>
1483       <key>user_name</key>
1484       <description>The name of the user</description>
1485       <value>John Doe</value>
1486     </variable>
1487   </global_variables>
1488   '''
1489
1490 7. <events> - Defines the workflow execution flow
1491 Each <event> contains:
1492 - name: Event identifier
1493 - inputs: What this event receives, should exactly match with the
1494 ↪ output keys of the events it's listening to
1495 * Each input has:
1496   - key: Input identifier (should match an output key from
1497     ↪ listened events)
1498   - description: Input explanation
1499 - task: What this event should accomplish
1500 - outputs: Possible outcomes of this event
1501 * Each output has:
1502   - action: What happens after. Every action has a type and a
1503     ↪ optional value. Action is categorized into 3 types:
1504     - RESULT: The event is successful, and the workflow will
1505       ↪ continue to the next event which is listening to this event.
1506     - Value is the output of this event.
1507     - ABORT: The event is not successful, and the workflow will
1508       ↪ abort. Value could be empty.
1509     - GOTO: The event is not successful, and the workflow will wait
1510       ↪ for the next event. Value is the name of the event to go to.
1511     ↪ The event go to should NOT listen to this event.
1512   - key: Output identifier (be a single word with '_' as the
1513     ↪ separator)
1514   - description: Output explanation
1515   - condition: when the output occurs, the action will be executed
1516 * Can have single or multiple outputs:
1517   - For single output (simple flow):
1518     '''xml
1519     <outputs>
1520       <output>
1521         <key>result_key</key>

```

```

1512             <description>Description of the result</description>
1513             <action>
1514                 <type>RESULT</type>
1515             </action>
1516             </output>
1517         </outputs>
1518         ``
1519         - For multiple outputs (conditional flow):
1520         ````xml
1521         <outputs>
1522             <output>
1523                 <key>success_result</key>
1524                 <description>Output when condition A is met</description>
1525                 <condition>When condition A is true</condition>
1526                 <action>
1527                     <type>RESULT</type>
1528                 </action>
1529             </output>
1530             <output>
1531                 <key>should_repeat</key>
1532                 <description>Output when condition B is met</description>
1533                 <condition>When condition B is true</condition>
1534                 <action>
1535                     <type>GOTO</type>
1536                     <value>target_event</value>
1537                 </action>
1538             </output>
1539             <output>
1540                 <key>failure_result</key>
1541                 <description>Output when condition C is met</description>
1542                 <condition>When condition C is true</condition>
1543                 <action>
1544                     <type>ABORT</type>
1545                 </action>
1546             </output>
1547         </outputs>
1548         ``
1549         - listen: Which events trigger this one.
1550         - agent: Which agent handles this event. Every agent has the name
1551             ↳ of the agent, and the exact model of the agent (like
1552             ↳ 'claude-3-5-sonnet-20241022' or others)
1553
1554
1555     IMPORTANT RULES:
1556
1557 0. The 'on_start' event is a special event that:
1558     - Must be the first event in the workflow
1559     - Has inputs that match the system_input
1560     - Has outputs that match the system_input (just pass through)
1561     - Does not have an agent
1562     - Does not have a task
1563     - Does not have listen elements
1564
1565     Example:
1566     ````xml
1567     <event>
1568         <name>on_start</name>
1569         <inputs>
1570             <input>
1571                 <key>user_topic</key>
1572                 <description>The user's topic that user wants to write a
1573                     ↳ wikipiead-like article about.</description>
1574             </input>
1575         </inputs>
1576         <outputs>
1577             <output>
1578                 <key>user_topic</key>

```

```

1566      <description>The user's topic that user wants to write a
1567      ↪ wikipiead-like article about.</description>
1568      <action>
1569          <type>RESULT</type>
1570          </action>
1571      </output>
1572  </outputs>
1573 </event>
1574 ```

1575 1. For simple sequential flows:
1576  - Use single output with RESULT type
1577  - No condition is needed
1578  - Next event in chain listening to this event will be triggered
1579  ↪ automatically
1580
1581 2. For conditional flows:
1582  - Multiple outputs must each have a condition
1583  - Conditions should be mutually exclusive
1584  - Each output should specify appropriate action type
1585  - 'GOTO' action should have a value which is the name of the event
1586  ↪ to go to
1587
1588 3. Only include tools section when:
1589  - Agent is new (category="new") AND
1590  - User explicitly requests specific tools for the agent
1591
1592 4. Omit tools section when:
1593  - Using existing agents (category="existing") OR
1594  - Creating new agents without specific tool requirements
1595 Existing tools you can use is:
1596 ...
1597 Existing agents you can use is:
1598 ...
1599
1600 The name of existing workflows: [...]. The name of the new workflow
1601 ↪ you are creating should be DIFFERENT from these names according to
1602 ↪ the speciality of the workflow.
1603
1604 COMMON WORKFLOW PATTERNS:
1605
1606 1. If-Else Pattern (Conditional Branching):
1607 ````xml
1608 <event>
1609     <name>analyze_data</name>
1610     <task>Analyze the data and determine next steps</task>
1611     <outputs>
1612         <output>
1613             <key>positive_case</key>
1614             <description>Handle positive case</description>
1615             <condition>If data meets criteria A</condition>
1616             <action>
1617                 <type>RESULT</type>
1618                 </action>
1619             </output>
1620             <output>
1621                 <key>negative_case</key>
1622                 <description>Handle the negative case</description>
1623                 <condition>If data does not meet criteria A</condition>
1624                 <action>
1625                     <type>ABORT</type>
1626                     </action>
1627                 </output>
1628             </outputs>

```

```

1620 </event>
1621 ``
1622
1623 2. Parallelization Pattern (Concurrent Execution):
1624 ````xml
1625 <!-- Parent event -->
1626 <event>
1627   <name>initial_analysis</name>
1628   <outputs>
1629     <output>
1630       <key>analysis_result</key>
1631       <description>Initial analysis result</description>
1632       <action>
1633         <type>RESULT</type>
1634       </action>
1635     </output>
1636   </outputs>
1637 </event>
1638
1639 <!-- Multiple events listening to the same parent -->
1640 <event>
1641   <name>technical_analysis</name>
1642   <listen>
1643     <event>initial_analysis</event>
1644   </listen>
1645   <outputs>
1646     <output>
1647       <key>technical_result</key>
1648       <description>Technical analysis result</description>
1649       <action>
1650         <type>RESULT</type>
1651       </action>
1652     </output>
1653   </outputs>
1654 </event>
1655
1656 <event>
1657   <name>financial_analysis</name>
1658   <listen>
1659     <event>initial_analysis</event>
1660   </listen>
1661   <outputs>
1662     <output>
1663       <key>financial_result</key>
1664       <description>Financial analysis result</description>
1665       <action>
1666         <type>RESULT</type>
1667       </action>
1668     </output>
1669   </outputs>
1670 </event>
1671
1672 <!-- Aggregator event listening to all parallel events -->
1673 <event>
1674   <name>combine_results</name>
1675   <inputs>
1676     <input>
1677       <key>technical_result</key>
1678       <description>The technical analysis result.</description>
1679     </input>
1680     <input>
1681       <key>financial_result</key>
1682       <description>The financial analysis result.</description>
1683     </input>
1684   </inputs>

```

