

AutoAgent: A Fully-Automated and Zero-Code Framework for LLM Agents

Anonymous ACL submission

Abstract

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

1 Introduction

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.

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

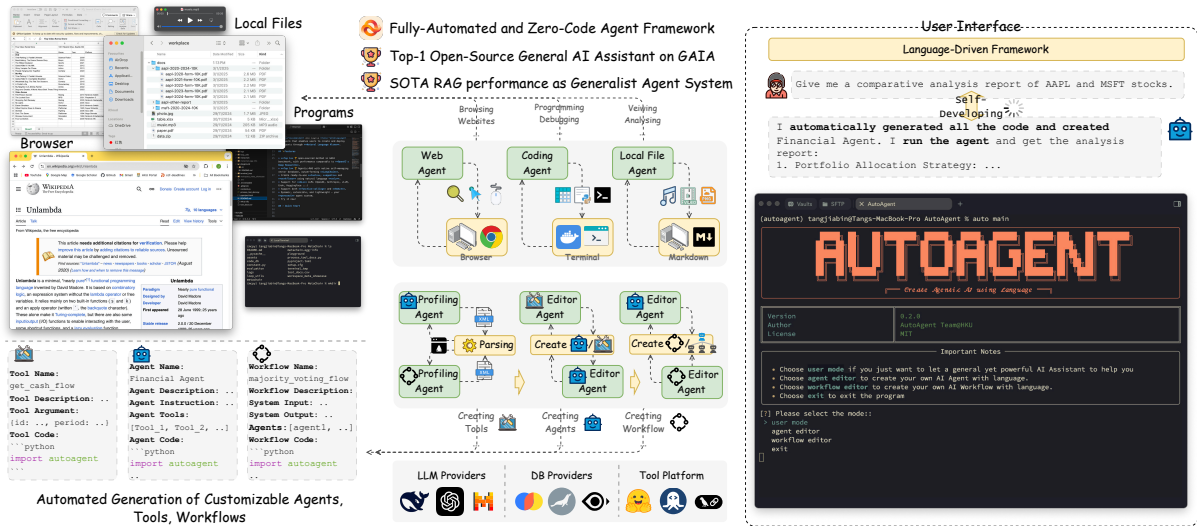


Figure 1: AutoAgent stands out as a new LLM Agent Framework that enables fully automated, zero-code development for complex task automation. Ranking #1 among open-source solutions on the GAIA benchmark, it delivers state-of-the-art RAG performance as a general AI assistant. Its revolutionary approach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

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.

This gap between universal needs and limited accessibility motivates a key question: *Can we democratize LLM agent development via Natural Language-based Creation and Customization?* We pursue this goal with AutoAgent, a framework that reframes agent development as a fully automated, language-driven process requiring zero programming. AutoAgent acts as an autonomous Agent Operating System with three core capabilities: 1) **Natural Language-Driven Multi-Agent Building**—automatically constructing and orchestrating agent systems through dialogue, without coding or manual configuration; 2) **Self-Managing Workflow Generation**—dynamically generating, optimizing, and adapting workflows from high-level task descriptions when users cannot specify details; and 3) **Intelligent Resource Orchestration**—offering unified natural-language access to tools, APIs, and compute while automatically managing allocation and optimization. Together, these features democratize LLM agent development with enterprise-grade sophistication, turning complex

engineering into an intuitive conversation for all users.

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.

AutoAgent’s exceptional capabilities have been rigorously validated through comprehensive empirical evaluation. In standardized benchmarks,

151 it secured a strong second place on the General- 203
152 ist Agent Benchmark (GAIA), while significantly 204
153 outperforming state-of-the-art RAG approaches on 205
154 the Retrieval-Augmented Generation benchmark. 206
155 Beyond these quantitative achievements, extensive 207
156 case studies demonstrated AutoAgent’s robust self- 208
157 development capabilities across diverse real-world 209
158 scenarios, highlighting its practical value in auto- 210
159 mated agent development. 211

160 2 Related Work and Preliminaries 212

161 LLM-empowered agents have transformed AI 213
162 systems via tool invocation. Frameworks 214
163 such as LangChain (LangChain, 2023), Auto- 215
164 GPT (Significant-Gravitas, 2023), CAMEL (Li 216
165 et al., 2023), MetaGPT (Hong et al., 2024a), 217
166 and OpenAgent (Xie et al., 2023) showcase di- 218
167 verse strengths: CAMEL introduced role-playing 219
168 communication, AutoGen (Wu et al., 2023) ex- 220
169 plored LLM collaborations for problem-solving, 221
170 MetaGPT (Hong et al., 2024a) incorporated 222
171 Standardized Operating Procedures, and OpenA- 223
172 gent (Xie et al., 2023) offered specialized web 224
173 agents. However, these frameworks often demand 225
174 coding skills and domain expertise, limiting ac- 226
175 cess for non-technical users. We propose a new 227
176 paradigm that democratizes agent development 228
177 through natural language, enabling broad access 229
178 regardless of technical background. 230

179 **LLM-Empowered Agent.** The task-solving pro- 231
180 cess of Large Language Model (LLM) agents 232
181 can be formalized as a Markov Decision Process 233
182 (MDP) to model environment interaction. Defined 234
183 as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P(\cdot), \mathcal{E})$, it specifies state space 235
184 \mathcal{S} , action space \mathcal{A} , observation space \mathcal{O} , transition 236
185 function $P(\cdot)$, and environments \mathcal{E} . At each time 237
186 step, the agent observes the state, selects an action 238
187 via its policy, interacts with the environment, and 239
188 updates its state (the agent “context”). State-to- 240
189 action mapping typically follows **Tool-Use** (Yao 241
190 et al., 2024), leveraging external capabilities, or **Re-** 242
191 **Act** (Yao et al., 2023) (Non-tool-use), producing 243
192 actions solely from the language model. This MDP 244
193 view supports understanding, analysis, and design 245
194 of LLM agents for complex multi-step tasks. 246

195 **Generalist Multi-Agent System.** Multi-agent 247
196 systems (MAS) address single-agent limits in com- 248
197 plex problems by combining multiple specialized 249
198 agents. Generalist Multi-Agent Systems coordinate 250
199 agent teams with an Orchestrator to solve diverse 251
200 tasks through collaboration. 252

201 In a Generalist MAS, there are multiple agents, 253
202 denoted as $\pi_0 : S_0 \rightarrow A_0, \pi_1 : S_1 \rightarrow A_1, \dots, \pi_n :$

$S_n \rightarrow A_n$. Each agent’s action set includes a spe- 203
cial **transfer action** $\hat{A}_i \in A_i$ for delegating tasks 204
to other agents. The key challenge is an effective 205
Task Transfer Mechanism that organizes agents 206
via transfer actions. We refer to this organization 207
as the “MAS Design Pattern”. A common design 208
is the **Orchestrator-Workers** paradigm (Fourney 209
et al., 2024; Anthropic, 2024), where the **Orches-** 210
trator decomposes tasks and assigns subtasks to 211
Workers, which execute and return results through 212
transfer actions. 213

214 For tasks with deterministic steps, workflow- 215
216 driven mechanisms are often effective. Examples 217
include GPTSwarm (Zhuge et al.), which repre- 218
sents workflows as computational graphs for com- 219
plex data operations, and systems for mathemat- 220
ical reasoning (Wang et al., 2023) and code gen- 221
eration (Wang et al., 2024a). These advances en- 222
able strong performance in GUI interaction (Hong 223
et al., 2024b; Xu et al., 2024), software develop- 224
ment (Wang et al., 2024b; Yang et al., 2024), web 225
browsing (Drouin et al., 2024; Song et al., 2024), 226
and embodied tasks (Li et al., 2024), highlighting 227
the versatility of LLM agent systems. 228

229 **Workflow Design in Generalist MAS.** While 230
231 Generalist Multi-Agent Systems are flexible for 232
233 open-ended tasks, they can incur higher cost, com- 234
235 plexity, and compounding errors (Anthropic, 2024). 236
237 For deterministic processes with domain expertise, 238
239 fixed workflows can be more stable. 240

241 A workflow in a MAS is defined as $\mathcal{W} =$ 242
243 $w_{i,j} : \pi_i \xrightarrow{c_k} \pi_j$, where π_i and π_j are agents and c_k 244
245 is the transfer condition. Workflow design defines 246
247 conditional transfer relations between agents to 248
249 ensure coordinated execution—the “Workflow De- 250
251 sign Pattern”. Common patterns include **Routing,** 252
253 **Parallelization,** and **Evaluator-Optimizer** (An- 254
255 thropic, 2024), supporting stable and efficient MAS 256
257 execution. 258

259 **Fully-Automated Generalist MAS.** Designing 260
261 and executing MAS and complex workflows typ- 262
263 ically requires substantial expertise; AutoAgent 264
265 aims to fully automate this. The key challenge 266
267 is bridging high-level user requirements to imple- 268
269 mentable MAS and workflow solutions through 270
271 natural language interactions. 272

273 3 The AutoAgent Framework 274

275 AutoAgent is built as an automated operating sys- 276
277 tem for LLM agents and general AI assistants. In- 278
279 spired by modern OS design, it integrates core com- 280
281 ponents for natural language-driven agent develop- 282
283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302

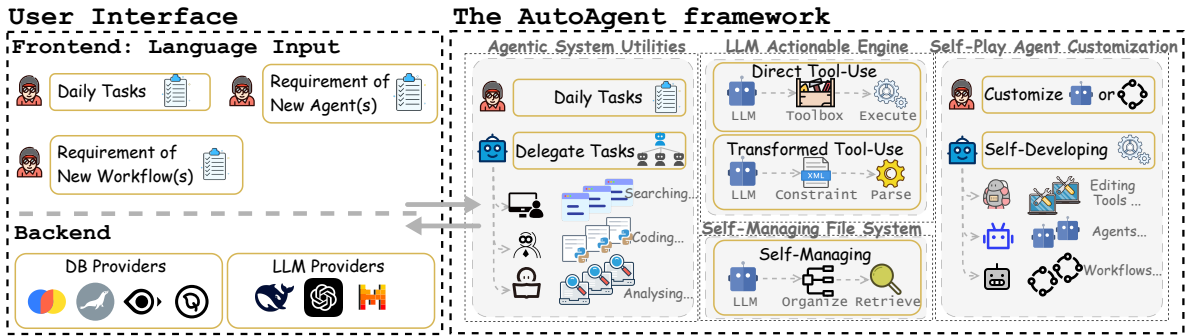


Figure 2: AutoAgent is a fully automated, language-driven generalist agent system. The core components that enable this include the Agentic System Utilities, the LLM-powered Actionable Engine, the Self-Managing File System, and the Self-Play Agent Customization module.

ment and task execution (Fig 2). **Agentic System Utilities** supply foundational building blocks, the **LLM-powered Actionable Engine** acts as the central brain for understanding inputs and orchestrating multi-agent coordination, the **Self-Managing File System** enables structured storage and retrieval of user multi-modal data, and **Self-Play Agent Customization** lets users create tailored agents and workflows via natural language without coding. Together, these capabilities make AutoAgent a versatile platform for autonomous agent-based solutions across diverse applications.

3.1 Agentic System Utilities

The AutoAgent framework adopts a modular multi-agent architecture to build intelligent personal assistant agents that integrate and coordinate diverse capabilities, including web browsing, information retrieval, data analysis, and code execution. It combines specialized agents (web, coding, file management) with an orchestrator that decomposes and delegates user requests, forming a versatile and extensible foundation that supports rapid development of tailored agent-driven solutions. Detailed system prompts and tool definitions for **Agentic System Utilities** are provided in Appendix Sec A.

3.1.1 Orchestrator Agent

The Orchestrator Agent serves as the main user interface. It receives user tasks, understands them, decomposes them into sub-tasks, and delegates each to suitable sub-agents via handoff tools (OpenAI, 2024). Sub-agents return results through handoff, and the Orchestrator iteratively assigns subsequent sub-tasks based on progress until the overall task is completed. This handoff-based design provides a simple, effective alternative to complex planning prompts.

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.

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

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

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

These three agent modules work collaboratively,

forming light-weight yet complete basic Agent System, laying the foundation for future functional expansion.

3.2 LLM-powered Actionable Engine

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

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

3.2.1 Generating Actionable Reflections

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

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

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

3.3 Self-Managing File System

The file system in AutoAgent is a vector database that LLM agents can retrieve and understand. In our design framework, users can upload text files in any format (*e.g.*, .pdf, .doc, .txt) or compressed archives and folders containing any

text files. The system tools in the file system automatically convert these files into a consistent text format and store them in a user-defined collection within the vector database (using the `save_raw_docs_to_vector_db` tool). This enables agents to self-manage their database memory and perform efficient and accurate retrieval and generation using tools like `query_db` and `answer_query`. The detailed definitions of the tools are presented in Tab 4.

3.4 Self-Play Agent Customization

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

3.4.1 Agent Creation without Workflow

Building effective multi-agent systems often requires domain expertise (*e.g.*, financial regulations or healthcare protocols), which many users lack. For instance, automating investment portfolio management in finance demands knowledge of asset allocation, risk modeling, and regulatory compliance.

