Week 1 Assignment: Understanding AI Agents

Part 1: Reflection Questions

Answer the following questions based on the concepts covered in the lecture:

1. **Foundational Understanding**
   How would you explain the difference between traditional AI systems and AI agents to someone with limited technical background?

Your Answer Here:

a. Traditional AI System is based on the predictive analytics but Agentic AI is an autonomous system which can think, plan, reflect, update and take action to achieve a given goal.
b. Traditional system was human directed, so it can predict on what's instructed by the human whereas Agentic AI system is AI-directed. It is self-reflective, so it can think ahead than human and ask for human to get feedback.
c. In traditional AI system, model was trained on train dataset and predicting on another set of data but in Agentic AI system, Model + reasoning + tools + memory + environment were the active parts of the agent building ecosystem.
d. Improving accuracy of the prediction was the aim of the traditional AI system, whereas achievement of long-term goals through reasoning, required context using memory is the aim of Agentic AI system.
e. Single LLM or Deep Learning or Machine Learning model was the core engine of traditional AI system, whereas in Agentic system multiple LLMs as part of multi-agent system orchestrate to achieve the goal autonomously.
f. Traditional AI system is stateless, whereas Agentic AI system is stateful with short and long-term memory.
g. In Traditional AI system, in order to perform some task a manually written code needs to be integrated with the model output, but in Agentic AI, this can be achieved autonomously through Function Calling.
h. As part of Predictive Engineering, a Forecasting needs to be built to forecast customer demand, but in Agentic AI system, Forecast agent applies ML timeseries model autonomously, by analysing returned data frame using Python.

2. **Professional Application**
   Think about your organization or industry. What specific tasks could benefit from AI agents? What would an implementation look like for one of these tasks?

Your Answer Here:

1. Text-to-SQL Agent on top of Data lakehouse: A natural-language interface to our data warehouse (Snowflake, Databricks, or BigQuery).
   1. Fin ops analyst can ask questions like “Show me the total cost of storage for October”, and the agent converts it to SQL using schema-aware reasoning, executes it, and explains the results in plain English.
   2. Sales Manager: "What's the total sales for Q3, 2025?" AI assistant: "Total Sales for Q3, 2025 is USD 251M".
2. RAG-based Knowledge Management Retriever: Fetches information specific to user query from Confluence, Slack, emails, Minutes of Meetings repositories and share it with the users

along with reference links, avoiding the manual search.
3. Autonomous Functional Test Generation and execution using Playwright MCP and GitHub copilot.

Supervisor based Multi-Agent architecture for Text-to-SQL agent

Agent Name	Responsibilities	Input	Output	Departments Served
Global Intent Router Agent	Detects which domain the query belongs to → FinOps, Sales, Marketing, DevOps, QA, Product, CFO etc.	User NL query	Domain label + task type (SQL_QUERY / FORECAST / RCA / REPORT)	All
Domain Router Agent	Once domain is detected, route to correct domain-specific pipeline	Domain label	Assigned domain pipeline	All
FinOps Query Planner Agent	Understand cost, usage, tagging, budget, optimization queries	Query + FinOps schema	FinOps query plan	FinOps, Engineering, CFO
Sales Analytics Planner Agent	Revenue, pipeline, ARR/MRR, forecasts, customer trends	Query + CRM schema	Sales analytics plan	Sales, Revenue Ops
Marketing Insights Planner Agent	Campaign analysis, CAC, attribution, engagement metrics	Query + marketing schema	Marketing plan	Marketing
DevOps Telemetry Planner Agent	Deployments, CI/CD metrics, failures, MTTR, efficiency	Logs + observability schema	DevOps analytics plan	DevOps, SRE
Product Analytics Planner Agent	Feature usage, churn, funnels, activation metrics	Product DB schema	Product plan	Product, PMs
QA Quality Planner Agent	Test coverage, pass rate, regression failures	QA test schema	QA analysis plan	QA, Engineering
CFO Financial Planner Agent	Budget, forecasting, unit economics, margin analytics	Finance schema	CFO analytics plan	CFO, Finance