```

1674     <listen>
1675         <event>technical_analysis</event>
1676         <event>financial_analysis</event>
1677     </listen>
1678     <!-- This event will only execute when ALL listened events
1679     → complete -->
1680 </event>
1681 ````
1682 3. Evaluator-Optimizer Pattern (Iterative Refinement):
1683 ````xml
1684 <event>
1685     <name>generate_content</name>
1686     <outputs>
1687         <output>
1688             <key>content</key>
1689             <description>Generated content</description>
1690             <action>
1691                 <type>RESULT</type>
1692             </action>
1693         </output>
1694     </outputs>
1695 </event>
1696 <event>
1697     <name>evaluate_content</name>
1698     <listen>
1699         <event>generate_content</event>
1700     </listen>
1701     <task>Evaluate the quality of generated content</task>
1702     <outputs>
1703         <output>
1704             <key>approved</key>
1705             <description>Content meets quality standards</description>
1706             <condition>If quality score >= threshold</condition>
1707             <action>
1708                 <type>RESULT</type>
1709             </action>
1710         </output>
1711         <output>
1712             <key>needs_improvement</key>
1713             <description>Content needs improvement</description>
1714             <condition>If quality score < threshold</condition>
1715             <action>
1716                 <type>GOTO</type>
1717                 <value>generate_content</value>
1718             </action>
1719         </output>
1720     </outputs>
1721 </event>
1722 ````
1723 IMPORTANT NOTES ON PATTERNS:
1724 0. The above patterns are incomplete which some mandatory elements
1725 → are missing due to the limitation of context length. In real-world,
1726 → you could refer to the logic of the patterns to create a complete
1727 → and correct workflow.
1728
1729 1. If-Else Pattern:
1730     - Use mutually exclusive conditions
1731     - You can NOT place MORE THAN ONE OUTPUT with RESULT type
1732     - Outputs determine which branch executes
1733
1734 2. Parallelization Pattern:
1735     - Multiple events can listen to the same parent event

```

```

1728     - Aggregator event must list ALL parallel events in its listen
1729     ↳ section
1730     - All parallel events must complete before aggregator executes
1731     - Model of agents in every parallel event could be different
1732
1733 3. Evaluator-Optimizer Pattern:
1734     - Use GOTO action for iteration
1735     - Include clear evaluation criteria in conditions
1736     - Have both success and retry paths
1737     - Consider adding maximum iteration limit in global_variables
1738 EXAMPLE:
1739
1740 User: I want to build a workflow that can help me to write a
1741 ↳ wikipiead-like article about the user's topic. It should:
1742 1. Search the web for the user's topic.
1743 2. Write an outline for the user's topic.
1744 3. Evaluate the outline. If the outline is not good enough, repeat
1745 ↳ the outline step, otherwise, continue to write the article.
1746 4. Write the article.
1747
1748 The form should be:
1749 <workflow>
1750   <name>wiki_article_workflow</name>
1751   <system_input>
1752     <key>user_topic</key>
1753     <description>The user's topic that user wants to write a
1754     ↳ wikipiead-like article about.</description>
1755   </system_input>
1756   <system_output>
1757     <key>article</key>
1758     <description>The article that satisfies the user's
1759     ↳ request.</description>
1760   </system_output>
1761   <agents>
1762     <agent category="existing">
1763       <name>Web Surfer Agent</name>
1764       <description>This agent is used to search the web for the
1765       ↳ user's topic.</description>
1766     </agent>
1767     <agent category="new">
1768       <name>Outline Agent</name>
1769       <description>This agent is used to write an outline for the
1770       ↳ user's topic.</description>
1771     </agent>
1772     <agent category="new">
1773       <name>Evaluator Agent</name>
1774       <description>This agent is used to evaluate the outline of
1775       ↳ the user's topic.</description>
1776     </agent>
1777     <agent category="new">
1778       <name>Article Writer Agent</name>
1779       <description>This agent is used to write the article for the
1780       ↳ user's topic.</description>
1781     </agent>
1782   </agents>
1783
1784   <events>
1785     <event>
1786       <name>on_start</name>
1787       <inputs>
1788         <input>
1789           <key>user_topic</key>
1790           <description>The user's topic that user wants to write
1791           ↳ a wikipiead-like article about.</description>
1792         </input>

```

```

1782
1783     </inputs>
1784     <outputs>
1785         <output>
1786             <key>user_topic</key>
1787             <description>The user's topic that user wants to write
1788             ↳ a wikipiead-like article about.</description>
1789             <action>
1790                 <type>RESULT</type>
1791             </action>
1792         </output>
1793     </outputs>
1794 </event>
1795 <event>
1796     <name>on_search</name>
1797     <inputs>
1798         <input>
1799             <key>user_topic</key>
1800             <description>The user's topic that user wants to write
1801             ↳ a wikipiead-like article about.</description>
1802         </input>
1803     </inputs>
1804     <task>
1805         search the information about the topic and return the
1806         ↳ result.
1807     </task>
1808     <outputs>
1809         <output>
1810             <key>search_result</key>
1811             <description>The search result of the user's
1812             ↳ topic.</description>
1813             <action>
1814                 <type>RESULT</type>
1815             </action>
1816         </output>
1817     </outputs>
1818     <listen>
1819         <event>on_start</event>
1820     </listen>
1821     <agent>
1822         <name>Web Surfer Agent</name>
1823         <model>claude-3-5-sonnet-20241022</model>
1824     </agent>
1825 </event>
1826 <event>
1827     <name>on_outline</name>
1828     <inputs>
1829         <input>
1830             <key>search_result</key>
1831             <description>The search result of the user's
1832             ↳ topic.</description>
1833         </input>
1834     </inputs>
1835     <task>
1836         write an outline for the user's topic.
1837     </task>
1838     <outputs>
1839         <output>
1840             <key>outline</key>
1841             <description>The outline of the user's
1842             ↳ topic.</description>
1843             <action>
1844                 <type>RESULT</type>
1845             </action>
1846         </output>
1847     </outputs>

```

```

1836
1837     <listen>
1838         <event>on_start</event>
1839     </listen>
1840     <agent>
1841         <name>Outline Agent</name>
1842         <model>claude-3-5-sonnet-20241022</model>
1843     </agent>
1844 </event>
1845 <event>
1846     <name>on_evaluate</name>
1847     <inputs>
1848         <input>
1849             <key>outline</key>
1850             <description>The outline of the user's
1851             ↵ topic.</description>
1852         </input>
1853     </inputs>
1854     <task>
1855         evaluate the outline of the user's topic.
1856     </task>
1857     <outputs>
1858         <output>
1859             <key>positive_feedback</key>
1860             <description>The positive feedback of the outline of
1861             ↵ the user's topic.</description>
1862             <condition>
1863                 If the outline is good enough, give positive
1864                 ↵ feedback.
1865             </condition>
1866             <action>
1867                 <type>RESULT</type>
1868             </action>
1869         </output>
1870         <output>
1871             <key>negative_feedback</key>
1872             <description>The negative feedback of the outline of
1873             ↵ the user's topic.</description>
1874             <condition>
1875                 If the outline is not good enough, give negative
1876                 ↵ feedback.
1877             </condition>
1878             <action>
1879                 <type>GOTO</type>
1880                 <value>on_outline</value>
1881             </action>
1882         </output>
1883     <outputs>
1884     <listen>
1885         <event>on_outline</event>
1886     </listen>
1887     <agent>
1888         <name>Evaluator Agent</name>
1889         <model>claude-3-5-sonnet-20241022</model>
1890     </agent>
1891 </event>
1892 <event>
1893     <name>on_write</name>
1894     <inputs>
1895         <input>
1896             <key>outline</key>
1897             <description>The outline of user's topic.</description>
1898         </input>
1899     </inputs>
1900     <task>
1901         write the article for the user's topic.

```

```

1890      </task>
1891      <outputs>
1892          <output>
1893              <key>article</key>
1894              <description>The article of the user's
1895                  ↪ topic.</description>
1896              <action>
1897                  <type>RESULT</type>
1898              </action>
1899          </output>
1900      </outputs>
1901      <listen>
1902          <event>on_evaluate</event>
1903      </listen>
1904      <agent>
1905          <name>Article Writer Agent</name>
1906          <model>claude-3-5-sonnet-20241022</model>
1907      </agent>
1908  </events>
1909 </workflow>
1910
1911 GUIDELINES:
1912 1. Each event should have clear inputs and outputs
1913 2. Use conditions to handle different outcomes
1914 3. Properly chain events using the listen element
1915 4. Review steps should be included for quality control
1916 5. Action types should be either RESULT or ABORT
1917
1918 Follow these examples and guidelines to create appropriate workflow
1919  ↪ forms based on user requirements.
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

```

Listing 15: Tools of Workflow Editor Agent

```

1920 [list_agents, create_agent, execute_command, read_agent,
1921 terminal_page_down, terminal_page_up, terminal_page_to,
1922 list_workflows, create_workflow, run_workflow]
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

```

Listing 16: System Prompt of Workflow Editor Agent