To mitigate this, AutoAgent offers a workflow-based mode that lets users create sophisticated agent systems with minimal domain knowledge. Users provide high-level specifications (*e.g.*, agent name and a short description of desired functions), and AutoAgent automatically generates the agent(s) and workflow(s) to coordinate them, grounded in the framework’s current tools, agents, and workflows. The main steps are:

- **Analyze Requirements and Existing Components.** The system analyzes user needs against available tools and agents, using a profiling agent to assess current capabilities and reusable resources.
- **Analyze Tools and Structure Agents.** It decides whether new tools are needed, evaluates reuse of existing tools, and organizes collaboration among agents to maximize resource use and ensure efficient system design.
- **Generate Detailed XML Agent Specifications.** It produces structured XML capturing agent requirements, including functionality, dependencies,

and interactions, to support downstream agent generation.

Optimized Tool Creation with Third-Party APIs. The Tool Editor Agent integrates third-party APIs (e.g., LangChain, RapidAPI, Hugging Face) to build tools, retrieving relevant documentation (names, descriptions, invocation). The system currently supports 145 RapidAPI APIs across 8 categories, LangChain (LangChain, 2023), and models from 9 Hugging Face categories, with plans to add platforms like Composio (Composio, 2024).

It also generates tool code with automatic syntax checking, test design and execution, and iterative debugging until success, enabling a customizable tool set without a bloated integrated system.

Agent Creation and Execution. When requirements span multiple agents, the Agent Editor Agent performs multi-step creation. After agents are created, the system calls `create_orchestrator_agent` to build an orchestrator connecting them, following the Orchestrator-Workers pattern with prompts covering task description, subtask decomposition rules, and scenario details. Algorithms and prompts are in Appendix Sec A.6.1.

3.4.2 Agent Creation with Workflow

When users have explicit workflow requirements and domain knowledge, AutoAgent supports a tailored mode. Users describe the desired agent(s) and specify target tasks, and AutoAgent generates both the agent(s) and workflow(s) to coordinate collaboration toward these objectives.

Traditional graph-based approaches require strict graph-theoretic structures (Zhuge et al., 2024; LangChain, 2024; Hu et al., 2024; Zhang et al., 2024), which can be difficult for LLMs to generate reliably. Instead, AutoAgent uses an event-driven design that treats each agent’s task-solving process as an event. With event listening and triggering, agents collaborate flexibly without rigid graph constraints.

Constructing New Workflows. Creating a new workflow is itself a multi-agent process. The Workflow Form Agent analyzes requirements and existing tools/agents to decide whether new agents are needed, which agents compose the workflow, and the event listening/triggering logic. It then outputs structured XML.

During parsing, an error detection mechanism checks whether the workflow form satisfies system constraints (e.g., on `on_start`). If violated, detailed errors are returned to the Workflow Form

Table 1: Performance comparison between the baseline models and our AutoAgent on the GAIA benchmark. 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 (o1)	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

Agent for regeneration. If valid, the form is sent to the Workflow Editor Agent, which creates agents if needed, builds the workflow, and executes it on the task. Algorithms and prompts are in Appendix Sec A.6.2.

4 Evaluation

4.1 Evaluation for a Generalist Agent System Dataset and Evaluation Protocols.

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

Baseline Methods. The baselines we selected are divided into two categories: Open-Source: FRIDAY (Wu et al., 2024), Magentic-1 (Fourney et al., 2024), Multi-Agent Experiment v0.1 (powered by AutoGen)(Microsoft, 2024), HuggingFace Agents(HuggingFace, 2024), Langfun Agent (Google, 2024); Closed-Source: TapeAgent, AgentIM, Trase Agent (Trase, 2024), Omne, Barcelona¹, and the h2oGPTE Agent (H2O.ai, 2024). These diverse baselines represent the current state-of-the-art in open-source and proprietary multi-agent systems, providing a comprehensive landscape for evaluating the performance and capabilities of our proposed AutoAgent framework.

Implementation Details. To address tasks in the GAIA benchmark, we utilize a combination of the System Utilities of the Model and the Tool Editor Agent from the Agentic-SDK. The basic agents

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

first attempt to complete the task while collecting relevant information and reflections. If successful, the result is directly returned. If not, the Tool Editor Agent creates new tools to continue the task. During validation, Claude-Sonnet-3.5 is used by default.

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

- **Obs.1. Overall Superiority of AutoAgent:** Our method significantly outperforms all open-source agent systems and achieves performance close to the latest agent system, h2oGPTe Agent v1.6.8 (submitted on December 16, 2024), securing a stable position in the top 2 rankings. Notably, our approach demonstrates superior performance on Level 1 tasks compared to all state-of-the-art baselines, becoming the first method to achieve over 70% accuracy rate. This success is attributed to the well-designed System Utilities and the stable interaction of basic agents with the environment, enabling efficient solutions to everyday simple tasks.

- **Obs.2. Effectiveness of Key Components:** Specifically, our framework demonstrates significantly superior performance compared to Magentic-1 (Fourney et al., 2024), a recent representative open-source MAS, and FRIDAY, a classic self-improved framework. While Magentic-1 leverages the powerful reasoning capabilities of o1-preview to design complex Orchestrator Agent (also the Coder Agent), our framework emphasizes the stability of interactions between sub-agents and their respective environments, as well as the precision of tool definitions. Under these conditions, the Orchestrator Agent achieves better results with simple prompts and handoff tools.

4.2 Evaluation of AutoAgent on the Retrieval-Augmented Generation Task Benchmark Dataset and Evaluation Protocols.

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

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

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

Table 2: Evaluation of AutoAgent for RAG.

Method	<i>acc</i>	<i>err</i>	<i>acc</i>	<i>err</i>
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%

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

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

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

4.3 AutoAgent’s Performance on Open-Ended Tasks

This section thoroughly explores the capabilities of the AutoAgent framework in generating agents and workflows based on even vague, natural language

inputs across various scenarios. To illustrate the breadth of AutoAgent’s abilities, we will examine its performance on tasks of varying difficulty - from the creation of a single agent to the orchestration of multiple, coordinated agents. (We put the results of single agent in Appendix A.7.1)

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

I want to create `Financial Agent` that can help me to do two kinds of tasks:

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:

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

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 advances the democratization of LLM-powered agents, making them accessible to the non-programming majority. By linking high-level natural language requirements to practical multi-agent systems and workflows, AutoAgent enables users to create, customize, and deploy agents, tools, and workflows without substantial technical expertise. Its modular architecture, Agentic System Utilities, and LLM-powered Actionable Engine together support automated agent development and task execution. Features such as the Self-Organizing File System and Self-Play Agent Customization further strengthen AutoAgent, enabling dynamic agent evolution and task-specific optimization. Extensive evaluations show AutoAgent’s strong performance, underscoring its potential to broaden access to LLM capabilities.

6 Limitations

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

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

Absence of GUI-based Agent Support. While AutoAgent effectively supports tool invocation and DOM-level interactions in web environments, it does not currently incorporate GUI-based agent capabilities. This limits its robustness in handling visually complex or dynamic web applications where visual cues and spatial layout play an essential role. Incorporating GUI agents—or hybrid agents that combine DOM and visual understanding—could enhance agent generalization and resilience, particularly in real-world human-computer interaction settings.

756

References

757
758
759

Anthropic. 2024. [Building effective agents](https://www.anthropic.com/research/building-effective-agents). <https://www.anthropic.com/research/building-effective-agents>.

760
761
762

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

763
764
765

Composio. 2024. [Composio: Production ready toolset for ai agents](https://github.com/ComposioHQ/composio). <https://github.com/ComposioHQ/composio>.

766
767
768
769
770
771
772
773
774

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. [WorkArena: How capable are web agents at solving common knowledge work tasks?](#) In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR.

775
776
777

E2B. 2024. [Secure open source cloud runtime for ai apps & ai agents](https://github.com/e2b-dev/e2b). <https://github.com/e2b-dev/e2b>.

778
779
780
781

Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. [Minirag: Towards extremely simple retrieval-augmented generation](#). *arXiv preprint arXiv:2501.06713*.

782
783
784
785
786
787
788
789

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

790
791
792

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. [Precise zero-shot dense retrieval without relevance labels](#). *arXiv preprint arXiv:2212.10496*.

793
794

Google. 2024. [Langfun: Oo for llms](https://github.com/google/langfun). <https://github.com/google/langfun>.

795
796
797

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. [Lightrag: Simple and fast retrieval-augmented generation](#).

798
799
800

H2O.ai. 2024. [Autonomous agentic ai: execute multi-step workflows autonomously](https://h2o.ai/platform/enterprise-h2ogpte/#AgenticAI). <https://h2o.ai/platform/enterprise-h2ogpte/#AgenticAI>.

801
802
803
804
805
806
807
808

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024a. [MetaGPT: Meta programming for a multi-agent collaborative framework](#). In *The Twelfth International Conference on Learning Representations*.

Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2024b. [Cogagent: A visual language model for GUI agents](#). In *CVPR*, pages 14281–14290. IEEE. 809
810
811
812
813

Shengran Hu, Cong Lu, and Jeff Clune. 2024. [Automated design of agentic systems](#). *CoRR*, abs/2408.08435. 814
815
816

HuggingFace. 2024. [Transformers documentation: Agents and toos](https://huggingface.co/docs/transformers/agents). <https://huggingface.co/docs/transformers/agents>. 817
818
819

LangChain. 2023. [Langchain: Build context-aware reasoning applications](https://github.com/langchain-ai/langchain). <https://github.com/langchain-ai/langchain>. 820
821
822

LangChain. 2024. [Langgraph: sbuild resilient language agents as graphs](https://github.com/langchain-ai/langgraph). <https://github.com/langchain-ai/langgraph>. 823
824
825

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. [CAMEL: communicative agents for "mind" exploration of large language model society](#). In *NeurIPS*. 826
827
828
829

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, and 1 others. 2024. [Embodied agent interface: Benchmarking llms for embodied decision making](#). In *NeurIPS 2024*. 830
831
832
833
834

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let's verify step by step](#). In *ICLR*. Open-Review.net. 835
836
837
838
839

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. [Generation-augmented retrieval for open-domain question answering](#). *arXiv preprint arXiv:2009.08553*. 840
841
842
843
844

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. [GAIA: a benchmark for general AI assistants](#). In *ICLR*. Open-Review.net. 845
846
847
848

Microsoft. 2024. [Multi-agent experiment v0.1 msr ai frontiers \(autogen team members\)](https://aka.ms/gaia_multiagent_v01_march_1st). https://aka.ms/gaia_multiagent_v01_march_1st. 849
850
851

OpenAI. 2023. [Gpt-4 technical report](https://arxiv.org/abs/2303.08774). *Preprint*, arXiv:2303.08774. 852
853

OpenAI. 2024. [Educational framework exploring ergonomic, lightweight multi-agent orchestration](https://github.com/openai/swarm). <https://github.com/openai/swarm>. 854
855
856

Significant-Gravitas. 2023. [Autogpt](https://github.com/Significant-Gravitas/AutoGPT). <https://github.com/Significant-Gravitas/AutoGPT>. 857
858

Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. 2024. [Beyond browsing: Api-based web agents](#). *arXiv preprint arXiv:2410.16464*. 859
860
861

935 **A Appendix**

936 In the supplementary materials, we provide a de-
937 tailed technical description of the 'Agentic System
938 Utilities' implementation within our AutoAgent
939 framework.

940 **A.1 System-level Tools**

941 To empower our diverse array of system-level
942 agents, we have carefully curated and predefined
943 seven distinct categories of powerful tools. These
944 tools span a wide range of functionalities, including
945 coding, web browsing, file management, creating
946 new tools, agents, and workflows, as well as natural
947 language question answering for documents. The
948 detailed names and comprehensive descriptions of
949 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.
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.

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.
input_text	Web	Types the given text value into the specified field.
click	Web	Clicks the mouse on the target with the given element bid.
page_down	Web	Scrolls the entire browser viewport one page DOWN towards the end.
page_up	Web	Scrolls the entire browser viewport one page UP towards the beginning.
history_back	Web	Navigates back one page in the browser's history. This is equivalent to clicking the browser back button.
history_forward	Web	Navigates forward one page in the browser's history. This is equivalent to clicking the browser forward button.
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 inputing queries into search boxes).
web_search	Web	Performs a web search on ' https://www.google.com.sg/?hl=en&gl=US ' with the given query.
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.
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.
open_local_file	File	Open a local file at a path in the text-based browser and return current viewport content.
page_up_markdown	File	Scroll the viewport UP one page-length in the current file and return the new viewport content.
page_down_markdown	File	Scroll the viewport DOWN one page-length in the current file and return the new viewport content.

find_next	File	Scroll the viewport to next occurrence of the search string.
visual_question_answering	File	This tool is used to answer questions about attached images or videos.
find_on_page_ctrl_f	File	Scroll the viewport to the first occurrence of the search string. This is equivalent to Ctrl+F.
list_tools	Tools Edit	List all plugin tools in the MetaChain.
create_tool	Tools Edit	Create a plugin tool.
delete_tool	Tools Edit	Delete a plugin tool.
run_tool	Tools Edit	Run a tool with the given code.
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'].
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.
get_api_plugin_tools_doc	Tools Edit	Retrieve satisfied tool documents based on the query text.
list_agents	Agents Edit	List all plugin agents in the MetaChain.
read_agents	Agents Edit	Get detailed information of plugin agents in the MetaChain.
delete_agent	Agents Edit	Delete a plugin agent.
run_agent	Agents Edit	Run a plugin agent.
create_agent	Agents Edit	Use this tool to create a new agent or modify an existing agent.

