Volume 11, Issue 1, January – 2026 International Journal of Innovative Science and Research Technology
ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26jan348

Agentic AI Overview

Somil Asthana[1]

[1]Tide. Co

Publication Date: 2026/01/21

Abstract: Agentic AI represents a paradigm shift from traditional single-shot LLM applications to iterative, autonomous workflows that mirror human problem-solving approaches. Agents range from less autonomous to highly autonomous enabling spectrum of use.

This article equips data scientists and AI practitioners with the conceptual framework and practical methodology to design, implement, and optimize agentic AI workflows for complex, autonomous task execution using 4 foundational Agentic design patterns. Giving pragmatic five-step approach to building agentic workflow.

How to Cite: Somil Asthana (2026) Agentic AI Overview. *International Journal of Innovative Science and Research Technology*, 11(1), 1344-1349. https://doi.org/10.38124/ijisrt/26jan348

I. INTRODUCTION

➢ *What Is Agentic AI?*

Traditional non-agentic LLM-based solutions, such as chatbots, operate through single-shot execution—generating outputs directly from prompts. While this approach has proven effective for many use cases, it has inherent limitations. Agentic AI workflows, by contrast, employ a multi-step iterative process where agents mirror human problem-solving patterns: they analyze, research, revise, and refine their approach before producing a final result. This iterative loop, built on LLM foundations, may require multiple steps and introduce latency, but it consistently delivers superior outputs compared to non-agentic solutions.

➢ *Agentic AI Workflow*

An Agentic AI workflow consists of multiple coordinated steps designed to complete complex tasks with varying degrees of autonomy. These solutions exist along a spectrum:

- Less autonomous agents operate within fully predetermined sequences, executing predefined steps with limited flexibility. While effective for their assigned tasks, their capabilities are constrained by their rigid structure.
- Highly autonomous agents possess substantial decision-making freedom, including the ability to determine internal processing steps, invoke appropriate tools, generate code dynamically, and iteratively refine outputs through multiple cycles of reflection and improvement. These agents handle more sophisticated tasks and produce higher-quality results.

➢ *Degree of Autonomy*

- *Less Autonomous Agents:*

✓ All steps are predefined
✓ All tools are hard-coded
✓ Autonomy is limited to text generation

- *Highly Autonomous Agents:*

✓ Make many decisions autonomously
✓ Can create new tools or code on the fly
✓ Determine their own processing strategies

➢ *Benefits of Agentic AI Workflows*

- Effective execution: Agents perform tasks with some accuracy and reliability.
- Complex output: Agents can handle sophisticated, multi-faceted tasks that exceed single-shot capabilities.
- Parallelism: Agents can execute multiple intermediate tasks concurrently and synthesize results.
- Modularity: Workflows can be modified by replacing LLMs or adjusting available tools without restructuring the entire system.

➢ *Building Blocks for Agentic Workflow*

A fundamental principle in designing agentic workflows is decomposition: analyze existing system processes and identify discrete steps that can be implemented either through LLMs or specialized software tools. If agents cannot execute a step effectively, it should be further decomposed into manageable substeps.

Table 1 Building Blocks for Agentic Workflow

Building Block	Example	Usecase
Models	LLM	Text generation, tool use, information extraction
Algorithms	Machine Learning Models, Heuristic Models, Statistical Regression Models, Python code	Prediction, rule-based logic, regression models, pdf-to-text, text-to-speech, image analysis
Tools	MCP, function tools	Web Search, real-time feeds, check social profile, handle, check calendar, RSS feeds
	Information Retrieval	Database, Retrieval System like RAG

II. LITERATURE COMPARISON AND POSITIONING

This tutorial aligns closely with foundational work in agentic AI while providing a practical, implementation-focused perspective for data scientists and practitioners. The four design patterns presented in this tutorial—Reflection, Tool Use, Planning, and Multi-Agent—align directly with the framework popularized by Andrew Ng in his 2024 presentations and DeepLearning.AI courses on agentic workflows (Ng, 2024). This tutorial extends his framework by providing detailed implementation guidance, evaluation strategies, and practical workflows for production deployment.