Implementation roadmap

Week	Milestone	Detailed Tasks (Expanded)
Week 1	Schema + Metadata Extraction & RAG Store	• Extract Snowflake/BigQuery schema using INFORMATION_SCHEMA• Build schema embeddings (table definitions, columns, relationships)• Build a Metadata Store (dbt docs or custom YAML)• Set up Vector DB for schema RAG• Prepare initial schema JSON for prompt conditioning
Week 2	Router + Entity Extractor + Prompt Design v1	• Implement Intent Router (FinOps-specific intents)• Implement Entity Extractor with schema lookup• Build Entity → SQL mapping rules (e.g., "compute cost" → SUM(cost))• Design modular prompts for router, entity extraction, SQL generator• Add PromptLayer versioning for each sub-agent
Week 3	Query Planner + SQL Generator v1	• Build Query Planner to detect joins, aggregations, date inference• Implement heuristic rules (e.g., always join resource_dim for resource_name)• SQL Generator Agent integrated with schema embeddings• Add pattern library for FinOps queries (cost by service, by region, amortized RI, etc.)• First working pipeline for simple queries
Week 4	SQL Guardrail Agent + Execution Layer	• Build validator for: SQL injection, SELECT * limits, missing columns, wrong table use• Add Query Cost Estimator (Snowflake PROFILE API)• Build RLS/Tagging policy checker• Implement Query Executor (Python + SQLAlchemy)• Add error-handling & fallback mechanisms
Week 5	Result Interpreter + FinOps Domain Reasoner	• Build business logic layer: cost allocation, cost per resource, BU mapping• Interpret results: "Your S3 cost increased by 28% MoM"• Add confidence scoring• Add chart templates (bar, trend, pie) for cost reporting• Add caching for repeated queries
Week 6	DeepEval Regression Tests + Prompt Tuning	• Create unit tests for each agent (router, SQL generator, interpreter)• Benchmark SQL accuracy on 50–100 test prompts• Add prompt alias + version hyperparameters• Build regression suite for multi-turn queries• Compare multiple prompt versions + LLM models
Week 7	Governance, Observability, Knowledge Graph	• Add monitoring for pipeline latency, SQL cost, errors• Build lineage: user query → plan → SQL → results• Create simple FinOps Knowledge Graph (services, pricing, usage types)• Integrate KG into entity extraction & RCA• Add incident tracing for debugging
Week 8	Deployment + CI/CD + UX Layer	• Deploy on Vercel API + Snowflake connectivity• Build FastAPI wrapper for agent graph• Add GitHub Actions for CI/CD• Add UI for "Chat with Your FinOps Warehouse"• Final load testing + production guardrails

3. **Architecture Analysis**
   The virtual assistance example we discussed in the lecture employs a specific multi-agent architecture. What type of architecture is it, and why is this approach effective for this application?

Your Answer Here:

The Virtual Assistant use case is a Supervision multi-agent where the architecture consists of:

- One central Orchestrator Agent
- Multiple specialized Executor (or Sub-) Agents, such as:
    - Gmail Agent
    - Slack Agent
    - Web Search Agent

Pipeline

a. **User's Prompt:** The user provides a natural language query like: "Summarize my unread emails from Gmail and recent messages from Slack."
b. **Orchestrator Agent:** The orchestrator interprets the intent and decides which specialized agents are needed. Example:
    - Route to Gmail Agent to fetch emails.
    - Route to Slack Agent to pull messages.
    - Optionally use Web Search Agent for context.
      Then, it aggregates or merges the outputs.

c. Executor Agents: Each performs its **domain-specific** task:
    - Gmail Agent → interacts with Gmail API
    - Slack Agent → retrieves workspace messages
    - Web Search Agent → searches the internet for extra info

d. Result Aggregation: The orchestrator collects all partial responses and composes a **final coherent answer** back to the user.

Benefits

1. Modularity: Each agent is focused on a single capability (email, chat, web search), making the system easier to extend and maintain.
2. Scalability: New agent e.g., "Calendar Agent" can be added without changing the core orchestrator logic.
3. Concurrency: Multiple agents can work concurrently, reducing overall response time.
4. Context-Awareness: The orchestrator can decide which sources are relevant for a given query and combine insights intelligently.
5. Flexibility: Tasks that span multiple domains (like "summarize my communications") require multi-source reasoning — perfect for orchestration.
6. Abstraction Layer: Users interact with a single assistant (the orchestrator), hiding the complexity of multi-system coordination.

4. **Components Analysis**
   Choose one cognitive component of AI agents (Perception, Reasoning, Action, or Feedback & Learning) and explain how it manifests in a real-world AI application of your choice.

Your Answer Here: I am considering the FinOps AI agent which I will be building for my upcoming week's assignments.