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.
create_workflow	Workflows Edit	Create a workflow.
list_workflows	Workflows Edit	List all workflows in the MetaChain.
run_workflow	Workflows Edit	Run a workflow.
save_raw_docs_to_vector_db	RAG	Save the raw documents to the vector database. The documents could be: - ANY text document with the extension of pdf, docx, txt, etc. - 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.
query_db	RAG	Retrieve information from the database. Use this function when you need to search for information in the database.
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.
answer_query	RAG	Answer the user query based on the supporting documents.
can_answer	RAG	Check if you have enough information to answer the user query.

A.2 Web Agent

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

Listing 1: Tools of **Web Agent**

```
[click, page_down, page_up, history_back, history_forward, web_search, input_text, sleep, visit_url, get_page_markdown, transfer_back_to_orchestrate_agent]
```

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

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

A.3 Local File Agent

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

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

```
[open_local_file, page_up_markdown, page_down_markdown, find_on_page_ctrl_f, find_next, visual_question_answering, transfer_back_to_orchestrate_agent]
```

Listing 4: System Prompt of **Local File Agent**

```
You are a file surfer agent that can handle
↳ local files.
```

```
You can only access the files in the folder
↳ `/workplace` and when you want to open a
↳ file, you should use absolute path from
↳ root like `/workplace/...`.
```

```
Note that `open_local_file` can read a file
↳ as markdown text and ask questions about
↳ it. And `open_local_file` can handle the
↳ following file extensions: [".html",
↳ ".htm", ".xlsx", ".pptx", ".wav", ".mp3",
↳ ".flac", ".pdf", ".docx"], and all other
↳ types of text files.
```

```
But IT DOES NOT HANDLE IMAGES, you should use
↳ `visual_question_answering` to see the
↳ image.
```

```
If the converted markdown text has more than
↳ 1 page, you can use `page_up`,
↳ `page_down`, `find_on_page_ctrl_f`,
↳ `find_next` to navigate through the pages.
```

```
When you think you have completed the task
↳ the `System Orchestrate Agent` asked you
↳ to do, you should use
↳ `transfer_back_to_orchestrate_agent` to
↳ transfer the conversation back to the
↳ `System Orchestrate Agent`. And you should
↳ not stop to try to solve the user's
↳ request by transferring to `System
↳ Orchestrate Agent` only until the task is
↳ completed.
```

```
If you are unable to open the file, you can
↳ transfer the conversation back to the
↳ `System Orchestrate Agent`, and let the
↳ `Coding Agent` 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**

```
[gen_code_tree_structure, execute_command, read_file, create_file, write_file, list_files, create_directory, run_python, terminal_page_up, terminal_page_down, terminal_page_to, 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.
You 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`.
```

1072 3. You must run python scripts using function
 1073 ↪ ``run_python`` rather than using the
 1074 ↪ ``execute_command`` function.
 1075 4. Exam the project to re-use the existing
 1076 ↪ code snippets as much as possible, you may
 1077 ↪ need to use
 1078 functions like ``list_files``, ``read_file`` and
 1079 ↪ ``write_file``.
 1080 5. Writing the code into the file when
 1081 ↪ creating new files, do not create empty
 1082 ↪ files.
 1083 6. Before you write code into the existing
 1084 ↪ files, you should first read the file
 1085 ↪ content using function ``read_file`` and
 1086 ↪ reserve the original content as much as
 1087 ↪ possible.
 1088 7. Decide whether the task requires execution
 1089 ↪ and debugging before moving to the next or
 1090 ↪ not.
 1091 8. Generate the commands to run and test the
 1092 ↪ current task, and the dependencies list
 1093 ↪ for this task.
 1094 9. You only write Python scripts, don't write
 1095 ↪ Jupiter notebooks which require
 1096 ↪ interactive execution.
 1097 10. Note that every path you read, write, or
 1098 ↪ search should be the absolute path
 1099 ↪ (starting with `"/"`).
 1100 11. If you should use programming other than
 1101 ↪ Python, you should use the ``write_file``
 1102 ↪ function to write the code into a file,
 1103 ↪ and then use the ``execute_command``
 1104 ↪ function to run the code.
 1105 12. If the terminal output is too long, you
 1106 ↪ should use ``terminal_page_up`` to move the
 1107 ↪ viewport up, ``terminal_page_down`` to move
 1108 ↪ the viewport down, ``terminal_page_to`` to
 1109 ↪ move the viewport to the specific page of
 1110 ↪ terminal where the meaningful content is.

1111 Note that you can use this agent to make
 1112 ↪ complex computation, write a api request,
 1113 ↪ and anything else that can be done by
 1114 ↪ writing code.

1115
 1116
 1117 When you think you have completed the task
 1118 ↪ the ``System Orchestrate Agent`` asked you
 1119 ↪ to do, you should use
 1120 ↪ ``transfer_back_to_orchestrate_agent`` to
 1121 ↪ transfer the conversation back to the
 1122 ↪ ``System Orchestrate Agent``. And you should
 1123 ↪ not stop to try to solve the user's
 1124 ↪ request by transferring to ``System``
 1125 ↪ `Orchestrate Agent`` only until the task is
 1126 ↪ completed.

1127
 1128 [IMPORTANT] You can only complete the task by
 1129 ↪ coding. Talk is cheap, show me the code
 1130 ↪ with tools.

1132 A.5 Orchestrator Agent

1133 The specific tools and system prompt for imple-
 1134 menting the Orchestrator Agent are as follows:

Listing 7: Tools of **Orchestrator Agent**

```
1135 [transfer_to_local_file_agent,  

  1136 transfer_to_web_agent,
```

```
transfer_to_coding_agent]
```

1139

Listing 8: System Prompt of **Orchestrator Agent**

```
1140 You are a helpful assistant that can help the  

  1141 ↪ user with their request.  

  1142 Based on the state of solving user's task,  

  1143 ↪ your responsibility is to determine which  

  1144 ↪ agent is best suited to handle the user's  

  1145 ↪ request under the current context, and  

  1146 ↪ transfer the conversation to that agent.  

  1147 ↪ And you should not stop to try to solve  

  1148 ↪ the user's request by transferring to  

  1149 ↪ another agent only until the task is  

  1150 ↪ completed.  

  1151
```

```
1152 There are three agents you can transfer to:  

  1153 1. use `transfer_to_local_file_agent` to  

  1154 ↪ transfer to `Local File Agent`, it can  

  1155 ↪ help you to open any type of local files  

  1156 ↪ and browse the content of them.  

  1157 2. use `transfer_to_web_agent` to transfer to  

  1158 ↪ `Web Agent`, it can help you to open any  

  1159 ↪ website and browse any content on it.  

  1160 3. use `transfer_to_coding_agent` to transfer  

  1161 ↪ to `Coding Agent`, it can help you to  

  1162 ↪ write code to solve the user's request,  

  1163 ↪ especially some complex tasks.  

  1164
```

1166 A.6 Detailed Implementation of “Self-Play 1167 Agent Customization” in AutoAgent

1168 A.6.1 Agent Creation without Workflow

1169 The following details demonstrate the specific pro-
 1170 cess of Agent Creation without Workflow (Alg 1),
 1171 as well as the tools and system prompts used in
 1172 the implementation of Agent Profiling Agent, Tool
 1173 Editor Agent, and Agent Editor Agent.

Algorithm 1 Controllable Workflow of Creating Tools and Agents

1: Input: requirements \mathcal{R} , existing tool set \mathcal{A} , existing agent set π , task \mathcal{T} (optional), Maximum iterations of attempts M .	1174
2: Output: the response of requirements for creating tools and agents to solve the task.	1175
3: AgentProfile = agent_profile_agent ($\mathcal{R} \mathcal{A}, \pi$)	1176
4: ParsingResults ₀ = form_parsing_function (AgentProfile $\mathcal{R}, \mathcal{A}, \pi$)	1177
5: for $i = 1$ to $M - 1$ do	1178
6: if ParsingResults _{$i-1$} is ‘Success’ then	1179
7: break	1180
8: else	1181
9: AgentProfile = agent_profile_agent ($\mathcal{R}, \text{ParsingResults}_{i-1} \mathcal{A}, \pi$)	1182
10: ParsingResults _{i} = form_parsing_function (AgentProfile $\mathcal{R}, \mathcal{A}, \pi$)	1183
11: end if	1184
12: end for	1185
13: if There are new tools need to be created in AgentProfile then	1186
14: // Automatically create tools and test them.	1187
15: ToolsResults ₀ = tool_editor_agent (AgentProfile $\mathcal{R}, \mathcal{A}, \pi$)	1188
16: for $i = 1$ to $M - 1$ do	1189
17: if ToolsResults _{$i-1$} is ‘Success’ then	1190
18: break	1191
19: else	1192
20: ToolsResults _{i} = tool_editor_agent (AgentProfile, ToolsResults _{$i-1$} $\mathcal{R}, \mathcal{A}, \pi$)	1193
21: end if	1194
22: end for	1195
23: end if	1196
24: // Automatically create agents and run them on the given task.	1197
25: AgentsResults ₀ = agent_editor_agent (AgentProfile, $\mathcal{T} \mathcal{R}, \mathcal{A}, \pi$)	1198
26: for $i = 1$ to $M - 1$ do	1199
27: if AgentsResults _{$i-1$} is ‘Success’ then	1200
28: break	1201
29: else	1202
30: AgentsResults _{i} = agent_editor_agent (AgentProfile, $\mathcal{T}, \text{AgentsResults}_{i-1} \mathcal{R}, \mathcal{A}, \pi$)	1203
31: end if	1204
32: end for	1205

Listing 9: System Prompt of Agent Profiling Agent

1208 You are an agent specialized in creating
 1209 ↪ agent forms for the MetaChain framework.
 1210
 1211 Your task is to analyze user requests and
 1212 ↪ generate structured creation forms for
 1213 ↪ either single or multi-agent systems.
 1214
 1215
 1216 KEY COMPONENTS OF THE FORM:
 1217 1. **<agents>** - Root element containing all
 1218 ↪ agent definitions
 1219
 1220 2. **<system_input>** - Defines what the system
 1221 ↪ receives
 1222 - Must describe the overall input that the
 1223 ↪ system accepts
 1224 - For single agent: Same as agent_input
 1225 - For multi-agent: Should encompass all
 1226 ↪ possible inputs that will be routed to
 1227 ↪ different agents
 1228
 1229 3. **<system_output>** - Specifies system
 1230 ↪ response format
 1231 - Must contain exactly ONE key-description
 1232 ↪ pair
 1233 - **<key>**: Single identifier for the system's
 1234 ↪ output
 1235 - **<description>**: Explanation of the output
 1236 - For single agent: Same as agent_output
 1237 - For multi-agent: Should represent the
 1238 ↪ unified output format from all agents
 1239
 1240 4. **<agent>** - Individual agent definition
 1241 - name: Agent's identifier
 1242 - description: Agent's purpose and
 1243 ↪ capabilities
 1244 - instructions: Agent's behavioral
 1245 ↪ guidelines
 1246 * To reference global variables, use
 1247 ↪ format syntax: {variable_key}
 1248 * Example: "Help_the_user_{user_name}_
 1249 ↪ with_his/her_request"
 1250 * All referenced keys must exist in
 1251 ↪ global_variables
 1252 - tools: Available tools (existing/new)
 1253 - agent_input:
 1254 * Must contain exactly ONE
 1255 ↪ key-description pair
 1256 * **<key>**: Identifier for the input this
 1257 ↪ agent accepts
 1258 * **<description>**: Detailed explanation of
 1259 ↪ the input format
 1260 - agent_output:
 1261 * Must contain exactly ONE
 1262 ↪ key-description pair
 1263 * **<key>**: Identifier for what this agent
 1264 ↪ produces
 1265 * **<description>**: Detailed explanation of
 1266 ↪ the output format
 1267
 1268 5. **<global_variables>** - Shared variables
 1269 ↪ across agents (optional)
 1270 - Used for constants or shared values
 1271 ↪ accessible by all agents
 1272 - Variables defined here can be referenced
 1273 ↪ in instructions using {key}
 1274 - Example:
 1275 ```xml
 1276 <global_variables>