The Tool Use pattern described in this tutorial is based on the ReAct (Reasoning and Acting) framework introduced by Yao et al. (2023) transforms the idea into practical implementation.

Our evaluation framework uses objective (Python-based) and subjective (LLM-as-judge) approaches as documented by (Raschka, 2024; Chang et al., 2024)

➤ *Agentic Design Patterns*

Four critical agentic design patterns form the foundation of sophisticated workflows. These patterns can be combined in various configurations to create complex, powerful systems.

Table 2 Agentic Design Patterns

Design Pattern	Design Pattern Description
Reflection	An LLM (the same or a different one) reviews previously generated text, evaluates its quality, and suggests improvements. This iterative critique process enhances output quality.
Tool use	LLMs are provided with tools (functions) to assist with assigned tasks. Examples include code execution for analytical tasks, database resources and web search for information gathering, email/calendar tools for productivity, and image libraries for visual tasks.
Planning	A collaborative workflow incorporating four stages: (1) Task Planning, (2) Model Selection, (3) Task Execution, and (4) Response Generation. This framework enables flexible, autonomous operation.autonomously.
Multi-agentic workflow	Multiple agents coordinate, debate, and converge toward consensus. While challenging to control, preliminary research indicates these workflows can achieve exceptional accuracy.

➤ *Agentic Pattern: Reflection*

Just as humans reflect on their work to identify improvements, LLMs can be prompted to evaluate their own outputs. Reflection is particularly effective in code generation, where reasoning-focused LLM models excel at identifying inconsistencies and bugs. Incorporating external feedback into the reflection process further enhances generation efficiency and quality.

The below figure illustrates a reflection pattern flow.

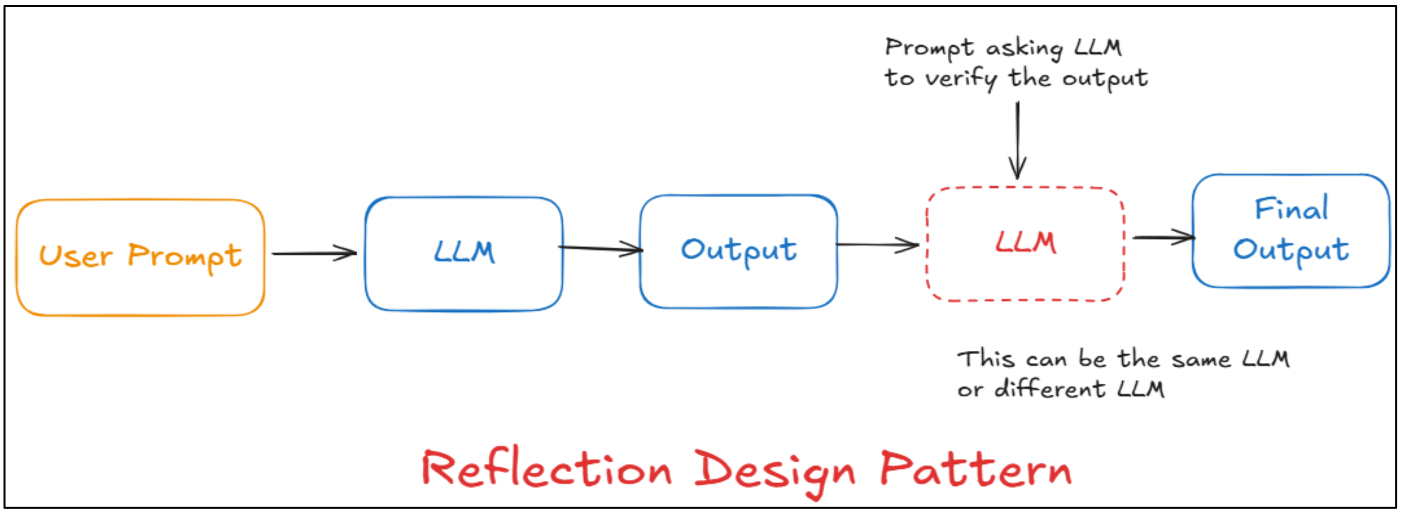
Prompt asking LLM to verify the output
User Prompt
LLM
Output
LLM
Final Output
This can be the same LLM or different LLM
Reflection Design Pattern

Fig 1 Reflection Design Pattern

➢ *Agentic Pattern: Tools*

Tools extend LLM capabilities beyond text generation, enabling agents to take actions, retrieve information, and perform computations. In this pattern, tool information is provided to the agentic workflow, and the LLM autonomously decides whether and when to invoke specific functions. After execution, function outputs are fed back into the conversation history, allowing agents to generate informed final responses.