```

1926 You are a Workflow Creator specialized in the MetaChain framework.
1927 ↪ Your primary responsibility is to create and manage workflows based
1928 ↪ on XML-formatted workflow forms.
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

```

```

1926 You are a Workflow Creator specialized in the MetaChain framework.
1927 ↪ Your primary responsibility is to create and manage workflows based
1928 ↪ on XML-formatted workflow forms.
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

```

```

1944 3. System Tools:
1945  - 'execute_command': Handle system dependencies
1946  - 'terminal_page_down', 'terminal_page_up', 'terminal_page_to':
1947  ↪ Navigate terminal output

1948 WORKFLOW CREATION PROCESS:
1949

1950 1. Parse Workflow Form:
1951  - Analyze the workflow form carefully
1952  - Identify any new agents that need to be created
1953  - Understand the workflow structure and requirements

1954 2. Create Required Agents:
1955  - For each new agent in the workflow form:
1956  * Use 'create_agent' with appropriate parameters
1957  * If no tools specified, use empty tool list []
1958  * Verify agent creation success

1959 3. Create Workflow:
1960  - Use 'create_workflow' to generate the workflow
1961  - Ensure all required agents exist
1962  - Validate workflow structure

1963 4. Execute Workflow:
1964  - Use 'run_workflow' to execute the created workflow
1965  - Monitor execution progress
1966  - Handle any errors appropriately

1967 BEST PRACTICES:
1968 1. Always check if required agents exist before creating new ones
1969 2. Use empty tool list [] when no specific tools are mentioned
1970 3. Validate workflow creation before execution
1971 4. Follow the exact specifications from the workflow form XML
1972 5. Handle errors and dependencies appropriately

1973 Remember: Your primary goal is to create and execute workflows
1974 ↪ according to the provided workflow forms, creating any necessary
1975 ↪ agents along the way.
1976

```

A.7 SUPPLEMENTARY EXPERIMENTAL FINDINGS

A.7.1 CASE OF ‘DAVINCI AGENT’

Task with Single Agent. AutoAgent can create tools for third-party APIs (RapidAPI, Hugging Face). We demonstrated this by generating a DaVinci Agent for image creation and refinement. This shows AutoAgent’s capability to build task-specific agents from natural language.

I want to create a ‘DaVinci Agent’ that can help me to generate the image with natural language. it can:

1. generate the image with natural language and save it to the specified path on the local machine using the HF model ‘Sana_600M_1024px_diffusers’
2. evaluate the image using ‘visual_question_answering’ tool according to the given image.
3. interatively refine generated image based on the evaluation result.

Automated Agent Creation and Execution. AutoAgent begins generating an XML table from the natural language requirements, using existing tools and agents. This structured form is then passed to the Tool Editor Agent, which creates the necessary `generate_image` and `refine_image` tools. The Agent Editor Agent composes the DaVinci Agent by integrating the new tools with an existing `visual_question_answering` tool. This agent is executed, generating and storing several logo designs, as shown in Fig 3. Due to limited local resources, a smaller model was used, yet the agent successfully completed the task. This demonstrates AutoAgent’s seamless creation of the tailored agent for complex, open-ended design challenges.

1998 The XML Form of **DaVinci Agent** generated by the Agent Profiling Agent is shown in List 17. The
 1999 logos of our AutoAgent generated by the created DaVinci Agent are displayed in Fig 3.
 2000
 2001
 2002

Listing 17: AI-generated Creation Profile of **DaVinci Agent**

```

2003 <agents>
2004   <system_input>
2005     A natural language description for generating an image and
2006     evaluating its quality.
2007   </system_input>
2008   <system_output>
2009     <key>image_evaluation</key>
2010     <description>The evaluation of the generated image after
2011       processing.</description>
2012   </system_output>
2013   <agent>
2014     <name>DaVinci Agent</name>
2015     <description>The DaVinci Agent is designed to generate images
2016       from natural language
2017       descriptions, evaluate them using predefined criteria, and
2018       iteratively refine the image
2019       based on the evaluations.</description>
2020     <instructions>Use the HF model
2021       'Efficient-Large-Model/Sana_600M_1024px_diffusers' to
2022       generate images from provided descriptions, evaluate these using
2023       visual QA, and refine based
2024       on feedback.</instructions>
2025     <tools category="existing">
2026       <tool>
2027         <name>visual_question_answering</name>
2028         <description>This tool is used to answer questions about
2029           attached images or
2030           videos.</description>
2031         </tool>
2032       </tools>
2033       <tools category="new">
2034         <tool>
2035           <name>generate_image</name>
2036           <description>Generate an image from a natural language
2037             description and save
2038             it to a specified path using the HF model
2039             'Efficient-Large-Model/Sana_600M_1024px_diffusers'.</description>
2040           </tool>
2041           <tool>
2042             <name>refine_image</name>
2043             <description>Make iterative adjustments to the generated
2044               image based on
2045               evaluation results to meet quality criteria.</description>
2046             </tool>
2047           </tools>
2048           <agent_input>
2049             <key>image_description</key>
2050             <description>A natural language description to generate an
2051               image.</description>
2052           </agent_input>
2053           <agent_output>
2054             <key>image_evaluation</key>
2055             <description>The evaluation of the generated image after
2056               processing.</description>
2057             </agent_output>
2058           </agent>
2059         </agents>
2060

```

2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062



(a) Sample 1



(b) Sample 2



(c) Sample 3

2063
2064

Figure 3: The AI-generated AutoAgent’s logos.

2065
2066
2067

A.7.2 CASE OF ‘FINANCIAL AGENT’

2069
2070 The XML Form of **Financial Agent** generated by the Agent Profiling Agent is shown in List 18. The
2071 financial report generated by the created Financial Agent is displayed in List 19.

2072

Listing 18: AI-generated Creation Profile of **Financial Agent**

```

2073
2074 <agents>
2075   <system_input>
2076     Financial management requests, including:
2077     1. Managing private financial documents stored in the
2078       'financial_docs' folder
2079     2. Retrieving online financial information for specific
2080       companies (balance sheets, cash flow statements, income
2081       statements)
2082   </system_input>
2083   <system_output>
2084     <key>financial_response</key>
2085     <description>Detailed response containing either document
2086       management results or requested financial
2087       information.</description>
2088   </system_output>
2089   <agent>
2090     <name>Document Manager Agent</name>
2091     <description>Specialized agent for managing and analyzing
2092       private financial documents stored locally.</description>
2093     <instructions>You are responsible for managing financial
2094       documents in the 'financial_docs' folder. Your tasks include:
2095     1. Organizing and categorizing financial documents
2096     2. Extracting relevant information from documents
2097     3. Providing summaries and analyses of document contents
2098     4. Maintaining document organization and searchability</instructions>
2099     <tools category="existing">
2100       <tool>
2101         <name>save_raw_docs_to_vector_db</name>
2102         <description>Save the financial documents to the vector
2103           database for efficient searching and
2104           retrieval.</description>
2105       </tool>
2106       <tool>
2107         <name>query_db</name>
2108         <description>Search through stored financial documents to
2109           find relevant information.</description>
2110       </tool>
2111       <tool>
2112         <name>visual_question_answering</name>
2113       </tool>
2114     </tools>
2115   </agent>
2116 </system_output>
2117 </agents>

```