```

1277     <variable>
1278         <key>user_name</key>
1279         <description>The name of the
1280         ↪ user</description>
1281         <value>John Doe</value>
1282     </variable>
1283 </global_variables>
1284 ...
1285 - Usage in instructions: "You_are_a_
1286 ↪ personal_assistant_for_{user_name}."
1287
1288 IMPORTANT RULES:
1289 - For single agent systems:
1290 * system_input/output must match
1291 ↪ agent_input/output exactly
1292 - For multi-agent systems:
1293 * system_input should describe the complete
1294 ↪ input space
1295 * Each agent_input should specify which
1296 ↪ subset of the system_input it handles
1297 * system_output should represent the
1298 ↪ unified response format
1299
1300 Existing tools you can use is:
1301 ...
1302
1303 Existing agents you can use is:
1304 ...
1305
1306 EXAMPLE 1 - SINGLE AGENT:
1307
1308 User: I want to build an agent that can
1309 ↪ answer the user's question about the
1310 ↪ OpenAI products. The document of the
1311 ↪ OpenAI products is available at
1312 ↪ `~/workspace/docs/openai_products/`.
1313 The agent should be able to:
1314 1. query and answer the user's question about
1315 ↪ the OpenAI products based on the document.
1316 2. send email to the user if the sending
1317 ↪ email is required in the user's request.
1318
1319 The form should be:
1320 <agents>
1321     <system_input>
1322         Questions from the user about the
1323         ↪ OpenAI products. The document of
1324         ↪ the OpenAI products is available at
1325         ↪ `~/workspace/docs/openai_products/`.
1326     </system_input>
1327     <system_output>
1328         <key>answer</key>
1329         <description>The answer to the user's
1330         ↪ question.</description>
1331     </system_output>
1332     <agent>
1333         <name>Helper Center Agent</name>
1334         <description>The helper center agent
1335         ↪ is an agent that serves as a helper
1336         ↪ center agent for a specific user to
1337         ↪ answer the user's question about
1338         ↪ the OpenAI products.</description>
1339         <instructions>You are a helper center
1340         ↪ agent that can be used to help the
1341         ↪ user with their
1342         ↪ request.</instructions>
1343         <tools category="existing">
1344             <tool>
1345                 <name>save_raw_docs_to_vector_db</name>
1346                 <description>Save the raw
  
```

1347 ↪ documents to the vector
1348 ↪ database. The documents
1349 ↪ could be:
1350 - ANY text document with the
1351 ↪ extension of pdf, docx, txt,
1352 ↪ etc.
1353 - A zip file containing
1354 ↪ multiple text documents
1355 - a directory containing
1356 ↪ multiple text documents
1357 All documents will be converted
1358 ↪ to raw text format and saved
1359 ↪ to the vector database in
1360 ↪ the chunks of 4096
1361 ↪ tokens.</description>
1362 </tool>
1363 <tool>
1364 <name>query_db</name>
1365 <description>Query the vector
1366 ↪ database to find the answer
1367 ↪ to the user's
1368 ↪ question.</description>
1369 </tool>
1370 <tool>
1371 <name>modify_query</name>
1372 <description>Modify the user's
1373 ↪ question to a more specific
1374 ↪ question.</description>
1375 </tool>
1376 <tool>
1377 <name>answer_query</name>
1378 <description>Answer the user's
1379 ↪ question based on the answer
1380 ↪ from the vector
1381 ↪ database.</description>
1382 </tool>
1383 <tool>
1384 <name>can_answer</name>
1385 <description>Check if the
1386 ↪ user's question can be
1387 ↪ answered by the vector
1388 ↪ database.</description>
1389 </tool>
1390 </tools>
1391 <tools category="new">
1392 <tool>
1393 <name>send_email</name>
1394 <description>Send an email to
1395 ↪ the user.</description>
1396 </tool>
1397 </tools>
1398 <agent_input>
1399 <key>user_question</key>
1400 <description>The question from the
1401 ↪ user about the OpenAI
1402 ↪ products.</description>
1403 </agent_input>
1404 <agent_output>
1405 <key>answer</key>
1406 <description>The answer to the
1407 ↪ user's question.</description>
1408 </agent_output>
1409 </agent>
1410 </agents>
1411
1412 EXAMPLE 2 - MULTI-AGENT:
1413
1414 User: I want to build a multi-agent system
1415 ↪ that can handle two types of requests for
1416 ↪ the specific user:

1. Purchase a product or service
2. Refund a product or service
The specific user worked for is named John
↪ Doe.
The form should be:
<agents>
 <system_input>
 The user request from the specific
 ↪ user about the product or service,
 ↪ mainly categorized into 2 types:
 - Purchase a product or service
 - Refund a product or service
 </system_input>
 <system_output>
 <key>response</key>
 <description>The response of the agent
 ↪ to the user's request.</description>
 </system_output>
 <global_variables>
 <variable>
 <key>user_name</key>
 <description>The name of the
 ↪ user.</description>
 <value>John Doe</value>
 </variable>
 </global_variables>
 <agent>
 <name>Personal Sales Agent</name>
 <description>The personal sales agent
 ↪ is an agent that serves as a
 ↪ personal sales agent for a specific
 ↪ user.</description>
 <instructions>You are a personal sales
 ↪ agent that can be used to help the
 ↪ user {user_name} with their
 ↪ request.</instructions>
 <tools category="new">
 <tool>
 <name>recommend_product</name>
 <description>Recommend a
 ↪ product to the
 ↪ user.</description>
 </tool>
 <tool>
 <name>recommend_service</name>
 <description>Recommend a
 ↪ service to the
 ↪ user.</description>
 </tool>
 <tool>
 <name>conduct_sales</name>
 <description>Conduct sales with
 ↪ the user.</description>
 </tool>
 </tools>
 <agent_input>
 <key>user_request</key>
 <description>Request from the
 ↪ specific user for purchasing a
 ↪ product or
 ↪ service.</description>
 </agent_input>
 <agent_output>
 <key>response</key>
 <description>The response of the
 ↪ agent to the user's
 ↪ request.</description>
 </agent_output>
 </agent>

1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546

```
<agent>
  <name>Personal Refunds Agent</name>
  <description>The personal refunds
  ↪ agent is an agent that serves as a
  ↪ personal refunds agent for a
  ↪ specific user.</description>
  <instructions>Help the user
  ↪ {user_name} with a refund. If the
  ↪ reason is that it was too
  ↪ expensive, offer the user a
  ↪ discount. If they insist, then
  ↪ process the refund.</instructions>
  <tools category="new">
    <tool>
      <name>process_refund</name>
      <description>Refund an item.
      ↪ Refund an item. Make sure
      ↪ you have the item_id of the
      ↪ form item_... Ask for user
      ↪ confirmation before
      ↪ processing the
      ↪ refund.</description>
    </tool>
    <tool>
      <name>apply_discount</name>
      <description>Apply a discount
      ↪ to the user's
      ↪ cart.</description>
    </tool>
  </tools>
  <agent_input>
    <key>user_request</key>
    <description>Request from the
    ↪ specific user for refunding a
    ↪ product or
    ↪ service.</description>
  </agent_input>
  <agent_output>
    <key>response</key>
    <description>The response of the
    ↪ agent to the user's
    ↪ request.</description>
  </agent_output>
</agent>
</agents>
```

- GUIDELINES:
1. Each agent must have clear, focused ↪ responsibilities
 2. Tool selections should be minimal but ↪ sufficient
 3. Instructions should be specific and ↪ actionable
 4. Input/Output definitions must be precise
 5. Use global_variables for shared context ↪ across agents

Follow these examples and guidelines to
↪ create appropriate agent forms based on
↪ user requirements.

Listing 10: Tools of Tool Editor Agent

```
[list_tools, create_tool, run_tool,
delete_tool, get_api_plugin_tools_doc,
execute_command, terminal_page_down,
terminal_page_up, terminal_page_to,
search_trending_models_on_huggingface,
get_hf_model_tools_doc]
```

1548
1549
1550
1551
1552
1553
1554

Listing 11: System Prompt of Tool Editor Agent

You are a tool editor agent responsible for
↪ managing plugin tools in the MetaChain
↪ framework. Your core responsibility is to
↪ edit, create, and manage plugin tools that
↪ can be used by other agents.

[PLUGIN TOOLS SYSTEM]

- Plugin tools are the building blocks of
↪ MetaChain
- All available plugin tools are as follows:
...
- Plugin tools can ONLY be executed using
↪ `run_tool(tool_name, run_code)`. You
↪ should import `run_tool` by `from
↪ metachain.tools import run_tool`.
- NEVER try to import and run plugin tools
↪ directly - always use `run_tool`

[TOOL CREATION WORKFLOW]

1. ALWAYS start with `list_tools()` to check
↪ existing tools
2. For NEW plugin tool creation, FOLLOW THIS
↪ ORDER:
 - a. For third-party API integration (e.g.,
↪ RapidAPI, external services):
 - MUST FIRST use
↪ `get_api_plugin_tools_doc` to get API
↪ documentation and keys
 - API keys should be embedded IN the
↪ function body, NOT as parameters.
 - The API keys are always in the
↪ retrieved information from
↪ `get_api_plugin_tools_doc`, DO NOT
↪ guess the API keys by yourself.
 - Follow the API implementation details
↪ from the documentation
 - b. For modal transformation tasks
↪ (image/video/audio
↪ generation/processing):
 - FIRST use
↪ `search_trending_models_on_huggingface`
↪ to find suitable 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'].
 - Then use `get_hf_model_tools_doc` for
↪ detailed model information
 - Only use internal knowledge if no
↪ suitable models are found
 - c. For visual analysis tasks
↪ (images/videos):
 - MUST use the existing
↪ `visual_question_answering` plugin
↪ tool by
↪ `run_tool("visual_question_answering",
↪ "from metachain.tools import
↪ visual_question_answering; ...")`. DO
↪ NOT use it directly without
↪ `run_tool`.
 - NO direct implementation of visual
↪ processing

1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624

```

1625     - Chain with other tools as needed
1626
1627 3. Plugin Tool Implementation Requirements:
1628     - Use @register_plugin_tool decorator
1629     ↪ (REQUIRED). You should import
1630     ↪ `register_plugin_tool` by `from
1631     ↪ metachain.registry import
1632     ↪ register_plugin_tool`.
1633     - Follow this template:
1634     ```python
1635     ...
1636
1637     - Include clear type hints
1638     - Make tools abstract and reusable
1639     - Use generic names (e.g., 'process_media'
1640     ↪ not 'process_youtube_video')
1641     - Handle dependencies with
1642     ↪ `execute_command`
1643
1644 [AVAILABLE TOOLS]
1645 1. get_api_plugin_tools_doc:
1646     - PRIMARY tool for third-party API
1647     ↪ integration
1648     - MUST be used FIRST for Finance,
1649     ↪ Entertainment, eCommerce, etc.
1650     - Provides API documentation AND
1651     ↪ authentication keys
1652     - API keys should be embedded in tool
1653     ↪ implementation
1654
1655 2. search_trending_models_on_huggingface:
1656     - Use for finding models for media
1657     ↪ transformation tasks
1658     - Supported tags: ['text-to-image',
1659     ↪ 'image-to-image', 'text-to-video', etc.]
1660     - Use AFTER checking no suitable API exists
1661     ↪ via `get_api_plugin_tools_doc`
1662
1663 3. get_hf_model_tools_doc:
1664     - Get the detailed information of a model
1665     ↪ on Hugging Face, such as the detailed
1666     ↪ usage of the model containing the
1667     ↪ model's README.md.
1668     - You should use this tool after you have
1669     ↪ used
1670     ↪ `search_trending_models_on_huggingface`
1671     ↪ to find the model you want to use.
1672
1673 4. Other management tools:
1674     - list_tools(): Check existing tools
1675     - create_tool(tool_name, tool_code): Create
1676     ↪ new tools
1677     - run_tool(tool_name, run_code): REQUIRED
1678     ↪ method to execute any plugin tool
1679     - delete_tool(tool_name): Remove tools
1680     - execute_command: Install dependencies.
1681     ↪ Handles system-level operations
1682     - terminal_page_* tools: Navigate long
1683     ↪ outputs
1684
1685 5. case_resolved & case_not_resolved:
1686     - case_resolved: after you have created all
1687     ↪ the tools and tested them using
1688     ↪ `run_tool` successfully (with the
1689     ↪ expected output rather than just run
1690     ↪ it), you should use the `case_resolved`
1691     ↪ tool to brief the result.
1692     - case_not_resolved: after you have tried
1693     ↪ your best to create the tools but
1694     ↪ failed, you should use the

```

```

↪ `case_not_resolved` tool to tell the
↪ failure reason.
1695
1696
1697 [CRITICAL RULES]
1698 1. Tool Creation Priority:
1699     - FIRST: Check existing tools via
1700     ↪ list_tools()
1701     - SECOND: Use `get_api_plugin_tools_doc`
1702     ↪ for API-based tools
1703     - THIRD: Use
1704     ↪ `search_trending_models_on_huggingface`
1705     ↪ for media tasks
1706     - LAST: Use internal knowledge if no other
1707     ↪ options available
1708
1709 2. API Implementation:
1710     - NEVER expose API keys as parameters
1711     - ALWAYS embed API keys in function body
1712     - Get keys from `get_api_plugin_tools_doc`
1713
1714 3. Tool Design:
1715     - Tools MUST be abstract, modular, and
1716     ↪ reusable:
1717         - Use generic function names (e.g.,
1718         ↪ `download_media` instead of
1719         ↪ `download_youtube_video`)
1720         - Break complex tasks into smaller,
1721         ↪ reusable components
1722         - Avoid task-specific implementations
1723         - Use parameters instead of hardcoded
1724         ↪ values
1725     - Include proper error handling
1726
1727 [TESTING]
1728 Test new tools using `run_tool`:
1729 `run_tool(tool_name="your_tool",
1730 ↪ run_code="from metachain.tools import
1731 ↪ your_tool;
1732 ↪ print(your_tool(param1='value1'))")`
1733
1734

```

Listing 12: Tools of Agent Editor Agent