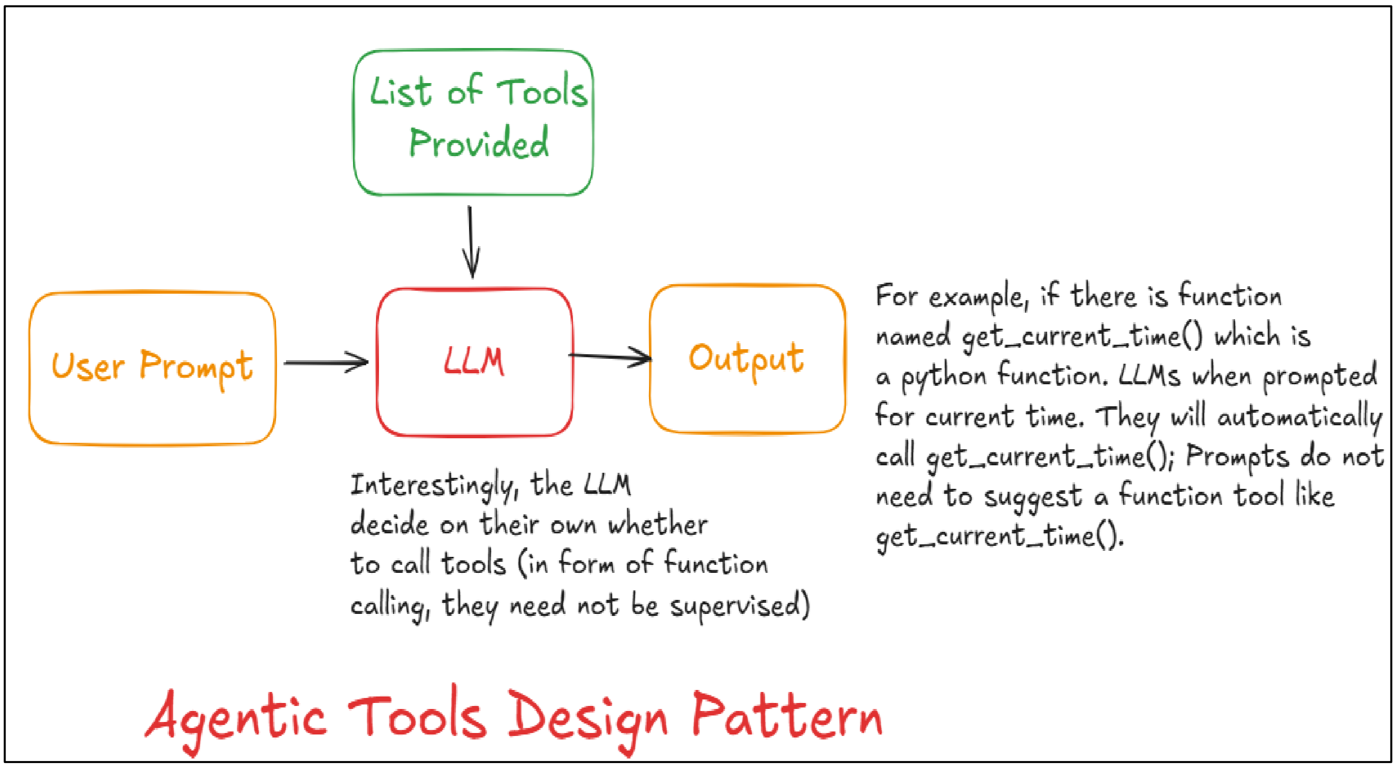
List of Tools Provided
User Prompt
LLM
Output
For example, if there is function named get_current_time() which is a python function. LLMs when prompted for current time. They will automatically call get_current_time(); Prompts do not need to suggest a function tool like get_current_time().
Interestingly, the LLM decide on their own whether to call tools (in form of function calling, they need not be supervised)
Agentic Tools Design Pattern