Reasoning in FinOps AI Agents

- **Purpose:** Analyze financial and operational data to derive insights and make cost-optimization decisions.

- **How it manifests:**
    - Evaluates cloud cost drivers (compute, storage, networking) using historical and real-time data.
    - Applies LLM reasoning to identify anomalies or cost spikes.
    - Correlates spending patterns with business activity (e.g., “Cost increased due to scaling of data pipeline jobs”).
    - Prioritizes actions based on impact and confidence — “Terminate idle VMs first; optimize storage next.”
    - Generates natural language explanations for finance or engineering teams (“Your EC2 spend rose 15% due to longer batch job runtimes.”).

Action in FinOps AI Agents

- **Purpose:** Execute or recommend cost-optimization steps based on reasoning outcomes.
- **How it manifests:**
    - Automates workflows — e.g., triggering instance rightsizing, shutting down unused resources, or adjusting budgets.
    - Recommends actions — “Move infrequent-access data to Glacier” or “Switch to reserved instances.”
    - Integrates with tools — via APIs (AWS Cost Explorer, Azure Cost Management, GCP Billing) to enforce actions.
    - Logs all actions into the Context Engineering Layer (CEL) for auditability and continuous learning.
    - Triggers feedback loops — evaluates post-action savings and updates its reasoning model for improved future decisions.

Part 2: Case Evaluation - When to Use Agentic Systems

For each of the following scenarios, evaluate whether an agentic AI system would be appropriate. Explain your reasoning by discussing the benefits and drawbacks of using an agent versus a non-agentic solution.

1. **Simple Data Lookup**: A system that retrieves specific information from a structured database in response to direct queries (e.g., "What is the price of Product X?")

Your Answer Here: **Not appropriate for agentic AI**

- Reasoning: Task consists of a set of pre-defined steps, so as deterministic and requires no multi-step reasoning or decision-making.
- Benefits of non-agentic: Faster, cheaper, reliable SQL/API query.
- Drawback of agentic: Adds unnecessary complexity and latency

2. **Basic Calculator**: A tool that performs straightforward mathematical operations like addition and multiplication

Your Answer Here: **Not appropriate for agentic AI**

- Reasoning: Simple arithmetic tasks, to be performed by a basic calculator, requires no autonomy or context understanding.
- Benefits of non-agentic: Lightweight, instant, and accurate computation.
- Drawback of agentic**:** Overkill — no need for reasoning or planning.

3. **Travel Planning Assistant**: A system that needs to coordinate flights, accommodations, transportation, and activities based on user preferences, budget constraints, and real-time availability

Your Answer Here: **Highly appropriate for agentic AI**

- Reasoning: Requires multi-step coordination (flights, hotels, preferences).
- Benefits: Agents can plan, compare, and rebook dynamically.
- Drawbacks: Higher complexity and cost, needs continuous context updates.

4. **Research Synthesis**: A system that searches across multiple sources, extracts relevant information, evaluates credibility, and compiles findings into a cohesive report

Your Answer Here: **Appropriate for agentic AI**

    - Reasoning: Needs reasoning, summarization, and judgment across sources.
    - Benefits: Automates literature review and report generation.
    - Drawbacks: Risk of hallucination, requires source validation.

5. **Smart Home Coordinator**: A system that manages multiple connected devices, anticipates user needs based on patterns, and proactively adjusts settings accordingly

Your Answer Here: **Appropriate for agentic AI**

- Reasoning: Must sense environment, infer intent, and act autonomously.
- Benefits: Proactive adaptation, energy savings, comfort optimization.
- Drawbacks: Privacy concerns, reliability in edge cases.

6. **Email Management**: A system that sorts incoming emails and drafts responses

Your Answer Here: **Appropriate for agentic AI**

    - Reasoning: Task involves prioritization, summarization, and drafting context-aware replies.
    - Benefits: By implementing using agentic ai, the system shall saves time and automate routine communication.
    - Drawbacks: Risk of misclassification or sending inaccurate responses.

7. **Medical Symptom Assessment**: A system designed to evaluate described symptoms and provide health information

Your Answer Here: **Partially appropriate for agentic AI**

- **Reasoning:** Useful for triage and guidance, but requires human oversight.
- **Benefits:** 24/7 access, early detection support.
- **Drawbacks:** Ethical risks, accuracy and liability concerns.