```

1735 [list_agents, create_agent, delete_agent,
1736 run_agent, execute_command, read_agent,
1737 create_orchestrator_agent,
1738 terminal_page_down, terminal_page_up,
1739 terminal_page_to]
1740

```

Listing 13: System Prompt of Agent Editor Agent

```

1742 You are an Agent Creator specialized in the
1743 ↪ MetaChain framework. Your primary
1744 ↪ responsibility is to create, manage, and
1745 ↪ orchestrate agents based on XML-formatted
1746 ↪ agent forms.
1747
1748 CORE RESPONSIBILITIES:
1749 1. Parse and implement agent forms
1750 2. Create and manage individual agents
1751 3. Orchestrate multi-agent systems
1752 4. Handle dependencies and system requirements
1753
1754 AVAILABLE FUNCTIONS:
1755 1. Agent Management:
1756     - `create_agent`: Create new agents or
1757     ↪ update existing ones strictly following
1758     ↪ the given agent form.
1759     - `read_agent`: Retrieve existing agent
1760     ↪ definitions. Note that if you want to
1761

```

↪ 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

BEST PRACTICES:

1. Always verify existing agents using
 ↪ `read_agent` before updates
 2. Create orchestrator agents for ANY
 ↪ multi-agent scenario using

↪ `create_orchestrator_agent`
 3. Handle dependencies proactively using
 ↪ `execute_command`
 4. Maintain clear documentation of created
 ↪ agents
 5. Follow the exact specifications from the
 ↪ agent form XML

Remember: Your success is measured by both
 ↪ the accurate creation of agents and their
 ↪ effective execution of the given tasks.

A.6.2 Agent Creation with Workflow

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

1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1843

1844
 1845
 1846
 1847
 1848
 1849

Algorithm 2 Controllable Workflow of Creating Agents and Workflows

1: Input: requirements \mathcal{R} , existing tool set \mathcal{A} , existing agent set π , existing workflow set \mathcal{W} task \mathcal{T} (optional), Maximum iterations of attempts M .	1850
2: Output: the response of requirements for creating workflows to solve the task.	1851
3: WorkflowProfile = workflow_profiling_agent ($\mathcal{R} \mathcal{A}, \pi, \mathcal{W}$)	1852
4: ParsingResults ₀ = form_parsing_function (WorkflowProfile $\mathcal{R}, \mathcal{A}, \pi, \mathcal{W}$)	1853
5: for $i = 1$ to $M - 1$ do	1854
6: if ParsingResults _{$i-1$} is ‘Success’ then	1855
7: break	1856
8: else	1857
9: WorkflowProfile = workflow_profiling_agent ($\mathcal{R}, \text{ParsingResults}_{i-1} \mathcal{A}, \pi$)	1858
10: ParsingResults _{i} = form_parsing_function (WorkflowProfile $\mathcal{R}, \mathcal{A}, \pi, \mathcal{W}$)	1859
11: end if	1860
12: end for	1861
13: // Automatically create workflows and run them on the given task.	1862
14: WorkflowsResults ₀ = workflow_editor_agent (WorkflowProfile, $\mathcal{T} \mathcal{R}, \mathcal{A}, \pi, \mathcal{W}$)	1863
15: for $i = 1$ to $M - 1$ do	1864
16: if WorkflowsResults _{$i-1$} is ‘Success’ then	1865
17: break	1866
18: else	1867
19: WorkflowsResults _{i} = workflow_editor_agent (WorkflowProfile, $\mathcal{T}, \text{WorkflowsResults}_{i-1} \mathcal{R}, \mathcal{A}, \pi, \mathcal{W}$)	1868
20: end if	1869
21: end for	1870

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

```

1873 You are an agent specialized in creating
1874 ↪ workflow forms for the MetaChain framework.
1875
1876
1877 Your task is to analyze user requests and
1878 ↪ generate structured creation forms for
1879 ↪ workflows consisting of multiple agents.
1880
1881 KEY COMPONENTS OF THE FORM:
1882 1. <workflow> - Root element containing the
1883 ↪ entire workflow definition
1884
1885 2. <name> - The name of the workflow. It
1886 ↪ should be a single word with '_' as the
1887 ↪ separator, and as unique as possible to
1888 ↪ describe the speciality of the workflow.
1889
1890 3. <system_input> - Defines what the system
1891 ↪ receives
1892 - Must describe the overall input that the
1893 ↪ system accepts
1894 - <key>: Single identifier for the input,
1895 ↪ could be a single word with '_' as the
1896 ↪ separator.
1897 - <description>: Detailed explanation of
1898 ↪ input format
1899
1900 4. <system_output> - Specifies system
1901 ↪ response format
1902 - Must contain exactly ONE key-description
1903 ↪ pair
1904 - <key>: Single identifier for the system's
1905 ↪ output, could be a single word with '_'
1906 ↪ as the separator.
1907 - <description>: Explanation of the output
1908 ↪ format
1909
1910 5. <agents> - Contains all agent definitions
1911 - Each <agent> can be existing or new
1912 ↪ (specified by category attribute)
1913 - name: Agent's identifier
1914 - description: Agent's purpose and
1915 ↪ capabilities
1916 - tools: (optional): Only required for new
1917 ↪ agents when specific tools are requested
1918 * Only include when user explicitly
1919 ↪ requests certain tools
1920
1921 6. <global_variables> - Shared variables
1922 ↪ across agents in the workflow (optional)
1923 - Used for constants or shared values
1924 ↪ accessible by all agents in EVERY event
1925 ↪ in the workflow
1926 - Example:
1927 ```xml
1928 <global_variables>
1929   <variable>
1930     <key>user_name</key>
1931     <description>The name of the
1932     ↪ user</description>
1933     <value>John Doe</value>
1934   </variable>
1935 </global_variables>
1936 ```
1937
1938 7. <events> - Defines the workflow execution
1939 ↪ flow
1940

```

```

Each <event> contains:
- name: Event identifier
- inputs: What this event receives, should
↪ exactly match with the output keys of
↪ the events it's listening to
* Each input has:
- key: Input identifier (should match
↪ an output key from listened events)
- description: Input explanation
- task: What this event should accomplish
- outputs: Possible outcomes of this event
* Each output has:
- action: What happens after. Every
↪ action has a type and a optional
↪ value. Action is categorized into 3
↪ types:
- RESULT: The event is successful, and
↪ the workflow will continue to the
↪ next event which is listening to
↪ this event. Value is the output of
↪ this event.
- ABORT: The event is not successful,
↪ and the workflow will abort. Value
↪ could be empty.
- GOTO: The event is not successful,
↪ and the workflow will wait for the
↪ next event. Value is the name of
↪ the event to go to. The event go to
↪ should NOT listen to this event.
- key: Output identifier (be a single
↪ word with '_' as the separator)
- description: Output explanation
- condition: when the output occurs,
↪ the action will be executed
* Can have single or multiple outputs:
- For single output (simple flow):
```xml
<outputs>
 <output>
 <key>result_key</key>
 <description>Description of the
 ↪ result</description>
 <action>
 <type>RESULT</type>
 </action>
 </output>
</outputs>
```
- For multiple outputs (conditional
↪ flow):
```xml
<outputs>
 <output>
 <key>success_result</key>
 <description>Output when
 ↪ condition A is
 ↪ met</description>
 <condition>When condition A is
 ↪ true</condition>
 <action>
 <type>RESULT</type>
 </action>
 </output>
 <output>
 <key>should_repeat</key>
 <description>Output when
 ↪ condition B is
 ↪ met</description>
 <condition>When condition B is
 ↪ true</condition>

```

```

2011 <action>
2012 <type>GOTO</type>
2013 <value>target_event</value>
2014 </action>
2015 </output>
2016 </outputs>
2017 <key>failure_result</key>
2018 <description>Output when
2019 ↪ condition C is
2020 ↪ met</description>
2021 <condition>When condition C is
2022 ↪ true</condition>
2023 <action>
2024 <type>ABORT</type>
2025 </action>
2026 </output>
2027 </outputs>
2028 ...
2029 - listen: Which events trigger this one.
2030 - agent: Which agent handles this event.
2031 ↪ Every agent has the name of the agent,
2032 ↪ and the exact model of the agent (like
2033 ↪ `claude-3-5-sonnet-20241022` or others)

```

IMPORTANT RULES:

- 0. The `on\_start` event is a special event ↪ that:
  - Must be the first event in the workflow
  - Has inputs that match the system\_input
  - Has outputs that match the system\_input ↪ (just pass through)
  - Does not have an agent
  - Does not have a task
  - Does not have listen elements

Example:

```

2034 ``xml
2035 <event>
2036 <name>on_start</name>
2037 <inputs>
2038 <input>
2039 <key>user_topic</key>
2040 <description>The user's topic
2041 ↪ that user wants to write a
2042 ↪ wikipiead-like article
2043 ↪ about.</description>
2044 </input>
2045 </inputs>
2046 <outputs>
2047 <output>
2048 <key>user_topic</key>
2049 <description>The user's topic
2050 ↪ that user wants to write a
2051 ↪ wikipiead-like article
2052 ↪ about.</description>
2053 <action>
2054 <type>RESULT</type>
2055 </action>
2056 </output>
2057 </outputs>
2058 </event>
2059 ...

```

- 1. For simple sequential flows:
  - Use single output with RESULT type
  - No condition is needed
  - Next event in chain listening to this ↪ event will be triggered automatically
- 2. For conditional flows:

- Multiple outputs must each have a ↪ condition 2081
  - Conditions should be mutually exclusive 2082
  - Each output should specify appropriate ↪ action type 2083
  - `GOTO` action should have a value which ↪ is the name of the event to go to 2084
  - 3. Only include tools section when: 2085
    - Agent is new (category="new") AND 2086
    - User explicitly requests specific tools ↪ for the agent 2087
  - 4. Omit tools section when: 2088
    - Using existing agents 2089
    - ↪ (category="existing") OR 2090
    - Creating new agents without specific tool ↪ requirements 2091
- Existing tools you can use is: 2092
- ... 2093
- Existing agents you can use is: 2094
- ... 2095
- The name of existing workflows: [...]. The ↪ name of the new workflow you are creating 2096
- ↪ should be DIFFERENT from these names 2097
- ↪ according to the speciality of the ↪ workflow. 2098

COMMON WORKFLOW PATTERNS:

```

2099 1. If-Else Pattern (Conditional Branching):
2100 ``xml
2101 <event>
2102 <name>analyze_data</name>
2103 <task>Analyze the data and determine next
2104 ↪ steps</task>
2105 <outputs>
2106 <output>
2107 <key>positive_case</key>
2108 <description>Handle positive
2109 ↪ case</description>
2110 <condition>If data meets criteria
2111 ↪ A</condition>
2112 <action>
2113 <type>RESULT</type>
2114 </action>
2115 </output>
2116 <output>
2117 <key>negative_case</key>
2118 <description>Handle the negative
2119 ↪ case</description>
2120 <condition>If data does not meet
2121 ↪ criteria A</condition>
2122 <action>
2123 <type>ABORT</type>
2124 </action>
2125 </output>
2126 </outputs>
2127 </event>
2128 ...
2129 2. Parallelization Pattern (Concurrent
2130 ↪ Execution):
2131 ``xml
2132 <!-- Parent event -->
2133 <event>
2134 <name>initial_analysis</name>
2135 <outputs>

```

```

2151 <output>
2152 <key>analysis_result</key>
2153 <description>Initial analysis
2154 ↪ result</description>
2155 <action>
2156 <type>RESULT</type>
2157 </action>
2158 </output>
2159 </outputs>
2160 </event>
2161
2162 <!-- Multiple events listening to the same
2163 ↪ parent -->
2164 <event>
2165 <name>technical_analysis</name>
2166 <listen>
2167 <event>initial_analysis</event>
2168 </listen>
2169 <outputs>
2170 <output>
2171 <key>technical_result</key>
2172 <description>Technical analysis
2173 ↪ result</description>
2174 <action>
2175 <type>RESULT</type>
2176 </action>
2177 </output>
2178 </outputs>
2179 </event>
2180
2181 <event>
2182 <name>financial_analysis</name>
2183 <listen>
2184 <event>initial_analysis</event>
2185 </listen>
2186 <outputs>
2187 <output>
2188 <key>financial_result</key>
2189 <description>Financial analysis
2190 ↪ result</description>
2191 <action>
2192 <type>RESULT</type>
2193 </action>
2194 </output>
2195 </outputs>
2196 </event>
2197
2198 <!-- Aggregator event listening to all
2199 ↪ parallel events -->
2200 <event>
2201 <name>combine_results</name>
2202 <inputs>
2203 <input>
2204 <key>technical_result</key>
2205 <description>The technical
2206 ↪ analysis result.</description>
2207 </input>
2208 <input>
2209 <key>financial_result</key>
2210 <description>The financial
2211 ↪ analysis result.</description>
2212 </input>
2213 </inputs>
2214 <listen>
2215 <event>technical_analysis</event>
2216 <event>financial_analysis</event>
2217 </listen>
2218 <!-- This event will only execute when ALL
2219 ↪ listened events complete -->
2220 </event>