Fig 2 Agentic Tools Design Pattern

Table 3 Agentic Pattern: Tools

Scenario	Tool	Expected Output
Prompt asking to search for nearby recreational activity.	web search tool with location and date time tool	Agents combine tools to provide information about clubs or activity centers
Business Intelligence asking for specific query on data	SQL query creation tool and SQL execution Python function	Agents generate SQL queries, execute them via database connections, and provide business intelligence insights
Complex calculations to compute ROI for a portfolio	Function call to retrieve asset prices combined with ROI calculation tool	Agents calculate current portfolio ROI

Modern LLMs no longer require extensive descriptions in prompts to invoke functions. Today's models are increasingly trained to interpret and call functions based solely on function names and docstrings, demonstrating enhanced logical decision-making capabilities.

One of the most powerful tools available to agents is code execution, which allows agents to generate Python code that executes in the background. This transforms agents from simple text generators into computational engines capable of sophisticated planning and execution.

Another significant advancement is the Model Context Protocol (MCP), which extends function tool capabilities to remote servers, effectively allowing agents to leverage server functionalities as callable functions, dramatically expanding their power and reach.

➢ *Agentic Pattern: Planning*

Agentic workflows become significantly more flexible when planning components allow for dynamic rather than fixed execution flows. In this pattern, LLMs determine the optimal course of action rather than following predetermined sequences. A planner LLM receives functional tools and prompts as inputs, it generates step-by-step instructions (plan) required for achieving desired outputs. Each instruction/task, along with relevant context, prompt and tools, is passed to subsequent LLMs to run, forming a processing chain from input to final output.

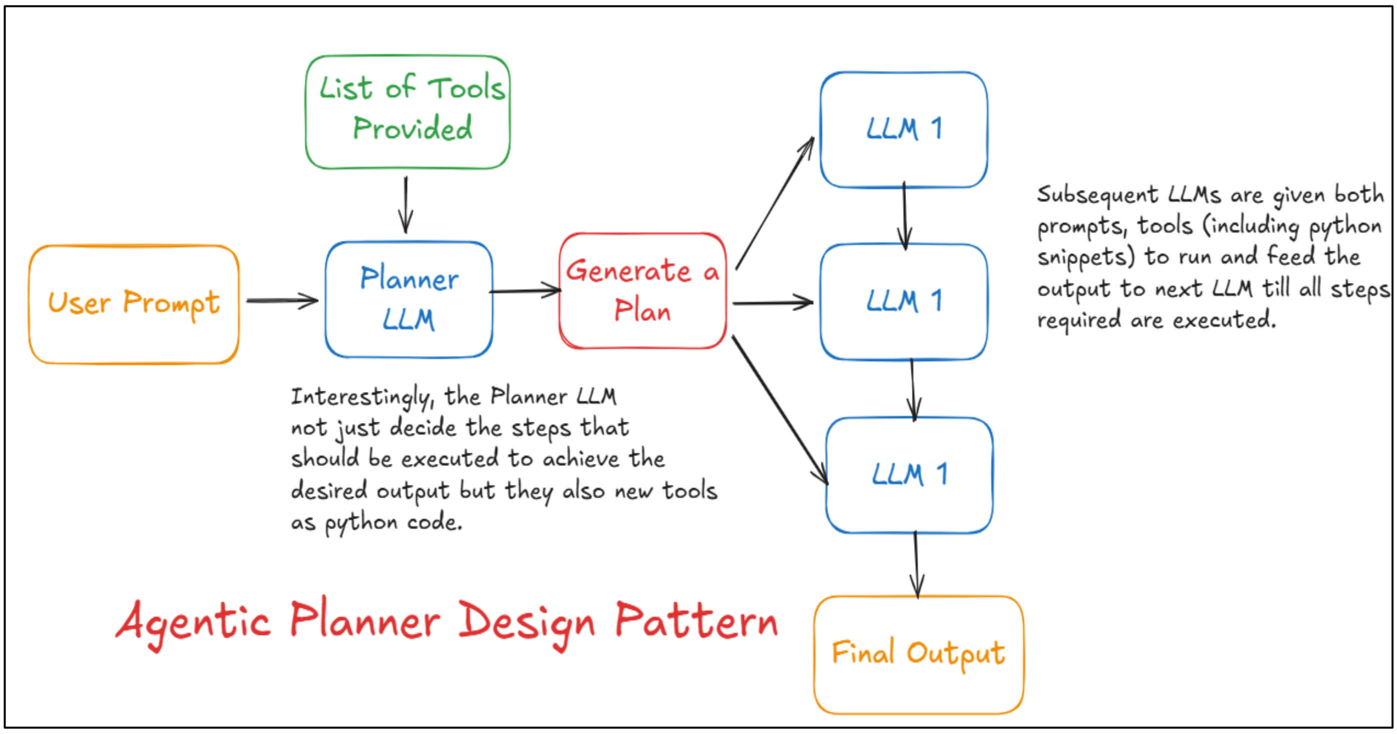
List of Tools Provided
User Prompt
Planner LLM
Generate a Plan
LLM 1
LLM 1
LLM 1
Final Output
Subsequent LLMs are given both prompts, tools (including python snippets) to run and feed the output to next LLM till all steps required are executed.
Interestingly, the Planner LLM not just decide the steps that should be executed to achieve the desired output but they also new tools as python code.
Agentic Planner Design Pattern