8. **Financial Investment Advisor**: A system providing investment recommendations based on market conditions and user goals

Your Answer Here: **Appropriate for agentic AI**

- **Reasoning:** Needs reasoning over market trends, user goals, and risk.
- **Benefits:** Personalized insights, real-time decision support.
- **Drawbacks:** Requires regulation, explainability, and bias control.

Part 3: Repository Analysis

Review the codebase at https://github.com/readytensor/rt-repo-assessment. Based on the definition of agentic AI systems we've covered in the lecture, would you classify this solution as agentic? Why or why not? Support your analysis with specific examples from the codebase that either align with or deviate from the key characteristics of AI agents we've discussed.

Your Answer Here:

From our lecture, an "agentic AI system" typically includes the following features:

- Autonomy: the system can plan, decide, and act without human-in the-loop on every step.
- Multi-step reasoning / orchestration: it reasons through tasks, breaks them down, calls tools or sub-agents, then aggregates results.
- Tool usage / environment interaction: it uses APIs, external services or executes actions, not just pure prediction.
- Memory / Context: it maintains state or history across steps/turns, to adapt or improve.
- Goal-driven behaviour: it pursues objectives (not just single-prompt outcomes) and may adjust the plan or reflect on outcomes.

Does the rt-repo-assessment solution qualify as agentic?

I say: **No — it does not currently meet full agentic AI criteria*.

Here is why, with specific examples from the codebase plus where it aligns and where it falls short.

What aligns with agentic traits

- The repository description: "This project implements an assessment tool to evaluate the quality of AI/ML project repositories using LLMs and rule based methods."
  - It uses an LLM to evaluate repository metadata — so there is some reasoning work going on (i.e., evaluating quality, making judgments).
  - It likely uses external tool(s) (LLM, rule engine) – so there is some "tool usage" component.
- It seems to serve a **goal**: assess repository quality. Thus there is a target objective.

Where it falls short for agentic-level architecture

1. **Lack of multi-step orchestration**
   - There is no evidence that the system breaks down tasks, delegates to sub-agents/modules, or adapts its strategy over multiple steps.
   - It appears to operate as: submit repository → evaluate via LLM/rules → output assessment. That is essentially single-turn.
2. **Limited autonomy / environment interaction**
   - While it interacts with LLM and perhaps file system/metadata, there is no evidence of automatic iterative tool calls, external APIs for branching, or agent monitoring and re-acting.
   - There is no indication that the system executes actions in the environment (e.g., modify repo, call CI/CD, schedule retests).
3. **Memory / state across sessions**
   - There is no clear mechanism for remembering past assessments, adapting its internal model over time based on outcomes, or learning from prior repository evaluations to inform future assessments.
   - This means it lacks persistent context or self-improvement loop.
4. **Goal-driven, self-correcting behaviour**
   - Although its goal is to evaluate repository quality, I did not see a mechanism for reflecting on its assessment (e.g., “my previous assessment was wrong, update rule”), or changing strategy if quality is low.
   - It’s more of a static tool than a dynamic agent.

Part 4: Future of Work Analysis

Based on what you have understood about agentic AI in Week 1, do you think the following job titles will be eliminated in the next 10 years due to agentic AI? Explain your reasoning for each:

- Software Engineer
- Data Analyst
- Data Scientist
- ML Engineer
- AI Engineer
- Data Engineer
- ML-Ops Engineer

Your Answer Here:
1. Software Engineer: Not eliminated but transformed

- **Reasoning:** Agentic AI (e.g., GitHub Copilot, Devin) will automate repetitive coding, debugging, and documentation.
- **Impact:** Engineers will shift from writing code to designing, reviewing, and orchestrating AI-generated components.
- **Future role:** AI-assisted software architect — focusing on logic, security, and integration.

2. Data Analyst: Significantly reduced and partially automated

- **Reasoning:** Agentic systems can autonomously query data (text-to-SQL), generate dashboards, and summarize insights.
- **Impact:** Routine analytics and reporting tasks will be handled by agents.
- **Future role:** Analysts will focus on business context, validation, and storytelling with AI outputs.

3. Data Scientist: Not eliminated but augmented

- **Reasoning:** Agents can automate model training and basic feature engineering, but not problem framing or ethical validation.
- **Impact:** Data scientists will evolve into AI supervisors — defining objectives, ensuring interpretability, and governing AI behavior.
- **Future role:** AI outcome auditor / experiment designer

ML Engineer: Partially automated

- **Reasoning:** Agentic AI can self-build pipelines, tune hyperparameters, and deploy models.
- **Impact:** Engineers will focus on custom architectures, scaling, compliance, and LLM agent integration.
- **Future role:** AI workflow engineer overseeing automated ML systems.