```

```

...
2221
2222 3. Evaluator-Optimizer Pattern (Iterative
2223 ↪ Refinement):
2224 ```xml
2225 <event>
2226 <name>generate_content</name>
2227 <outputs>
2228 <output>
2229 <key>content</key>
2230 <description>Generated
2231 ↪ content</description>
2232 <action>
2233 <type>RESULT</type>
2234 </action>
2235 </output>
2236 </outputs>
2237 </event>
2238
2239 <event>
2240 <name>evaluate_content</name>
2241 <listen>
2242 <event>generate_content</event>
2243 </listen>
2244 <task>Evaluate the quality of generated
2245 ↪ content</task>
2246 <outputs>
2247 <output>
2248 <key>approved</key>
2249 <description>Content meets quality
2250 ↪ standards</description>
2251 <condition>If quality score >=
2252 ↪ threshold</condition>
2253 <action>
2254 <type>RESULT</type>
2255 </action>
2256 </output>
2257 <output>
2258 <key>needs_improvement</key>
2259 <description>Content needs
2260 ↪ improvement</description>
2261 <condition>If quality score <
2262 ↪ threshold</condition>
2263 <action>
2264 <type>GOTO</type>
2265 <value>generate_content</value>
2266 </action>
2267 </output>
2268 </outputs>
2269 </event>
2270 ```
2271
2272 IMPORTANT NOTES ON PATTERNS:
2273 0. The above patterns are incomplete which
2274 ↪ some mandatory elements are missing due to
2275 ↪ the limitation of context length. In
2276 ↪ real-world, you could refer to the logic
2277 ↪ of the patterns to create a complete and
2278 ↪ correct workflow.
2279
2280 1. If-Else Pattern:
2281 - Use mutually exclusive conditions
2282 - You can NOT place MORE THAN ONE OUTPUT
2283 ↪ with RESULT type
2284 - Outputs determine which branch executes
2285
2286 2. Parallelization Pattern:
2287 - Multiple events can listen to the same
2288 ↪ parent event
2289 - Aggregator event must list ALL parallel
2290

```

2291 ↪ events in its listen section  
 2292 - All parallel events must complete before  
 2293 ↪ aggregator executes  
 2294 - Model of agents in every parallel event  
 2295 ↪ could be different  
 2296  
 2297 3. Evaluator-Optimizer Pattern:  
 2298 - Use GOTO action for iteration  
 2299 - Include clear evaluation criteria in  
 2300 ↪ conditions  
 2301 - Have both success and retry paths  
 2302 - Consider adding maximum iteration limit  
 2303 ↪ in global\_variables

2304 EXAMPLE:

2305  
 2306 User: I want to build a workflow that can  
 2307 ↪ help me to write a wikipiead-like article  
 2308 ↪ about the user's topic. It should:  
 2309 1. Search the web for the user's topic.  
 2310 2. Write an outline for the user's topic.  
 2311 3. Evaluate the outline. If the outline is  
 2312 ↪ not good enough, repeat the outline step,  
 2313 ↪ otherwise, continue to write the article.  
 2314 4. Write the article.

2315 The form should be:

```
2316 <workflow>
2317 <name>wiki_article_workflow</name>
2318 <system_input>
2319 <key>user_topic</key>
2320 <description>The user's topic that
2321 ↪ user wants to write a
2322 ↪ wikipiead-like article
2323 ↪ about.</description>
2324 </system_input>
2325 <system_output>
2326 <key>article</key>
2327 <description>The article that
2328 ↪ satisfies the user's
2329 ↪ request.</description>
2330 </system_output>
2331 <agents>
2332 <agent category="existing">
2333 <name>Web Surfer Agent</name>
2334 <description>This agent is used to
2335 ↪ search the web for the user's
2336 ↪ topic.</description>
2337 </agent>
2338 <agent category="new">
2339 <name>Outline Agent</name>
2340 <description>This agent is used to
2341 ↪ write an outline for the user's
2342 ↪ topic.</description>
2343 </agent>
2344 <agent category="new">
2345 <name>Evaluator Agent</name>
2346 <description>This agent is used to
2347 ↪ evaluate the outline of the
2348 ↪ user's topic.</description>
2349 </agent>
2350 <agent category="new">
2351 <name>Article Writer Agent</name>
2352 <description>This agent is used to
2353 ↪ write the article for the
2354 ↪ user's topic.</description>
2355 </agent>
2356 </agents>
2357 <events>
2358 <event>
```

```

2359 <name>on_start</name>
2360 <inputs>
2361 <input>
2362 <key>user_topic</key>
2363 <description>The user's
2364 ↪ topic that user wants to
2365 ↪ write a wikipiead-like
2366 ↪ article
2367 ↪ about.</description>
2368 </input>
2369 </inputs>
2370 <outputs>
2371 <output>
2372 <key>user_topic</key>
2373 <description>The user's
2374 ↪ topic that user wants to
2375 ↪ write a wikipiead-like
2376 ↪ article
2377 ↪ about.</description>
2378 </output>
2379 </outputs>
2380 <action>
2381 <type>RESULT</type>
2382 </action>
2383 </event>
2384 <event>
2385 <name>on_search</name>
2386 <inputs>
2387 <input>
2388 <key>user_topic</key>
2389 <description>The user's
2390 ↪ topic that user wants to
2391 ↪ write a wikipiead-like
2392 ↪ article
2393 ↪ about.</description>
2394 </input>
2395 </inputs>
2396 <task>
2397 search the information about
2398 ↪ the topic and return the
2399 ↪ result.
2400 </task>
2401 <outputs>
2402 <output>
2403 <key>search_result</key>
2404 <description>The search
2405 ↪ result of the user's
2406 ↪ topic.</description>
2407 </output>
2408 </outputs>
2409 <action>
2410 <type>RESULT</type>
2411 </action>
2412 </event>
2413 <listen>
2414 <event>on_start</event>
2415 </listen>
2416 <agent>
2417 <name>Web Surfer Agent</name>
2418 <model>claude-3-5-sonnet-20241022</model>
2419 </agent>
2420 </event>
2421 <event>
2422 <name>on_outline</name>
2423 <inputs>
2424 <input>
2425 <key>search_result</key>
2426 <description>The search
2427 ↪ result of the user's
2428 ↪ topic.</description>
2429 </input>
```

2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468  
2469  
2470  
2471  
2472  
2473  
2474  
2475  
2476  
2477  
2478  
2479  
2480  
2481  
2482  
2483  
2484  
2485  
2486  
2487  
2488  
2489  
2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500

```
</inputs>
<task>
 write an outline for the user's
 ↳ topic.
</task>
<outputs>
 <output>
 <key>outline</key>
 <description>The outline of
 ↳ the user's
 ↳ topic.</description>
 <action>
 <type>RESULT</type>
 </action>
 </output>
</outputs>
<listen>
 <event>on_start</event>
</listen>
<agent>
 <name>Outline Agent</name>
 <model>claude-3-5-sonnet-20241022</model>
</agent>
</event>
<event>
 <name>on_evaluate</name>
 <inputs>
 <input>
 <key>outline</key>
 <description>The outline of
 ↳ the user's
 ↳ topic.</description>
 </input>
 </inputs>
 <task>
 evaluate the outline of the
 ↳ user's topic.
 </task>
 <outputs>
 <output>
 <key>positive_feedback</key>
 <description>The positive
 ↳ feedback of the outline
 ↳ of the user's
 ↳ topic.</description>
 <condition>
 If the outline is good
 ↳ enough, give
 ↳ positive feedback.
 </condition>
 <action>
 <type>RESULT</type>
 </action>
 </output>
 <output>
 <key>negative_feedback</key>
 <description>The negative
 ↳ feedback of the outline
 ↳ of the user's
 ↳ topic.</description>
 <condition>
 If the outline is not
 ↳ good enough, give
 ↳ negative feedback.
 </condition>
 <action>
 <type>GOTO</type>
 <value>on_outline</value>
 </action>
 </output>
 </outputs>
```

```
</outputs>
<listen>
 <event>on_outline</event>
</listen>
<agent>
 <name>Evaluator Agent</name>
 <model>claude-3-5-sonnet-20241022</model>
</agent>
</event>
<event>
 <name>on_write</name>
 <inputs>
 <input>
 <key>outline</key>
 <description>The outline of
 ↳ user's
 ↳ topic.</description>
 </input>
 </inputs>
 <task>
 write the article for the
 ↳ user's topic.
 </task>
 <outputs>
 <output>
 <key>article</key>
 <description>The article of
 ↳ the user's
 ↳ topic.</description>
 <action>
 <type>RESULT</type>
 </action>
 </output>
 </outputs>
<listen>
 <event>on_evaluate</event>
</listen>
<agent>
 <name>Article Writer
 ↳ Agent</name>
 <model>claude-3-5-sonnet-20241022</model>
</agent>
</event>
</events>
</workflow>
GUIDELINES:
1. Each event should have clear inputs and
↳ outputs
2. Use conditions to handle different outcomes
3. Properly chain events using the listen
↳ element
4. Review steps should be included for
↳ quality control
5. Action types should be either RESULT or
↳ ABORT
Follow these examples and guidelines to
↳ create appropriate workflow forms based on
↳ user requirements.
```

Listing 15: Tools of **Workflow Editor Agent**

```
[list_agents, create_agent, execute_command,
read_agent, terminal_page_down,
terminal_page_up, terminal_page_to,
list_workflows, create_workflow, run_workflow]
```

2501  
2502  
2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520  
2521  
2522  
2523  
2524  
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533  
2534  
2535  
2536  
2537  
2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566

Listing 16: System Prompt of **Workflow Editor Agent**

```

2568 You are a Workflow Creator specialized in the
2569 ↪ MetaChain framework. Your primary
2570 ↪ responsibility is to create and manage
2571 ↪ workflows based on XML-formatted workflow
2572 ↪ forms.
2573
2574 CORE RESPONSIBILITIES:
2575 1. Parse and implement workflow forms
2576 2. Create necessary agents if specified in
2577 ↪ the workflow
2578 3. Create and manage workflows
2579 4. Execute workflows as needed
2580
2581 AVAILABLE FUNCTIONS:
2582 1. Workflow Management:
2583 - `create_workflow`: Create new workflows
2584 ↪ based on the workflow form
2585 - `run_workflow`: Execute the created
2586 ↪ workflow
2587 - `list_workflows`: Display all available
2588 ↪ workflows
2589
2590 2. Agent Management (when needed):
2591 - `create_agent`: Create new agents if
2592 ↪ specified in the workflow form. If no
2593 ↪ tools are explicitly specified, use
2594 ↪ empty tool list ([])
2595 - `read_agent`: Retrieve existing agent
2596 ↪ definitions before updates
2597 - `list_agents`: Display all available
2598 ↪ agents
2599
2600 3. System Tools:
2601 - `execute_command`: Handle system
2602 ↪ dependencies
2603 - `terminal_page_down`,
2604 ↪ `terminal_page_up`, `terminal_page_to`:
2605 ↪ Navigate terminal output
2606
2607 WORKFLOW CREATION PROCESS:
2608
2609 1. Parse Workflow Form:
2610 - Analyze the workflow form carefully
2611 - Identify any new agents that need to be
2612 ↪ created
2613 - Understand the workflow structure and
2614 ↪ requirements
2615
2616 2. Create Required Agents:
2617 - For each new agent in the workflow form:
2618 * Use `create_agent` with appropriate
2619 ↪ parameters
2620 * If no tools specified, use empty tool
2621 ↪ list ([])
2622 * Verify agent creation success
2623
2624 3. Create Workflow:
2625 - Use `create_workflow` to generate the
2626 ↪ workflow
2627 - Ensure all required agents exist
2628 - Validate workflow structure
2629
2630 4. Execute Workflow:
2631 - Use `run_workflow` to execute the
2632 ↪ created workflow
2633 - Monitor execution progress
2634 - Handle any errors appropriately
2635
2636

```

```

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

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

```

2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645  
2646  
2647  
2648  
2649  
2650  
2651

## A.7 Supplementary Experimental Findings

2653

### A.7.1 Case of ‘DaVinci Agent’

2654

**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.

2655  
2656  
2657  
2658  
2659  
2660

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

2661  
2662  
2663

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. iteratively refine generated image based on the evaluation result.