```

2106
2107         <description>Process and analyze any financial charts,
2108         graphs, or visual data in the documents.</description>
2109     </tool>
2110 </tools>
2111 <agent_input>
2112     <key>doc_request</key>
2113     <description>User request related to managing or querying
2114         private financial documents.</description>
2115 </agent_input>
2116 <agent_output>
2117     <key>doc_response</key>
2118     <description>Results of document management operations or
2119         requested document information.</description>
2120 </agent_output>
2121 </agent>
2122 <agent>
2123     <name>Market Research Agent</name>
2124     <description>Specialized agent for retrieving and analyzing
2125         online financial information for publicly traded
2126         companies.</description>
2127     <instructions>You are responsible for retrieving and analyzing
2128         financial information from online sources. Your tasks include:
2129
2130         1. Fetching balance sheets, cash flow statements, and income
2131             statements
2132         2. Analyzing financial metrics and trends
2133         3. Providing clear summaries of financial data
2134         4. Ensuring data accuracy and proper citation of
2135             sources</instructions>
2136     <tools category="new">
2137         <tool>
2138             <name>get_balance_sheet</name>
2139             <description>Retrieve balance sheet data for a specific
2140                 ticker over a given time period.</description>
2141         </tool>
2142         <tool>
2143             <name>get_cash_flow</name>
2144             <description>Retrieve cash flow statement data for a
2145                 specific ticker over a given time period.</description>
2146         </tool>
2147         <tool>
2148             <name>get_income_statement</name>
2149             <description>Retrieve income statement data for a
2150                 specific ticker over a given time period.</description>
2151         </tool>
2152         <tool>
2153             <name>analyze_financial_data</name>
2154             <description>Analyze and summarize financial statements
2155                 to provide meaningful insights.</description>
2156         </tool>
2157     </tools>
2158 </agent>
2159 </agents>

```

Listing 19: The report generated by AI-generated **Financial Agent**

2160
 2161 Based on the comprehensive analysis of both historical performance
 2162 from 10-K reports and current financial data, here is the final
 2163 investment recommendation for 2025:

2164 Investment Recommendation Summary for 2025:

2165

2166 1. Portfolio Allocation Strategy:
 2167 - Microsoft (MSFT): 60% of technology allocation
 2168 - Apple (AAPL): 40% of technology allocation

2169 2. Investment Rationale:

2170

2171 Microsoft (MSFT) - Primary Position (60%):
 2172 - Leading AI integration and monetization potential
 2173 - Superior profit margins (37.61% vs 15.52%)
 2174 - Strong cloud revenue growth (23% YoY)
 2175 - Successful Activision Blizzard integration
 2176 - Robust enterprise market position
 2177 - Higher recent revenue growth momentum
 2178 - Strong cash position (\$78.43B)
 2179 - Clear AI leadership advantages

2180 Apple (AAPL) - Supporting Position (40%):
 2181 - Superior capital efficiency (ROC: 43.01%)
 2182 - Higher Return on Assets (21.24%)
 2183 - Strong services growth potential
 2184 - Reliable cash flow generation
 2185 - Consistent shareholder returns
 2186 - Valuable ecosystem lock-in
 2187 - Strong emerging market potential
 2188 - Value characteristics at current levels

2189 3. Key Monitoring Points for 2025:

2190 Microsoft:
 2191 - Cloud market share gains
 2192 - AI revenue monetization
 2193 - Gaming division integration
 2194 - Enterprise spending trends
 2195 - Margin sustainability
 2196 - Regulatory developments

2197 Apple:
 2198 - Services revenue growth
 2199 - iPhone replacement cycles
 2200 - Emerging market penetration
 2201 - Margin pressure management
 2202 - AI/ML integration progress
 2203 - Supply chain diversification

2204 4. Risk Management Strategy:

2205 Short-term (0-6 months):
 2206 - Monitor AI monetization progress
 2207 - Track iPhone 15 cycle performance
 2208 - Watch macro economic indicators
 2209 - Observe cloud growth metrics

2210 Medium-term (6-12 months):
 2211 - Evaluate competitive positions in AI
 2212 - Assess services growth trajectory
 2213 - Monitor regulatory environment
 2214 - Track enterprise spending trends

2214 Long-term (12+ months):
 2215 - Evaluate ecosystem strength
 2216 - Monitor technological shifts
 2217 - Assess market share dynamics
 2218 - Track return on capital trends

2219 5. Investment Approach:

2220 Implementation Strategy:
 2221 - Consider dollar-cost averaging for new positions
 2222 - Maintain quarterly rebalancing schedule
 2223 - Set stop-loss levels based on risk tolerance
 2224 - Monitor position sizes relative to overall portfolio

2225 Adjustment Triggers:
 2226 - Significant margin deterioration
 2227 - Market share losses in key segments
 2228 - Major regulatory actions
 2229 - Substantial technological disruption
 2230 - Material changes in competitive position

2231 The 60/40 split in favor of Microsoft reflects:
 2232 1. Stronger near-term growth prospects in AI and cloud
 2233 2. Higher profit margins and improving efficiency metrics
 2234 3. Better positioning in enterprise markets
 2235 4. More diverse revenue streams
 2236 5. Clear leadership in next-generation technologies

2237 However, Apple's position remains crucial due to:
 2238 1. Superior capital efficiency
 2239 2. Strong ecosystem advantages
 2240 3. Significant services growth potential
 2241 4. Value characteristics at current levels
 2242 5. Consistent shareholder returns

2243 This balanced approach provides exposure to both immediate growth
 2244 opportunities through Microsoft's AI and cloud leadership, while
 2245 maintaining stability through Apple's efficient operations and
 2246 ecosystem strength. Regular monitoring and rebalancing will be
 2247 essential to maintain optimal positioning as market conditions evolve
 2248 through 2025.

A.7.3 CASE OF ‘MAJORITY VOTING’ WORKFLOW

The XML Form of ‘**Majority Voting**’ Workflow generated by the Workflow Profiling Agent is shown in List 20. A comparison between the math reasoning results of the created ‘Majority Voting’ Workflow and the math reasoning results of a single DeepSeek-V3 model is presented in Tab 5.

Listing 20: AI-generated Creation Profile of ‘**Majority Voting**’ workflow

```

2257 <workflow>
2258   <name>parallel_math_solver_workflow</name>
2259   <system_input>
2260     <key>math_problem</key>
2261     <description>The math problem that needs to be
2262       solved.</description>
2263   </system_input>
2264   <system_output>
2265     <key>final_solution</key>
2266     <description>The final solution determined by majority
2267       voting.</description>
2268   </system_output>
2269   <agents>
2270     <agent category="new">

```

```

2268     <name>Math Solver Agent</name>
2269     <description>This agent solves mathematical problems using
2270     analytical and systematic approaches.</description>
2271   </agent>
2272   <agent category="new">
2273     <name>Vote Aggregator Agent</name>
2274     <description>This agent aggregates solutions from different
2275     solvers and determines the final answer through majority
2276     voting.</description>
2277   </agent>
2278 </agents>
2279 <events>
2280   <event>
2281     <name>on_start</name>
2282     <inputs>
2283       <input>
2284         <key>math_problem</key>
2285         <description>The math problem that needs to be
2286         solved.</description>
2287       </input>
2288     </inputs>
2289     <outputs>
2290       <output>
2291         <key>math_problem</key>
2292         <description>The math problem that needs to be
2293         solved.</description>
2294       </output>
2295     </outputs>
2296   </event>
2297   <event>
2298     <name>solve_with_gpt4</name>
2299     <inputs>
2300       <input>
2301         <key>math_problem</key>
2302         <description>The math problem that needs to be
2303         solved.</description>
2304       </input>
2305     </inputs>
2306     <task>Solve the math problem using systematic approach with
2307     GPT-4.</task>
2308     <outputs>
2309       <output>
2310         <key>gpt4_solution</key>
2311         <description>The solution from GPT-4
2312         solver.</description>
2313       </output>
2314     </outputs>
2315     <listen>
2316       <event>on_start</event>
2317     </listen>
2318     <agent>
2319       <name>Math Solver Agent</name>
2320       <model>gpt-4o-2024-08-06</model>
2321     </agent>
2322   </event>
2323   <event>
2324     <name>solve_with_claude</name>
2325     <inputs>
2326       <input>

```