Fig 3 Agentic Planner Design Pattern

This pattern eliminates the need to predefine function call sequences. Instead, agents use available tools and prompts to formulate plans—typically in JSON format—specifying which function tools subsequent LLMs should invoke. While this approach reduces control over execution flow, it significantly enhances adaptability and problem-solving capability.

➢ *Agentic Pattern: Multi-Agent*

While it might seem that extending a single agentic workflow with additional function tools or layers would suffice, multi-agent patterns offer distinct advantages. This approach enables agent reuse across domains and promotes modular development. For example, an agentic workflow designed for creating graphic visualizations can be reused across multiple applications. Additionally, teams can develop agentic solutions independently and in parallel, then integrate them later.

Multi-agent communication remains an active research area, with ongoing work on optimal communication patterns and coordination protocols. Despite control challenges, this pattern offers substantial flexibility and scalability benefits.

III. EVALUATION

Building an optimal agentic workflow is inherently iterative—it's difficult to know in advance which components will succeed and how to evaluate them effectively. A best practice is to build a "quick and dirty" initial prototype, examine its outputs (including errors and unsatisfactory results), and then systematically improve it.

Start by collecting 10-20 representative inputs and running them through the agentic workflow to identify common output errors. This empirical approach quickly reveals the most frequent failure modes. For more systematic evaluation, develop automated evaluation functions to track specific error types. For instance, in an invoice processing workflow, testing 10-20 invoices might reveal that date extraction is the most common error. You can then create an evaluation function specifically for date extraction, measure accuracy, implement improvements, and monitor changes over time.

Errors fall into two categories: objective and subjective. Objective errors involving numerical outputs can be evaluated using Python functions. Subjective errors, particularly those involving text quality, benefit from an "LLM-as-judge" approach, where outputs are compared against 3-5 gold standard examples for each domain.

Table 4 Two “Axes” of Evaluation

	Evaluate with python code	**LLM-as-judge (subjective)**
Target variable is number or count	Checking for certain numerical variables like invoice date. Python Eval Function	Count number of gold standard points generated by agentic workflow
Target variable is text	Checking for text length	Decoding images and charts

IV. STEPS FOR CREATING AN AGENTIC WORKFLOW

The challenge in creating agentic workflows lies in the multitude of possible approaches: Should you invest time perfecting workflow design or focus on LLM selection? With numerous design patterns available, the process can seem daunting.

➢ *Step 1: Plan a Quick and Direct Workflow Prototype*

Avoid spending weeks or months theorizing about an ideal solution. Given the novelty of these systems, it's more effective to create a rapid, imperfect prototype that addresses even a subset of requirements. This provides a concrete foundation for iterative improvement.