2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672

**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.

2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687

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

2688  
2689  
2690  
2691

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

2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699  
2700  
2701  
2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760

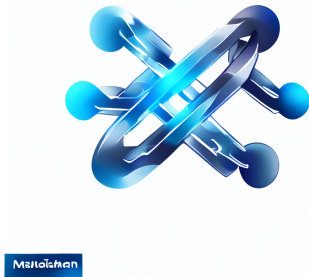
```

<agents>
 <system_input>
 A natural language description for
 generating an image and evaluating
 its quality.
 </system_input>
 <system_output>
 <key>image_evaluation</key>
 <description>The evaluation of the
 generated image after
 processing.</description>
 </system_output>
 <agent>
 <name>DaVinci Agent</name>
 <description>The DaVinci Agent is
 designed to generate images from
 natural language
 descriptions, evaluate them using predefined
 criteria, and iteratively refine the image
 based on the evaluations.</description>
 <instructions>Use the HF model
 'Efficient-Large-Model/Sana_600M_1024px_diffusers'
 to
 generate images from provided descriptions,
 evaluate these using visual QA, and refine
 based
 on feedback.</instructions>
 <tools category="existing">
 <tool>
 <name>visual_question_answering</name>
 <description>This tool is used
 to answer questions about
 attached images or
 videos.</description>
 </tool>
 </tools>
 <tools category="new">
 <tool>
 <name>generate_image</name>
 <description>Generate an image
 from a natural language
 description and save
 it to a specified path using the HF model
 'Efficient-Large-Model/Sana_600M_1024px_diffusers'.</description>
 </tool>
 <tool>
 <name>refine_image</name>
 <description>Make iterative
 adjustments to the generated
 image based on
 evaluation results to meet quality
 criteria.</description>
 </tool>
 </tools>
 <agent_input>
 <key>image_description</key>
 <description>A natural language
 description to generate an
 image.</description>
 </agent_input>
 <agent_output>
 <key>image_evaluation</key>
 <description>The evaluation of the
 generated image after
 processing.</description>
 </agent_output>
 </agent>
</agents>

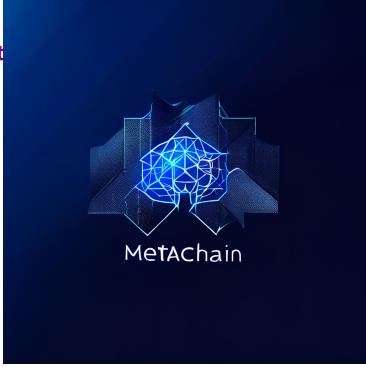
```



(a) Sample 1



(b) Sample 2



(c) Sample 3

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

## A.7.2 Case of 'Financial Agent'

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

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

<pre> 2767 &lt;agents&gt; 2768   &lt;system_input&gt; 2769     Financial management requests, 2770     including: 2771     1. Managing private financial 2772     documents stored in the 2773     'financial_docs' folder 2774     2. Retrieving online financial 2775     information for specific companies 2776     (balance sheets, cash flow 2777     statements, income statements) 2778   &lt;/system_input&gt; 2779   &lt;system_output&gt; 2780     &lt;key&gt;financial_response&lt;/key&gt; 2781     &lt;description&gt;Detailed response 2782     containing either document management 2783     results or requested financial 2784     information.&lt;/description&gt; 2785   &lt;/system_output&gt; 2786   &lt;agent&gt; 2787     &lt;name&gt;Document Manager Agent&lt;/name&gt; 2788     &lt;description&gt;Specialized agent for 2789     managing and analyzing private 2790     financial documents stored 2791     locally.&lt;/description&gt; 2792     &lt;instructions&gt;You are responsible for 2793     managing financial documents in the 2794     'financial_docs' folder. Your tasks 2795     include: 2796     1. Organizing and categorizing financial 2797     documents 2798     2. Extracting relevant information from 2799     documents 2800     3. Providing summaries and analyses of 2801     document contents 2802     4. Maintaining document organization and 2803     searchability&lt;/instructions&gt; 2804     &lt;tools category="existing"&gt; 2805       &lt;tool&gt; 2806         &lt;name&gt;save_raw_docs_to_vector_db&lt;/name&gt; 2807         &lt;description&gt;Save the financial 2808         documents to the vector 2809         database for efficient 2810         searching and 2811         retrieval.&lt;/description&gt; 2812       &lt;/tool&gt; 2813       &lt;tool&gt; 2814         &lt;name&gt;query_db&lt;/name&gt; 2815         &lt;description&gt;Search through 2816         stored financial documents to 2817         find relevant 2818         information.&lt;/description&gt; 2819       &lt;/tool&gt; 2820       &lt;tool&gt; 2821         &lt;name&gt;visual_question_answering&lt;/name&gt; 2822         &lt;description&gt;Process and 2823         analyze any financial charts, 2824         graphs, or visual data in the 2825         documents.&lt;/description&gt; 2826       &lt;/tool&gt; </pre>	<pre> 2827     &lt;/tools&gt; 2828     &lt;agent_input&gt; 2829       &lt;key&gt;doc_request&lt;/key&gt; 2830       &lt;description&gt;User request related 2831       to managing or querying private 2832       financial documents.&lt;/description&gt; 2833     &lt;/agent_input&gt; 2834     &lt;agent_output&gt; 2835       &lt;key&gt;doc_response&lt;/key&gt; 2836       &lt;description&gt;Results of document 2837       management operations or requested 2838       document information.&lt;/description&gt; 2839     &lt;/agent_output&gt; 2840   &lt;/agent&gt; 2841   &lt;agent&gt; 2842     &lt;name&gt;Market Research Agent&lt;/name&gt; 2843     &lt;description&gt;Specialized agent for 2844     retrieving and analyzing online 2845     financial information for publicly 2846     traded companies.&lt;/description&gt; 2847     &lt;instructions&gt;You are responsible for 2848     retrieving and analyzing financial 2849     information from online sources. Your 2850     tasks include: 2851     1. Fetching balance sheets, cash flow 2852     statements, and income statements 2853     2. Analyzing financial metrics and trends 2854     3. Providing clear summaries of financial data 2855     4. Ensuring data accuracy and proper citation 2856     of sources&lt;/instructions&gt; 2857     &lt;tools category="new"&gt; 2858       &lt;tool&gt; 2859         &lt;name&gt;get_balance_sheet&lt;/name&gt; 2860         &lt;description&gt;Retrieve balance 2861         sheet data for a specific 2862         ticker over a given time 2863         period.&lt;/description&gt; 2864       &lt;/tool&gt; 2865       &lt;tool&gt; 2866         &lt;name&gt;get_cash_flow&lt;/name&gt; 2867         &lt;description&gt;Retrieve cash flow 2868         statement data for a specific 2869         ticker over a given time 2870         period.&lt;/description&gt; 2871       &lt;/tool&gt; 2872       &lt;tool&gt; 2873         &lt;name&gt;get_income_statement&lt;/name&gt; 2874         &lt;description&gt;Retrieve income 2875         statement data for a specific 2876         ticker over a given time 2877         period.&lt;/description&gt; 2878       &lt;/tool&gt; 2879       &lt;tool&gt; 2880         &lt;name&gt;analyze_financial_data&lt;/name&gt; 2881         &lt;description&gt;Analyze and 2882         summarize financial statements 2883         to provide meaningful 2884         insights.&lt;/description&gt; 2885       &lt;/tool&gt; 2886     &lt;/tools&gt; 2887     &lt;agent_input&gt; 2888       &lt;key&gt;market_request&lt;/key&gt; 2889       &lt;description&gt;User request for 2890       online financial information 2891       including ticker symbol and time 2892       period.&lt;/description&gt; 2893     &lt;/agent_input&gt; 2894     &lt;agent_output&gt; 2895       &lt;key&gt;market_response&lt;/key&gt; 2896       &lt;description&gt;Requested financial </pre>	<pre> 2897 </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------

2898  
2899  
2900  
2901  
2903  
  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915  
2916  
2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964

```
information and analysis from
online sources.</description>
</agent_output>
</agent>
</agents>
```

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

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:

Investment Recommendation Summary for 2025:

1. Portfolio Allocation Strategy:

- Microsoft (MSFT): 60% of technology allocation
- Apple (AAPL): 40% of technology allocation

2. Investment Rationale:

Microsoft (MSFT) - Primary Position (60%):

- Leading AI integration and monetization potential
- Superior profit margins (37.61% vs 15.52%)
- Strong cloud revenue growth (23% YoY)
- Successful Activision Blizzard integration
- Robust enterprise market position
- Higher recent revenue growth momentum
- Strong cash position (\$78.43B)
- Clear AI leadership advantages

Apple (AAPL) - Supporting Position (40%):

- Superior capital efficiency (ROC: 43.01%)
- Higher Return on Assets (21.24%)
- Strong services growth potential
- Reliable cash flow generation
- Consistent shareholder returns
- Valuable ecosystem lock-in
- Strong emerging market potential
- Value characteristics at current levels

3. Key Monitoring Points for 2025:

Microsoft:

- Cloud market share gains
- AI revenue monetization
- Gaming division integration
- Enterprise spending trends
- Margin sustainability
- Regulatory developments

Apple:

- Services revenue growth
- iPhone replacement cycles
- Emerging market penetration
- Margin pressure management
- AI/ML integration progress
- Supply chain diversification

4. Risk Management Strategy:

Short-term (0-6 months):

- Monitor AI monetization progress
- Track iPhone 15 cycle performance
- Watch macro economic indicators
- Observe cloud growth metrics

Medium-term (6-12 months):

- Evaluate competitive positions in AI
- Assess services growth trajectory
- Monitor regulatory environment
- Track enterprise spending trends

Long-term (12+ months):

- Evaluate ecosystem strength
- Monitor technological shifts
- Assess market share dynamics
- Track return on capital trends

5. Investment Approach:

Implementation Strategy:

- Consider dollar-cost averaging for new positions
- Maintain quarterly rebalancing schedule
- Set stop-loss levels based on risk tolerance
- Monitor position sizes relative to overall portfolio

Adjustment Triggers:

- Significant margin deterioration
- Market share losses in key segments
- Major regulatory actions
- Substantial technological disruption
- Material changes in competitive position

The 60/40 split in favor of Microsoft reflects:

1. Stronger near-term growth prospects in AI and cloud
2. Higher profit margins and improving efficiency metrics
3. Better positioning in enterprise markets
4. More diverse revenue streams
5. Clear leadership in next-generation technologies

However, Apple's position remains crucial due to:

1. Superior capital efficiency
2. Strong ecosystem advantages
3. Significant services growth potential
4. Value characteristics at current levels
5. Consistent shareholder returns

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

2965  
2966  
2967  
2968  
2969  
2970  
2971  
2972  
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  
  
3023  
3024  
3025  
3026  
3027  
3028  
3029

**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

```

3030 <workflow>
3031 <name>parallel_math_solver_workflow</name>
3032 <system_input>
3033 <key>math_problem</key>
3034 <description>The math problem that
3035 needs to be solved.</description>
3036 </system_input>
3037 <system_output>
3038 <key>final_solution</key>
3039 <description>The final solution
3040 determined by majority
3041 voting.</description>
3042 </system_output>
3043 <agents>
3044 <agent category="new">
3045 <name>Math Solver Agent</name>
3046 <description>This agent solves
3047 mathematical problems using
3048 analytical and systematic
3049 approaches.</description>
3050 </agent>
3051 <agent category="new">
3052 <name>Vote Aggregator Agent</name>
3053 <description>This agent aggregates
3054 solutions from different solvers
3055 and determines the final answer
3056 through majority
3057 voting.</description>
3058 </agent>
3059 </agents>
3060 <events>
3061 <event>
3062 <name>on_start</name>
3063 <inputs>
3064 <input>
3065 <key>math_problem</key>
3066 <description>The math
3067 problem that needs to be
3068 solved.</description>
3069 </input>
3070 </inputs>
3071 <outputs>
3072 <output>
3073 <key>math_problem</key>
3074 <description>The math
3075 problem that needs to be
3076 solved.</description>
3077 <action>
3078 <type>RESULT</type>
3079 </action>
3080 </output>
3081 </outputs>
3082 </event>
3083 <event>
3084 <name>solve_with_gpt4</name>
3085 <inputs>
3086 <input>
3087 <key>math_problem</key>
3088 <description>The math
3089 problem that needs to be
3090 solved.</description>
3091 </input>
3092 </inputs>
3093 <task>Solve the math problem using
3094 systematic approach with
3095 GPT-4.</task>
3096 </event>
3097 </events>

```

```

3098 <output>
3099 <key>gpt4_solution</key>
3100 <description>The solution
3101 from GPT-4
3102 </description>
3103 <action>
3104 <type>RESULT</type>
3105 </action>
3106 </output>
3107 </outputs>
3108 <listen>
3109 <event>on_start</event>
3110 </listen>
3111 <agent>
3112 <name>Math Solver Agent</name>
3113 <model>gpt-4o-2024-08-06</model>
3114 </agent>
3115 </event>
3116 <event>
3117 <name>solve_with_claude</name>
3118 <inputs>
3119 <input>
3120 <key>math_problem</key>
3121 <description>The math
3122 problem that needs to be
3123 solved.</description>
3124 </input>
3125 </inputs>
3126 <task>Solve the math problem using
3127 systematic approach with
3128 Claude.</task>
3129 </outputs>
3130 <output>
3131 <key>claude_solution</key>
3132 <description>The solution
3133 from Claude
3134 solver.</description>
3135 <action>
3136 <type>RESULT</type>
3137 </action>
3138 </output>
3139 </outputs>
3140 <listen>
3141 <event>on_start</event>
3142 </listen>
3143 <agent>
3144 <name>Math Solver Agent</name>
3145 <model>claude-3-5-sonnet-20241022</model>
3146 </agent>
3147 </event>
3148 <event>
3149 <name>solve_with_deepseek</name>
3150 <inputs>
3151 <input>
3152 <key>math_problem</key>
3153 <description>The math
3154 problem that needs to be
3155 solved.</description>
3156 </input>
3157 </inputs>
3158 <task>Solve the math problem using
3159 systematic approach with
3160 DeepSeek.</task>
3161 </outputs>
3162 <output>
3163 <key>deepseek_solution</key>
3164 <description>The solution
3165 from DeepSeek
3166 solver.</description>
3167 <action>