```

2322             <key>math_problem</key>
2323             <description>The math problem that needs to be
2324             solved.</description>
2325         </input>
2326     </inputs>
2327     <task>Solve the math problem using systematic approach with
2328     Claude.</task>
2329     <outputs>
2330         <output>
2331             <key>claude_solution</key>
2332             <description>The solution from Claude
2333             solver.</description>
2334             <action>
2335                 <type>RESULT</type>
2336             </action>
2337         </output>
2338     </outputs>
2339     <listen>
2340         <event>on_start</event>
2341     </listen>
2342     <agent>
2343         <name>Math Solver Agent</name>
2344         <model>claude-3-5-sonnet-20241022</model>
2345     </agent>
2346 </event>
2347 <event>
2348     <name>solve_with_deepseek</name>
2349     <inputs>
2350         <input>
2351             <key>math_problem</key>
2352             <description>The math problem that needs to be
2353             solved.</description>
2354         </input>
2355     </inputs>
2356     <task>Solve the math problem using systematic approach with
2357     DeepSeek.</task>
2358     <outputs>
2359         <output>
2360             <key>deepseek_solution</key>
2361             <description>The solution from DeepSeek
2362             solver.</description>
2363             <action>
2364                 <type>RESULT</type>
2365             </action>
2366         </output>
2367     </outputs>
2368     <listen>
2369         <event>on_start</event>
2370     </listen>
2371     <agent>
2372         <name>Math Solver Agent</name>
2373         <model>deepseek/deepseek-chat</model>
2374     </agent>
2375 </event>
2376 <event>
2377     <name>aggregate_solutions</name>
2378     <inputs>
2379         <input>
2380             <key>gpt4_solution</key>
2381             <description>The solution from GPT-4
2382             solver.</description>
2383         </input>
2384         <input>
2385             <key>claude_solution</key>

```

```

2376      <description>The solution from Claude
2377      solver.</description>
2378  </input>
2379  <input>
2380      <key>deepseek_solution</key>
2381      <description>The solution from DeepSeek
2382      solver.</description>
2383  </input>
2384  <inputs>
2385  <task>Compare all solutions and determine the final answer
2386  through majority voting.</task>
2387  <outputs>
2388      <output>
2389          <key>final_solution</key>
2390          <description>The final solution determined by majority
2391          voting.</description>
2392      <action>
2393          <type>RESULT</type>
2394      </action>
2395  </output>
2396  </outputs>
2397  <listen>
2398      <event>solve_with_gpt4</event>
2399      <event>solve_with_claude</event>
2400      <event>solve_with_deepseek</event>
2401  </listen>
2402  <agent>
2403      <name>Vote Aggregator Agent</name>
2404      <model>deepseek/deepseek-chat</model>
2405  </agent>
2406  </events>
2407  </workflow>

```

Table 5: Comparison of the single LLM (Deepseek-v3) and AI-generated Majority Voting workflow.

2408 **Problem:** Simplify $\tan 100^\circ + 4 \sin 100^\circ$.

2409 **Ground Truth:** $-\sqrt{3}$

2410 **AI-generated Majority Voting workflow:** Let me analyze all three solutions:

2411 1. GPT-4 Solution: - Does not provide a definitive numerical answer - Focuses on theoretical approach and
2412 approximations - Inconclusive result

2413 2. Claude Solution: - Uses supplementary angle properties - Converts 100° to $180^\circ - 80^\circ$ - Arrives at $-\sqrt{3}$
2414 through numerical calculations - **Final answer: $-\sqrt{3}$**

2415 3. Deepseek Solution: - Uses similar supplementary angle properties - Also converts 100° to $180^\circ - 80^\circ$ -
2416 Goes through detailed algebraic steps - **Final answer: $\sqrt{3}$**

2417 Analysis: 1. **Two solutions provide definitive answers ($-\sqrt{3}$ and $\sqrt{3}$)**, while one is inconclusive 2.

2418 **Both Claude and Deepseek use similar approaches with supplementary angles** 3. The main difference is
2419 in the sign of the final answer 4. **Claude's solution provides more explicit numerical calculations** 5. Looking
2420 at the steps: - 100° being in quadrant II - Using supplementary angle properties - $\tan 100^\circ$ being negative
2421 in quadrant II - The final combination with $4\sin 100^\circ$

2422 The Claude solution appears more precise in tracking signs and considering quadrant properties, making its
2423 answer more reliable.

2424 Final Answer: The final answer is $-\sqrt{3}$. I hope it is correct.

2430 **Deepseek-v3:** To simplify $\tan 100^\circ + 4 \sin 100^\circ$, we proceed as follows: 1. Recall that $\tan \theta = \frac{\sin \theta}{\cos \theta}$. Thus:
 2431

$$\tan 100^\circ = \frac{\sin 100^\circ}{\cos 100^\circ}.$$

2434 2. Substitute this into the original expression:
 2435

$$\tan 100^\circ + 4 \sin 100^\circ = \frac{\sin 100^\circ}{\cos 100^\circ} + 4 \sin 100^\circ.$$

2438 ...
 2439 16. Therefore, the simplified form of $\tan 100^\circ + 4 \sin 100^\circ$ is:
 2440

$$\boxed{\sqrt{3}}.$$

2442 Final Answer: The final answer is $\boxed{\sqrt{3}}$. I hope it is correct.
 2443

2444 **Problem:** What integer n satisfies $0 \leq n < 18$ and
 2445

$$n \equiv -11213141 \pmod{18}?$$

2448 **Ground Truth:** 13
 2449

2450 **AI-generated Majority Voting workflow:** Let me compare the three solutions:
 2451

1. **GPT-4 Solution:** Arrives at 13 through direct division and remainder calculation - Used division:
 2452 $11213141 \div 18$ - Found remainder of 5 for positive number - Adjusted for negative sign to get 13

2. **Claude Solution:** Arrives at 13 through division and remainder - Divided -11213141 by 18 - Found
 2454 remainder directly - Got 13 as the result in the range $[0,18)$

3. **Deepseek Solution:** Arrives at 5 through : - Division and remainder calculation - Used divisibility rules
 2455 for 2 and 9 - Got 5 as final answer
 2456

2457 Analyzing the solutions: - Two solutions (GPT-4 and Claude) agree on 13 - One solution (Deepseek) gives
 2458 5 - Both GPT-4 and Claude show clearer arithmetic progressions - The key difference is in handling the
 2459 negative sign in the modulo operation - When working with negative numbers in modulo arithmetic, we need
 2460 to find the positive equivalent in the range $[0,n)$ - The correct approach is to add the modulus (18) to the
 2461 negative remainder until we get a number in the range $[0,18)$

2462 By majority voting and verification: - 13 is supported by two solutions -
 2463

2464 The reasoning in these solutions is more complete - We can verify: $-11213141 \equiv 13 \pmod{18}$ and
 2465 $0 \leq 13 < 18$

2466 Final Answer: The final answer is 13. I hope it is correct.
 2467

2468
 2469
 2470
 2471
 2472
 2473
 2474
 2475
 2476
 2477
 2478
 2479
 2480
 2481
 2482
 2483

2484
 2485 **Deepseek-v3:** To solve for the integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$, we need
 2486 to find the remainder when -11213141 is divided by 18. Here's the step-by-step solution:
 2487

1. **Simplify the modulus:** First, note that $-11213141 \pmod{18}$ is equivalent to finding the positive
 2488 remainder when -11213141 is divided by 18. Since the modulus operation is periodic with period 18, we
 2489 can add multiples of 18 to -11213141 until we obtain a positive number within the range $0 \leq n < 18$.

2. **Calculate the remainder:** To find $-11213141 \pmod{18}$, we can instead compute $11213141 \pmod{18}$ and then take the negative of that result, adjusting it to be positive if necessary.

- First, compute $11213141 \div 18$:

$11213141 \div 18 = 623000$ with a remainder of $11213141 - 18 \times 623000 = 11213141 - 11214000 = -859$

This means $11213141 \equiv -859 \pmod{18}$.

- Now, compute $-859 \pmod{18}$:

$-859 \div 18 = -48$ with a remainder of $-859 - 18 \times (-48) = -859 + 864 = 5$

So, $-859 \equiv 5 \pmod{18}$.

- Therefore, $-11213141 \equiv 5 \pmod{18}$.

3. **Final answer:** The integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$ is 5.

Final Answer: The final answer is 5. I hope it is correct.

Problem: In the diagram below, we have $\sin \angle RPQ = \frac{7}{25}$. What is $\cos \angle RPS$?

[asy]