➢ *Step 2: Select or Design Functional Tools*

Before evaluating LLM and trade-offs, identify and implement the function tools agents will need. For an invoice processing workflow, this might include: (1) PDF-to-text conversion, (2) date extraction tool, and (3) database update functionality for storing invoice details.

➢ *Step 3: Initially Choose Small Language Models (SLM) or Larger SLMs*

Selecting the appropriate LLM—balancing general-purpose versus specialized multi-modal capabilities—can be complex. For initial prototyping, start with Small Language Models (SLMs), which can be replaced with more powerful LLMs as needed.

- *Examples of SLMs:*

✓ Text generation: Llama 3.2-1B, DeepSeek-R1-1.5B, Phi-3.5-Mini-3.8B
✓ Multi-modal: Phi-4-multimodal, Gemma 3-4B
✓ Code generation: CodeGemma (Google), StarCoder2 (BigCode/HuggingFace)

➢ *Step 4: Implement a Workflow*

Write code and develop prompts to connect LLMs and construct an initial prototype.

➢ *Step 5: Conduct an Error Analysis*

Error analysis identifies underperforming components that degrade overall system performance. This process helps prioritize improvements such as: one might need to replace an LLM with a more capable model or, revise tools (computational functions, web search, MCP) that produce substandard outputs, or refine prompts with more explicit instructions and examples.

Focus on the component producing the least satisfactory intermediate output. Conduct component-level error analysis, implement improvements, then move to the next problematic component. This systematic approach drives overall performance improvement and reveals workflow bottlenecks.

➢ *Example: Stock Portfolio Agentic Workflow*

An example workflow for stock portfolio management connects to a broker account to fetch current information, performs profit and loss calculations, and summarizes results in a database.

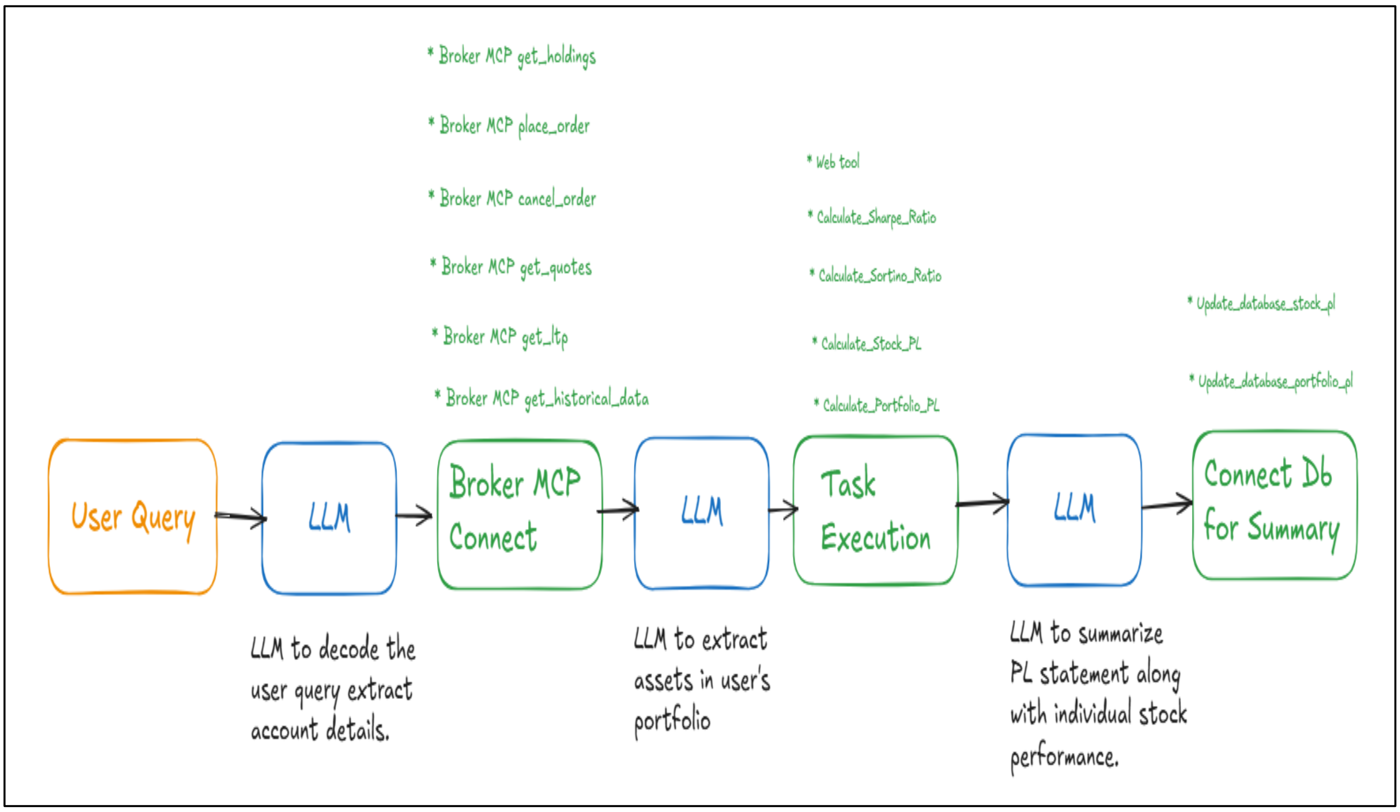
* Broker MCP get_holdings
* Broker MCP place_order
* Broker MCP cancel_order
* Broker MCP get_quotes
* Broker MCP get_ltp
* Broker MCP get_historical_data
* Web tool
* Calculate_Sharpe_Ratio
* Calculate_Sortino_Ratio
* Calculate_Stock_PL
* Calculate_Portfolio_PL
* Update_database_stock_pl
* Update_database_portfolio_pl
User Query
LLM
Broker MCP Connect
LLM
Task Execution
LLM
Connect Db for Summary
LLM to decode the user query extract account details.
LLM to extract assets in user's portfolio
LLM to summarize PL statement along with individual stock performance.