```

```

3168 <type>RESULT</type>
3169 </action>
3170 </output>
3171 </outputs>
3172 <listen>
3173 <event>on_start</event>
3174 </listen>
3175 <agent>
3176 <name>Math Solver Agent</name>
3177 <model>deepseek/deepseek-chat</model>
3178 </agent>
3179 </event>
3180 <event>
3181 <name>aggregate_solutions</name>
3182 <inputs>
3183 <input>
3184 <key>gpt4_solution</key>
3185 <description>The solution
3186 from GPT-4
3187 solver.</description>
3188 </input>
3189 <input>
3190 <key>claude_solution</key>
3191 <description>The solution
3192 from Claude
3193 solver.</description>
3194 </input>
3195 <input>
3196 <key>deepseek_solution</key>
3197 <description>The solution
3198 from DeepSeek
3199 solver.</description>
3200 </input>
3201 </inputs>
3202 <task>Compare all solutions and
3203 determine the final answer through
3204 majority voting.</task>
3205 <outputs>
3206 <output>
3207 <key>final_solution</key>
3208 <description>The final
3209 solution determined by
3210 majority
3211 voting.</description>
3212 <action>
3213 <type>RESULT</type>
3214 </action>
3215 </output>
3216 </outputs>
3217 <listen>
3218 <event>solve_with_gpt4</event>
3219 <event>solve_with_claude</event>
3220 <event>solve_with_deepseek</event>
3221 </listen>
3222 <agent>
3223 <name>Vote Aggregator
3224 Agent</name>
3225 <model>deepseek/deepseek-chat</model>
3226 </agent>
3227 </event>
3228 </events>
3229 </workflow>

```

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

---

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

---

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

---

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

1. GPT-4 Solution: - Does not provide a definitive numerical answer - Focuses on theoretical approach and approximations - Inconclusive result
2. Claude Solution: - Uses supplementary angle properties - Converts  $100^\circ$  to  $180^\circ - 80^\circ$  - Arrives at  $-\sqrt{3}$  through numerical calculations - **Final answer:  $-\sqrt{3}$**
3. Deepseek Solution: - Uses similar supplementary angle properties - Also converts  $100^\circ$  to  $180^\circ - 80^\circ$  - Goes through detailed algebraic steps - **Final answer:  $\sqrt{3}$**

Analysis: 1. Two solutions provide definitive answers ( $-\sqrt{3}$  and  $\sqrt{3}$ ), while one is inconclusive 2. Both Claude and Deepseek use similar approaches with supplementary angles 3. The main difference is in the sign of the final answer 4. Claude's solution provides more explicit numerical calculations 5. Looking at the steps: -  $100^\circ$  being in quadrant II - Using supplementary angle properties -  $\tan 100^\circ$  being negative in quadrant II - The final combination with  $4 \sin 100^\circ$

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

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

---

**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:

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

2. Substitute this into the original expression:

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

...

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

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

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

---

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

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


---

**Ground Truth:** 13

---

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

1. **GPT-4 Solution: Arrives at 13** through direct division and remainder calculation - Used division:  $11213141 \div 18$  - 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 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 for 2 and 9 - Got 5 as final answer

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

By majority voting and verification: - **13 is supported by two solutions** - **The reasoning in these solutions is more complete** - We can verify:  $-11213141 \equiv 13 \pmod{18}$  and  $0 \leq 13 < 18$

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

---

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

1. **Simplify the modulus:** First, note that  $-11213141 \pmod{18}$  is equivalent to finding the positive remainder when  $-11213141$  is divided by 18. Since the modulus operation is periodic with period 18, we 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  $11213141 \div 18$ :

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

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

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

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

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  $\angle RPQ = 24/25$  - Assumed  $\angle RPS = \angle RPQ$
2. Claude Solution: Final answer =  $-\frac{24}{25}$  - Used supplementary angles property - Found  $\cos \angle RPQ = 24/25$  - Used the fact that  $\cos(\text{supplementary angles})$  are negatives of each other - **Concluded  $\cos \angle RPS = -\cos \angle RPQ = -24/25$**
3. DeepSeek Solution: Final answer =  $\frac{7}{25}$  - Used complementary angles property - **Found  $\cos \angle RPQ = 24/25$**  - Incorrectly concluded  $\cos \angle RPS = \sin \angle RPQ$

Analyzing the approaches: 1. Looking at the diagram,  $\angle RPS$  and  $\angle RPQ$  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  $\angle RPS$  and  $\angle RPQ$  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.

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

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

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

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

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

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

1. GPT-4's solution:  $r = 7$  2. Claude's solution:  $r = 14/3$  3.

Deepseek's solution:  $r = 14(\sqrt{2} - 1)$

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

Let's verify which one makes more sense:

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

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

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

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

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

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

---

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

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

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

...

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

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

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

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

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

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

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

---

---

3231 **A.7.4 Detailed Trajectory of Creating**  
3232 **‘Financial Agent’**

3233 To provide a more comprehensive demonstration of  
3234 how AutoAgent receives user requirements and au-  
3235 tomates the creation of tools and agents, we present  
3236 the specific agent trajectory for the creation of the  
3237 Financial Agent as follows.

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

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"> <li>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.</li> <li>2. Search the financial information              ⇨ online. You may help me to:             <ul style="list-style-type: none"> <li>- get balance sheets for a given                ⇨ ticker over a given period.</li> <li>- get cash flow statements for a                ⇨ given ticker over a given period.</li> <li>- get income statements for a given                ⇨ ticker over a given period.</li> </ul> </li> </ol> <p>Directly output the form in the XML            ⇨ format without ANY other text.</p>
🤖: Agent Profiling Agent	output	<pre> &lt;agents&gt;   &lt;system_input&gt;     Financial management     ⇨ requests, including:     ...   &lt;/system_input&gt;   &lt;system_output&gt;     &lt;key&gt;financial_response&lt;/key&gt;     ...   &lt;/system_output&gt;   &lt;agent&gt;     &lt;name&gt;Document Manager     ⇨ Agent&lt;/name&gt;     ...   &lt;/agent&gt;   &lt;agent&gt;     &lt;name&gt;Market Research     ⇨ Agent&lt;/name&gt;     ...   &lt;/agent&gt; &lt;/agents&gt;           </pre>
🤖: Tool Editor Agent	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>

<b>👤: Tool Editor Agent</b>	<pre>get_api_plugin_tools_doc ('query_text'='API for ↳ getting financial statements (balance sheet, ↳ cash flow, income ↳ statement) from stock market')</pre>	<p>The rank 1 referenced tool  ↳ documentation is:  API Name: Company Income Statement  ...  The rank 2 referenced tool  ↳ documentation is:  ...  ...</p>
<b>👤: Tool Editor Agent</b>	<pre>create_tool( 'tool_name'= 'get_balance_sheet', 'tool_code'= 'from metachain.types ↳ import Result, Agent ... ')</pre>	<p>[SUCCESS] Successfully created tool:  ↳ get_balance_sheet in  ↳ /workplace/metachain/metachain/ tools/get_balance_sheet.py</p>
<b>👤: Tool Editor Agent</b>	<pre>run_tool( 'tool_name'= 'get_balance_sheet', 'run_code'= 'from metachain.tools ↳ import get_balance_sheet result = get_balance_sheet("AAPL", ↳ "QUARTERLY") print(result)')</pre>	<p>[The output of the tool `run_tool`  ↳ showing in the interactive  ↳ terminal]  Terminal viewport position: Showing  ↳ page 1 of 1.  =====</p> <p>[SUCCESS]  {  "symbol": "AAPL:NASDAQ",  "type": "stock",  "period": "QUARTERLY",  "balance_sheet": [  ...  =====</p>
<b>👤: Tool Editor Agent</b>	<pre>create_tool( 'tool_name'= 'get_cash_flow', 'tool_code'= 'from metachain.types ↳ import Result, Agent ... ')</pre>	<p>[SUCCESS] Successfully created tool:  ↳ get_cash_flow in  ↳ /workplace/metachain/metachain/ tools/get_cash_flow.py</p>
<b>👤: Tool Editor Agent</b>	<pre>run_tool( 'tool_name'= 'get_cash_flow', 'run_code'= 'from metachain.tools ↳ import get_cash_flow result = get_cash_flow("AAPL", ↳ "QUARTERLY") print(result)')</pre>	<p>[The output of the tool `run_tool`  ↳ showing in the interactive  ↳ terminal]  Terminal viewport position: Showing  ↳ page 1 of 1.  =====</p> <p>[SUCCESS]  {  "symbol": "AAPL:NASDAQ",  "type": "stock",  "period": "QUARTERLY",  "cash_flow": [  ...  =====</p>
<b>👤: Tool Editor Agent</b>	<pre>create_tool( 'tool_name'= 'get_income_statement', 'tool_code'= 'from metachain.types ↳ import Result, Agent ...')</pre>	<p>[SUCCESS] Successfully created tool:  ↳ get_income_statement in  ↳ /workplace/metachain/metachain/ tools/get_income_statement.py</p>

<b>👤: Tool Editor Agent</b>	<pre>run_tool( 'tool_name'= 'get_income_statement', 'run_code'='from ↳ metachain.tools import ↳ get_income_statement result = ↳ get_income_statement ("AAPL", "QUARTERLY") print(result)')</pre>	<pre>[The output of the tool `run_tool` ↳ showing in the interactive ↳ terminal] Terminal viewport position: Showing ↳ page 1 of 1. ===== [SUCCESS] {   "symbol": "AAPL:NASDAQ",   "type": "stock",   "period": "QUARTERLY",   "income_statement": [     ...   ] } =====</pre>
<b>👤: Tool Editor Agent</b>	<pre>create_tool( 'tool_name'= 'analyze_financial_data', 'tool_code'= 'from metachain.types ↳ import Result, Agent ... ')</pre>	<pre>[SUCCESS] Successfully created tool: ↳ analyze_financial_data in ↳ /workplace/metachain/metachain/ tools/analyze_financial_data.py</pre>
<b>👤: Tool Editor Agent</b>	<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>	<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: ... =====</pre>
<b>👤: Tool Editor Agent</b>	<b>Output</b>	<pre>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.</pre>

**👤: Agent Editor  
Agent**

```
create_agent(
 'agent_name'=
 'Document Manager Agent',
 'agent_description'=
 'Specialized agent for
 ↳ managing and analyzing
 ↳ private financial
 ↳ documents stored
 ↳ locally.',
 'agent_tools'=
 ['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:...'')
```

Successfully created agent: Document  
↳ Manager Agent in  
↳ /workplace/metachain/metachain/  
agents/document\_manager\_agent.py

**👤: Agent Editor  
Agent**


```
create_agent(
 'agent_name'=
 'Market Research Agent',
 'agent_description'=
 'Specialized agent for
 ↳ retrieving and analyzing
 ↳ online financial
 ↳ information for publicly
 ↳ traded companies.',
 'agent_tools'=
 ['get_cash_flow', ...],
 'agent_instructions'=
 'You are a specialized
 ↳ market research agent
 ↳ responsible for
 ↳ retrieving and analyzing
 ↳ financial information
 ↳ from online sources.
 ↳ Your primary
 ↳ responsibilities
 ↳ include: ...'')
```

Successfully created agent: Market  
↳ Research Agent in  
↳ /workplace/metachain/metachain/  
agents/market\_research\_agent.py

## 👤: Agent Editor Agent


```
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: ...')
```

```
[ERROR] Failed to create agent.
↳ Error: File
↳ "/workplace/metachain/metachain/
agents/financial_analysis_orchestrator.py",
↳ line 32
def
↳ transfer_to_document_manager_agent(Document
↳ analysis request including
↳ directory paths and analysis
↳ requirements: str):
^^^^^^^^^^
SyntaxError: invalid syntax
```

 **Agent Editor**  
**Agent**

```
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':
↳ 'doc_request',
↳ 'agent_output':
↳ 'doc_response'},
↳ {'name': 'Market
↳ Research Agent',
↳ 'agent_input':
↳ 'market_request',
↳ 'agent_output':
↳ 'market_response'}]],
'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: ...')
```

Successfully created agent:  
↳ Financial Analysis Orchestrator  
↳ in /workplace/metachain/metachain/  
agents/financial\_analysis\_orchestrator.py

 Agent Editor  
Agent

```
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'={'})
```


```
[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.
=====
```

```
[SUCCESS]
**** Receive Task ****
Receiving 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.

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:
...

**** Assistant Message ****
Document Manager Agent: ...
=====
```

 Agent Editor  
Agent

```
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.
=====
```

```
**** 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:

Investment Recommendation Summary
↳ for 2025:

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

2. Investment Rationale:

Microsoft (MSFT) - Primary Position
↳ (60%):
...
=====
```

🤖: Agent Editor  
Agent

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.

## **A.8 LLM usage**

In the writing of this paper, the LLM was used only for correcting word choice and grammar. The LLM was employed exclusively as a research subject in the development of agents and was not involved in any critical stages of human-driven research, including idea generation, data analysis, or other core research activities.