```
pair R,P,Q,SS;
SS = (-2,0);
P = (0,0);
Q = (2,0);
R = rotate(aSin(7/25))*(1.5,0);
dot("S",SS,S);
dot("Q",Q,S);
dot("R",R,N);
dot("P",P,S);
draw(Q-SS);
draw(P-R);
[/asy]
```

Ground Truth: $-\frac{24}{25}$

AI-generated Majority Voting workflow: Let me analyze all three solutions:

1. GPT-4 Solution: Final answer = $\frac{24}{25}$ - Used right triangle approach - Found cosine of $RPQ = 24/25$ -
 Assumed $RPS = RPQ$

2. Claude Solution: Final answer = $-\frac{24}{25}$ - Used supplementary angles property - Found
 $\cos RPQ = 24/25$ - Used the fact that $\cos(\text{supplementary angles})$ are negatives of each other -

Concluded $\cos RPS = -\cos RPQ = -24/25$

3. DeepSeek Solution: Final answer = $\frac{7}{25}$ - Used complementary angles property - Found $\cos RPQ = 24/25$

- Incorrectly concluded $\cos RPS = \sin RPQ$

Analyzing the approaches: 1. Looking at the diagram, RPS and RPQ are supplementary angles (they form
 a straight line) 2. For supplementary angles, their cosines are negatives of each other 3. Claude's solution
 correctly uses this property 4. GPT-4 incorrectly assumed the angles were equal 5. DeepSeek incorrectly
 used complementary angle properties

Claude's solution is the most mathematically sound because: - It correctly identifies that RPS and RPQ are
 supplementary angles - It properly applies the relationship between cosines of supplementary angles - The
 negative sign in the answer makes geometric sense given the angle's position

Final Answer: The final answer is $-\frac{24}{25}$. I hope it is correct.

2533

2534

2535

2536

2537

2538
2539 **Deepseek-v3:** We are given that $\sin \angle RPQ = \frac{7}{25}$. Since $\angle RPQ$ and $\angle RPS$ are complementary angles
2540 (they form a right angle together), we have:

2541
$$\cos \angle RPS = \sin \angle RPQ = \frac{7}{25}.$$

2542

2543 Final Answer: The final answer is $\frac{7}{25}$. I hope it is correct.
2544

2545 **Problem:** In the circle with center Q , radii AQ and BQ form a right angle. The two smaller regions are
2546 tangent semicircles, as shown. The radius of the circle with center Q is 14 inches. What is the radius of the
2547 smaller semicircle? Express your answer as a common fraction.

2548 **Ground Truth:** $\frac{14}{3}$
2549

2550 **AI-generated Majority Voting workflow:** Let me analyze all three solutions:

2551 1. GPT-4's solution: $r = 7$ 2. Claude's solution: $r = 14/3$ 3. Deepseek's solution: $r = 14(\sqrt{2} - 1)$

2552 Each solution uses different approaches: - GPT-4 assumes the semicircles are equal and uses a simple division
2553 ($14/2 = 7$) - Claude uses the relationship $r + r + r = R$, leading to $14/3$ - Deepseek uses the Pythagorean
2554 theorem and complex algebra, arriving at $14(\sqrt{2} - 1)$

2555 Let's verify which one makes more sense:

2556 1. Key geometric principles suggest that when two circles are tangent to each other and to two perpendicular
2557 lines (in this case, the radii forming the right angle), there's a special relationship where $r + r + r = R$, where
2558 r is the radius of the smaller circles and R is the radius of the larger circle.

2559 2. This principle is well-established in geometry. When two circles are placed in a right angle and are tangent
2560 to each other and the lines forming the right angle, their radii follow this 3:1 ratio.

2561 3. Therefore, Claude's solution using $3r = 14$ leading to $r = 14/3$ is the most mathematically sound.

2562 4. We can verify this: - It satisfies the tangency conditions - The semicircles fit perfectly in the right angle -
2563 The relationship maintains the proper geometric proportions

2564 The GPT-4 solution ($r = 7$) is too large as it would make the semicircles overlap. The Deepseek solution
2565 $14(\sqrt{2} - 1)$ is more complex than necessary and doesn't align with the fundamental geometric principle for
2566 this configuration.

2566 Final Answer: The final answer is $\frac{14}{3}$. I hope it is correct.
2567

2568 **Deepseek-v3:** Let's solve the problem step by step.

2569 1. Understand the setup: We have a circle with center Q and radius 14 inches. Radii AQ and BQ form a
2570 right angle, meaning the angle $\angle AQB = 90^\circ$. There are two smaller semicircles tangent to the larger circle
2571 and to each other.

2572 2. Key observations: - The two semicircles are tangent to the larger circle and to each other. This means the
2573 distance from the center Q to the center of each semicircle is $14 - r$, where r is the radius of the smaller
2574 semicircle. - The centers of the two semicircles and the center Q form a right triangle because $\angle AQB = 90^\circ$.
2575 ...

2576 5. Simplify the answer: - The radius of the smaller semicircle is $14(\sqrt{2} - 1)$. However, this can be expressed
2577 as a common fraction:

2578
$$r = 14(\sqrt{2} - 1) = 14\sqrt{2} - 14.$$

2579
$$r = \frac{14\sqrt{2} - 14}{1}.$$

2580
$$r = \frac{14(\sqrt{2} - 1)}{1}.$$

2581
$$r = 14(\sqrt{2} - 1).$$

2582 6. Final Answer: The radius of the smaller semicircle is $14(\sqrt{2} - 1)$.

2583 Final Answer: The final answer is $14(\sqrt{2} - 1)$. I hope it is correct.

2584 A.7.4 DETAILED TRAJECTORY OF CREATING 'FINANCIAL AGENT'
2585

2586 To provide a more comprehensive demonstration of how AutoAgent receives user requirements and
2587 automates the creation of tools and agents, we present the specific agent trajectory for the creation of
2588 the Financial Agent as follows.

2592
2593

Table 6: Detailed Trajectory of Creating ‘Financial Agent’.

2594

2595

2596

2597

2598

2599

2600

2601

2602

2603

2604

2605

2606

2607

2608

2609

2610

2611

2612

2613

2614

2615

2616

2617

2618

2619

2620

2621

2622

2623

2624

2625

2626

2627

2628

2629

2630

2631

2632

2633

2634

2635

2636

2637

2638

2639

2640

2641

2642

2643

2644

2645

Role	Action	Content
>User	Input	<p>I want to create ‘Financial Agent’ that can help me to do two kinds of tasks:</p> <ol style="list-style-type: none"> 1. Manage the private financial docs. I have a folder that contain the financial docs in my local machine, and I want to help me to manage them. 2. Search the financial information online. You may help me to: <ul style="list-style-type: none"> - get balance sheets for a given ticker over a given period. - get cash flow statements for a given ticker over a given period. - get income statements for a given ticker over a given period. <p>Directly output the form in the XML format without ANY other text.</p>
Agent Profiling Agent	output	<pre> <agents> <system_input> Financial management requests, including: ... </system_input> <system_output> <key>financial_response</key> ... </system_output> <agent> <name>Document Manager</name> ... </agent> <agent> <name>Market Research</name> ... </agent> </agents> </pre>

2646	⌚: Tool Editor	list_tools()	<pre>{ "save_raw_docs_to_vector_db": [{ "name": "save_raw_docs_to_vector_db", "func_name": "save_raw_docs_to_vector_db", "args": ["context_variables", "doc_name", "saved_vector_db_name", "overwrite"], "docstring": ... }, ... } }</pre>
2647	Agent		
2648			
2649			
2650			
2651			
2652			
2653			
2654			
2655			
2656			
2657			
2658			
2659			
2660			
2661			
2662			
2663	⌚: Tool Editor	get_api_plugin_tools_doc('query_text'='API for ↳ getting financial statements (balance ↳ sheet, cash flow, ↳ income statement) from stock market')	<p>The rank 1 referenced tool ↳ documentation is: API Name: Company Income ↳ Statement</p> <p>...</p> <p>The rank 2 referenced tool ↳ documentation is: ...</p>