AI Engineer: Expands- Not eliminated

- **Reasoning:** Agentic systems increase demand for AI engineers to design, orchestrate, and govern multi-agent ecosystems.
- **Impact:** They'll focus on agent orchestration frameworks, reasoning optimization, and safety layers.
- **Future role:** Agent systems architect or AI capability designer.

Data Engineer: Transformed -less manual ETL, validation but more orchestration

- **Reasoning:** Agentic AI will automate schema discovery, data cleaning, and integration.
- **Impact:** Engineers will manage data contracts, observability, and governance for autonomous systems.
- **Future role:** Data infrastructure strategist.

ML-Ops Engineer: Evolving -not eliminated

- **Reasoning:** Many CI/CD and monitoring workflows will be agent-driven.
- **Impact:** Focus shifts from manual deployment to trust, compliance, rollback safety, and agent lifecycle management.
- **Future role:** AI-Ops / Agent-Ops Engineer.

Part 5: Choosing the Right Architecture for an Automated Peer Review System

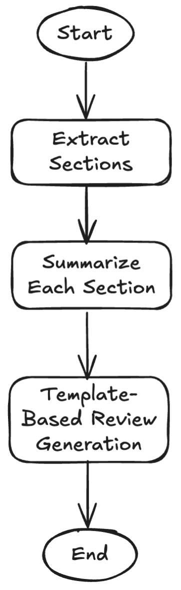
Background:

Imagine you are tasked with building a system to automatically review academic papers for a journal submission platform. The system should read the paper, identify strengths and weaknesses, and generate a structured review.

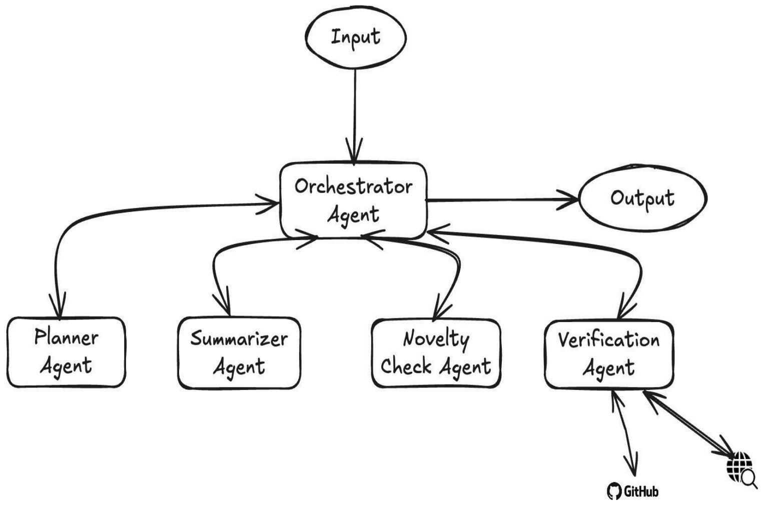
Exercise:

Below are two different architectural designs for this system:

WORKFLOW ARCHITECTURE



AGENTIC ARCHITECTURE



- One is a non-agentic workflow,
- One is a multi-agent system,

Your task:

- Carefully study the two architectures.
- Based on what you learned about Workflows vs. Agents, choose the architecture you think will perform best for this task.
- There is no right or wrong answer - we want you to think strategically about tradeoffs like flexibility, transparency, cost, complexity, and reliability.
- You can also propose another architecture that you think would be best for this case study.

In 2–8 sentences, explain why you made your choice. To better help you answer this question, refer to this article: Building effective agents

Your Answer Here:

For an academic paper review system, a multi-agent architecture would be the best approach because the task involves reasoning, evaluation, and judgment that is beyond the simple sequential processing. The planner can dynamically decide which sections need deeper analysis such as methodology vs. results. The summarizer agent ensures focused understanding of each section having relevant context in the summary. The novelty checker agent compares content against prior literature or existing databases to detect originality and duplication. The verification agent validates factual accuracy and prevents hallucinations, improving reliability. The orchestrator coordinates these agents, integrating their outputs into a coherent, structured review. While a workflow is simpler, the multi-agent system offers flexibility, depth, and critical reasoning, which are essential for academic paper evaluation where judgment, cross-checking, and validation matter more than linear execution.