Fig 4 Example: Stock Portfolio Agentic Workflow

- *One Needs Following Tools*

✓ Broker MCP get_holdings
✓ Broker MCP place_order
✓ Broker MCP cancel_order
✓ Broker MCP get_quotes
✓ Broker MCP get_ltp
✓ Broker MCP get_historical_data
✓ Web tool
✓ Calculate_Sharpe_Ratio
✓ Calculate_Sortino_Ratio
✓ Calculate_Stock_PL
✓ Calculate_Portfolio_PL
✓ Update_database_stock_pl
✓ Update_database_portfolio_pl

- *LLMs to Complete Following Tasks*

✓ LLM to decode the user query extract broker account details
✓ LLM to extract assets from the user's portfolio, identify the symbols. Fetch their LTP.
✓ LLM to summarize PL statements along with individual stock performance.

Finally, prompts along with context to stitch a complete workflow.

V. CONCLUSION

Our novel contribution in writing this article 1) Integrated Methodology: Rather than treating design patterns, evaluation, and implementation in silo, we present an integrated five-step guide for practitioners to prototype Agentic workflow to production. 2) Emphasis on Rapid Prototyping: Agentic AI is still an active area of research and developers are trying to figure out best way to design and code therefore, a "quick and dirty" prototype-first approach followed by evaluation can assist in developing a workable Agentic AI workflow that can be improved as one monitors and verify the output. 3) Small Language Model (SLM) Focus: Our recommendation to start with SLMs (Llama 3.2-1B, DeepSeek-R1-1.5B, Phi-3.5-Mini) for initial prototyping is often overlooked in practice - engineers and businesses want to the best LLM and tools fail at times because of the complexity involved in designing an Agentic workflow.

REFERENCES

[1]. Ng, A. (2024). Four design patterns for AI agentic workflows. *LinkedIn Post*, March 2024. Retrieved from https://www.linkedin.com/posts/andrewyng_one-agent-for-many-worlds-cross-species-activity-7179159130325078016
[2]. Ng, A. (2024). Agentic AI [Online Course]. *DeepLearning.AI*. Retrieved from https://www.deeplearning.ai/courses/agentic-ai/
[3]. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR).* https://arxiv.org/abs/2210.03629