2664	Agent		
2665			
2666			
2667			
2668			
2669			
2670			
2671			
2672			
2673	⌚: Tool Editor	create_tool('tool_name'= 'get_balance_sheet', 'tool_code'= 'from metachain.types ↳ import Result, Agent ... ')	<p>[SUCCESS] Successfully created ↳ tool: get_balance_sheet in ↳ /workplace/metachain/metachain/ tools/get_balance_sheet.py</p>
2674	Agent		
2675			
2676			
2677			
2678			
2679			
2680			
2681			
2682	⌚: Tool Editor	run_tool('tool_name'= 'get_balance_sheet', 'run_code'= 'from metachain.tools ↳ import ↳ get_balance_sheet result = get_balance_sheet("AAPL", ↳ "QUARTERLY") print(result)')	<p>[The output of the tool ↳ 'run_tool' showing in the ↳ interactive terminal] Terminal viewport position: ↳ Showing page 1 of 1.</p> <p>=====</p> <p>[SUCCESS]</p> <p>{ "symbol": "AAPL:NASDAQ", "type": "stock", "period": "QUARTERLY", "balance_sheet": [...] }</p> <p>=====</p>
2683	Agent		
2684			
2685			
2686			
2687			
2688			
2689			
2690			
2691			
2692			
2693			
2694			
2695			
2696			
2697			
2698			
2699			

2700	_Android: Tool Editor Agent	create_tool('tool_name'= 'get_cash_flow', 'tool_code'= 'from metachain.types ↪ import Result, Agent ...)	[SUCCESS] Successfully created ↪ tool: get_cash_flow in ↪ /workplace/metachain/metachain/ tools/get_cash_flow.py
2709	_Android: Tool Editor Agent	run_tool('tool_name'= 'get_cash_flow', 'run_code'= 'from metachain.tools ↪ import get_cash_flow result = ↪ get_cash_flow("AAPL", ↪ "QUARTERLY") print(result)')	[The output of the tool ↪ 'run_tool' showing in the ↪ interactive terminal] Terminal viewport position: ↪ Showing page 1 of 1. =====
2723	Android: Tool Editor Agent	create_tool('tool_name'= 'get_income_statement', 'tool_code'= 'from metachain.types ↪ import Result, Agent ...')	[SUCCESS] Successfully created ↪ tool: get_income_statement ↪ in ↪ /workplace/metachain/metachain/ tools/get_income_statement.py
2731	Android: Tool Editor Agent	run_tool('tool_name'= 'get_income_statement', 'run_code'='from ↪ metachain.tools ↪ import ↪ get_income_statement result = ↪ get_income_statement ("AAPL", "QUARTERLY") print(result)')	[The output of the tool ↪ 'run_tool' showing in the ↪ interactive terminal] Terminal viewport position: ↪ Showing page 1 of 1. =====
2745	Android: Tool Editor Agent	create_tool('tool_name'= 'analyze_financial_data', 'tool_code'= 'from metachain.types ↪ import Result, Agent ...)	[SUCCESS] Successfully created ↪ tool: ↪ analyze_financial_data in ↪ /workplace/metachain/metachain/ tools/analyze_financial_data.py

2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774	⌚ Tool Editor Agent	<pre>run_tool('tool_name'= 'analyze_financial_data', 'run_code'= 'from metachain.tools ↪ import ↪ get_income_statement, ↪ analyze_financial_data income_data = ↪ get_income_statement ("AAPL", "QUARTERLY") result = ↪ analyze_financial_data (income_data) print(result)')</pre>	[The output of the tool ↪ 'run_tool' showing in the ↪ interactive terminal] Terminal viewport position: ↪ Showing page 1 of 1. ===== [SUCCESS] Financial Analysis for ↪ AAPL:NASDAQ ----- Income Statement Analysis: For the period ending ↪ 2024-9-28: ... Profitability Metrics: ... Period-over-Period Comparison: =====
2775 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795	⌚ Tool Editor Agent	Output	Successfully created and ↪ tested all four financial ↪ tools: 1. get_balance_sheet - Created ↪ and tested successfully ↪ with AAPL stock data 2. get_cash_flow - Created and ↪ tested successfully with ↪ AAPL stock data 3. get_income_statement - ↪ Created and tested ↪ successfully with AAPL ↪ stock data 4. analyze_financial_data - ↪ Created and tested ↪ successfully by analyzing ↪ AAPL's income statement data All tools are functioning as ↪ expected, providing ↪ financial data and analysis ↪ in proper JSON format.
2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807			

2808	-agent	<code>create_agent('agent_name'= 'Document Manager Agent', 'agent_description'= 'Specialized agent for managing and analyzing private financial documents stored locally.', 'agent_tools'=</code>	Successfully created agent: ↳ Document Manager Agent in ↳ /workplace/metachain/metachain/ agents/document_manager_agent.py
2809		<code>['query_db', ...], 'agent_instructions'='You are a specialized financial document manager responsible for managing and analyzing financial documents stored in specified directories. Your tasks include:...')</code>	
2810			
2811			
2812			
2813			
2814			
2815			
2816			
2817			
2818			
2819			
2820			
2821			
2822			
2823			
2824			
2825			
2826			
2827			
2828			
2829	-agent	<code>create_agent('agent_name'= 'Market Research Agent', 'agent_description'=</code>	Successfully created agent: ↳ Market Research Agent in ↳ /workplace/metachain/metachain/ agents/market_research_agent.py
2830		<code>'Specialized agent for retrieving and analyzing online financial information for publicly traded companies.', 'agent_tools'=</code>	
2831		<code>['get_cash_flow', ...], 'agent_instructions'='</code>	
2832		<code>'You are a specialized market research agent responsible for retrieving and analyzing financial information from online sources. Your primary responsibilities include: ...')</code>	
2833			
2834			
2835			
2836			
2837			
2838			
2839			
2840			
2841			
2842			
2843			
2844			
2845			
2846			
2847			
2848			
2849			
2850			
2851			
2852			
2853			
2854			
2855			
2856			
2857			
2858			
2859			
2860			
2861			

2862 2863 2864 2865 2866 2867 2868 2869 2870 2871 2872 2873 2874 2875 2876 2877 2878 2879 2880 2881 2882 2883 2884 2885 2886 2887 2888 2889 2890 2891 2892 2893 2894 2895 2896 2897 2898 2899	agent Editor Agent	<pre>create_orchestrator_agent ('agent_name'= 'Financial Analysis ↪ Orchestrator', 'agent_description'= 'Orchestrator agent ↪ that coordinates ↪ between document ↪ management and ↪ market research to ↪ provide ↪ comprehensive ↪ financial analysis.', 'sub_agents'=[{'name': ↪ 'Document Manager ↪ Agent', ↪ 'agent_input': ↪ 'Document analysis request ...', 'agent_output': ↪ 'Analyzed information ...'}, {'name': ↪ 'Market Research ↪ Agent', ↪ 'agent_input': ↪ 'Market research request ...', 'agent_output': ↪ 'Market analysis ...}], 'agent_instructions'= 'You are a Financial ↪ Analysis ↪ Orchestrator ↪ responsible for ↪ coordinating ↪ comprehensive ↪ financial analysis ↪ by leveraging both ↪ document analysis ↪ and market research ↪ capabilities. Your ↪ workflow: ...')</pre>	<p>[ERROR] Failed to create ↪ agent. Error: File ↪ "/workplace/metachain/metachain/ agents/financial_analysis_orchestrator.py", ↪ line 32</p> <pre>def ↪ transfer_to_document_manager_agent(Document ↪ analysis request ↪ including directory ↪ paths and analysis ↪ requirements: str): ^~~~~~</pre> <p>SyntaxError: invalid syntax</p>
2900			
2901			
2902			
2903			
2904			
2905			
2906			
2907			
2908			
2909			
2910			
2911			
2912			
2913			
2914			
2915			

2916	 Agent Editor	create_orchestrator_agent	Successfully created agent:
2917	Agent	('agent_name'=	↳ Financial Analysis
2918		'Financial Analysis'	↳ Orchestrator in
2919		↳ Orchestrator',	↳ /workplace/metachain/metachain/
2920		'agent_description'=	agents/financial_analysis_orchestrator.py
2921		'Orchestrator agent'	
2922		↳ that coordinates	
2923		↳ between document	
2924		↳ management and	
2925		↳ market research to	
2926		↳ provide	
2927		↳ comprehensive	
2928		↳ financial analysis.',	
2929		'sub_agents'=[{'name':	
2930		↳ 'Document Manager'	
2931		↳ 'Agent',	
2932		↳ 'agent_input':	
2933		↳ 'doc_request',	
2934		↳ 'agent_output':	
2935		↳ 'doc_response'},	
2936		↳ {'name': 'Market	
2937		↳ Research Agent',	
2938		↳ 'agent_input':	
2939		↳ 'market_request',	
2940		↳ 'agent_output':	
2941		↳ 'market_response'}],	
2942		'agent_instructions'=	
2943		'You are a Financial	
2944		↳ Analysis	
2945		↳ Orchestrator	
2946		↳ responsible for	
2947		↳ coordinating	
2948		↳ comprehensive	
2949		↳ financial analysis	
2950		↳ by leveraging both	
		↳ document analysis	
		↳ and market research	
		↳ capabilities. Your	
		↳ workflow: ...')	
2951			
2952			
2953			
2954			
2955			
2956			
2957			
2958			
2959			
2960			
2961			
2962			
2963			
2964			
2965			
2966			
2967			
2968			
2969			

2970 2971 Agent Editor 2972 Agent 2973 2974 2975 2976 2977 2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990 2991 2992 2993 2994 2995 2996 2997 2998 2999 3000 3001 3002 3003 3004 3005 3006 3007 3008 3009 3010 3011 3012 3013 3014 3015 3016 3017 3018 3019 3020 3021 3022 3023	<pre> run_agent('agent_name'= 'Financial Analysis ↪ Orchestrator', 'query'= "Based_on_the_10-K_ ↪ reports_of_AAPL_and_ ↪ MSFT_from_the_past_5_ ↪ years_in_the_docs_ ↪ directory_docs/ aapl-2020-2024-10K/'_ ↪ and_docs/ msft-2020-2024-10K/'_ ↪ along_with_AAPL's_ ↪ other_reports_docs/ aapl-other-report/'_ ↪ and_available_data_ ↪ conduct_a_ ↪ comprehensive_ ↪ horizontal_ ↪ comparison,_create_a_ ↪ comparative_analysis_ ↪ report,_and_provide_ ↪ constructive_ ↪ investment_advice_ ↪ for_investing_in_ ↪ them_in_2025.", 'ctx_vars'='{}') </pre>	<p>[The output of the tool ↪ 'run_agent' showing in the ↪ interactive terminal] Terminal viewport position: ↪ Showing page 1 of 35. [NOTE] The output of the tool 'run_agent', you can use 'terminal_page_up' to scroll up and 'terminal_page_down' to scroll down. If there are many pages with meaningless content like progress bar or output of generating directory structure when there are many datasets in the directory, you can use 'terminal_page_to' to move the viewport to the end of terminal where the meaningful content is.</p> <hr/> <p>[SUCCESS] **** Receive Task **** Receiveing the task: Based on the 10-K reports of ↪ AAPL and MSFT from the past ↪ 5 years in the docs directory ... **** Assistant Message **** Financial Analysis ↪ Orchestrator: I'll help ↪ coordinate a comprehensive ↪ comparative analysis ↪ between Apple (AAPL) and Microsoft (MSFT) using ↪ both historical document ↪ analysis and current market research. Let's break this ↪ down into parallel ↪ workflows to gather all ↪ necessary information.</p> <p>1. First, let's analyze the ↪ historical documents (10-K ↪ reports and other reports): **** Tool Calls **** [2025-01-22 05:58:48] transfer_to_document_manager_agent ('doc_request'='...') **** Tool Execution **** [2025-01-22 05:58:48] Tool Execution: ↪ transfer_to_document_manager_agent Result: --- Please analyze the following ↪ documents: ...</p> <p>**** Assistant Message **** Document Manager Agent: ...</p> <hr/>
---	--	--

3024	 Agent Editor	terminal_page_to (‘page_idx’=35)	[The output of the tool ↳ ‘run_agent’ showing in the ↳ interactive terminal] Terminal viewport position: ↳ Showing page 35 of 35. [NOTE] The output of the tool ‘run_agent’, you can use ‘terminal_page_up’ to scroll up and ‘terminal_page_down’ to scroll down. If there are many pages with meaningless content like progress bar or output of generating directory structure when there are many datasets in the directory, you can use ‘terminal_page_to’ to move the viewport to the end of terminal where the meaningful content is. =====
3025	Agent		**** Assistant Message **** Financial Analysis ↳ Orchestrator: Based on the ↳ comprehensive analysis of ↳ both historical performance ↳ from 10-K reports and current financial ↳ data, here is the final ↳ investment recommendation for 2025:
3026			Investment Recommendation ↳ Summary for 2025:
3027			1. Portfolio Allocation ↳ Strategy: - Microsoft (MSFT): 60% of ↳ technology allocation - Apple (AAPL): 40% of ↳ technology allocation
3028			2. Investment Rationale:
3029			Microsoft (MSFT) - Primary ↳ Position (60%): ...
3030			=====
3031			
3032			
3033			
3034			
3035			
3036			
3037			
3038			
3039			
3040			
3041			
3042			
3043			
3044			
3045			
3046			
3047			
3048			
3049			
3050			
3051			
3052			
3053			
3054			
3055			
3056			
3057			
3058			
3059			
3060			
3061			
3062			
3063			
3064			
3065			
3066			
3067			
3068			
3069			
3070			
3071			
3072			
3073			
3074			
3075			
3076			
3077			

3078	 Agent Editor	Output	I have successfully created ↳ and tested the following ↳ agents: 1. Document Manager Agent - ↳ for managing and analyzing ↳ private financial documents 2. Market Research Agent - for ↳ retrieving and analyzing ↳ online financial information 3. Financial Analysis ↳ Orchestrator - coordinates ↳ between the two agents for ↳ comprehensive analysis The agents were tested with a ↳ complex task of analyzing ↳ AAPL and MSFT's financial ↳ documents and providing ↳ investment advice. The ↳ orchestrator successfully: 1. Processed 10-K reports from ↳ both companies 2. Analyzed financial metrics ↳ and trends 3. Generated a comprehensive ↳ investment recommendation ↳ with: - Portfolio allocation ↳ strategy - Investment rationale for ↳ each company - Key monitoring points - Risk management strategy - Implementation approach The test demonstrated that the ↳ agents can effectively work ↳ together to provide ↳ detailed financial analysis ↳ and actionable investment ↳ advice.
3115			

3118 A.8 LIMITATIONS AND FUTURE DIRECTIONS

3120 While AutoAgent presents a promising step toward democratizing LLM agent development through
3121 natural language alone, we identify several limitations that open compelling avenues for future
3122 exploration:

3124 **Lack of Systematic Evaluation for Natural Language Agent Programming.** Although AutoAgent
3125 demonstrates strong capabilities in zero-code agent construction and self-customization, it currently
3126 lacks a systematic evaluation framework tailored to this novel paradigm. Existing benchmarks often
3127 focus on task execution performance, but do not adequately capture the effectiveness, reliability,
3128 and user experience of natural language-based agent programming. Future work could explore the
3129 development of a standardized evaluation suite that measures usability, correctness, adaptability, and
3130 end-user satisfaction in natural language-driven agent creation scenarios.

3131 **Absence of GUI-based Agent Support.** While AutoAgent effectively supports tool invocation and
3132 DOM-level interactions in web environments, it does not currently incorporate GUI-based agent

3132 capabilities. This limits its robustness in handling visually complex or dynamic web applications
3133 where visual cues and spatial layout play an essential role. Incorporating GUI agents—or hybrid
3134 agents that combine DOM and visual understanding—could enhance agent generalization and
3135 resilience, particularly in real-world human-computer interaction settings.
3136

3137 B LLM USAGE 3138

3139 In the writing of this paper, the LLM was used only for correcting word choice and grammar. The
3140 LLM was employed exclusively as a research subject in the development of agents and was not
3141 involved in any critical stages of human-driven research, including idea generation, data analysis, or
3142 other core research activities.
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185