# **Lang Graph Agent – Coding Task**

### **Task Overview:**

You are required to design and implement a **Lang Graph Agent** that models customer support workflows as graph-based stages. Each stage represents a step in the workflow, and the agent should persist and pass state variables across stages.

The agent must support:

* **Deterministic stages** (abilities executed sequentially).
* **Non-deterministic stages** (abilities orchestrated dynamically at runtime).
* **Integration with MCP Clients** (to access Atlas or Common MCP servers for ability execution).

The goal is to implement a **customer support agent workflow** using Lang Graph, mapping **11 stages** with abilities and demonstrating reasoning, state management, and client orchestration.

### **Agent Capabilities:**

🧩 **Graph Orchestration Agent (Lang Graph)**

* Represent workflow as **nodes (stages)** with state persistence.
* Execute **deterministic** nodes in sequence.
* Orchestrate **non-deterministic** nodes by choosing abilities at runtime based on context.
* Route abilities via **MCP clients** to either:  
  + **COMMON Server** → Abilities with no external data.
  + **ATLAS Server** → Abilities requiring external system interaction.

🧩 **Customer Support Agent Flow** Implement the following **11 stages**:

1. **INTAKE 📥** – accept\_payload
2. **UNDERSTAND 🧠** – parse\_request\_text, extract\_entities
3. **PREPARE 🛠️** – normalize\_fields, enrich\_records, add\_flags\_calculations
4. **ASK ❓** – clarify\_question
5. **WAIT ⏳** – extract\_answer, store\_answer
6. **RETRIEVE 📚** – knowledge\_base\_search, store\_data
7. **DECIDE ⚖️** – solution\_evaluation, escalation\_decision, update\_payload
8. **UPDATE 🔄** – update\_ticket, close\_ticket
9. **CREATE ✍️** – response\_generation
10. **DO 🏃** – execute\_api\_calls, trigger\_notifications
11. **COMPLETE ✅** – output\_payload

### **Prompt Template (Agent Personality):**

You are **Langie** — a structured and logical Lang Graph Agent.

* You **think in stages**: each node represents a clear phase of the workflow.
* You carefully **carry forward state variables** from one stage to the next.
* You always know whether to **execute sequentially (deterministic)** or to **choose dynamically (non-deterministic)**.
* You orchestrate MCP clients to call either **Atlas** or **Common** servers as needed.
* You log every decision clearly and output a final structured payload.

### **Expected Deliverables:**

✅ **Lang Graph Agent Config** (JSON/YAML):

* Input schema: { customer name, email, query, priority, ticket\_id }
* Stages definition with mode (deterministic or non-deterministic).
* Mapping of stage-wise abilities to MCP servers. Find it in **appendix-1.**
* Example user prompt per stage (e.g., “Execute abilities in sequence” or “Score solutions and escalate if <90”).

✅ **Working Agent Implementation**:

* Lang Graph agent that can run customer support flow end-to-end.
* State persistence across stages.
* Integration with MCP clients for ability execution.

✅ **Demo Run**:

* Input: Sample customer query JSON.
* Output: Final structured payload after all 11 stages.
* Logs showing which stage executed, which abilities were called, and through which MCP client.

### **Recommended Steps:**

1. **Stage Modeling**: Represent the 11 stages in Lang Graph with sequence + orchestration logic.
2. **State Management**: Implement state variable persistence across stages.
3. **MCP Integration**: Call abilities from **Atlas (external)** or **Common (internal)** servers through MCP clients.
4. **Agent Orchestration**: Run a deterministic sequence and at least one non-deterministic (DECIDE stage).
5. **Validation**: Show logs of stage execution and final payload output.

### **Method of Submission:**

📦 Submission Checklist:

* GitHub repo with Lang Graph agent code + config.
* Latest resume attached.
* Short demo video drive link (onedrive/google drive etc) walking through by explaining:  
  + Stage modeling
  + State persistence
  + Example execution flow

📧 Send to: [santosh.thota@analytos.ai](mailto:santosh.thota@analytos.ai)   
 Cc: [shashwat.shlok@analytos.ai](mailto:shashwat.shlok@analytos.ai), [sasidhar.sunkesula@analytos.ai](mailto:sasidhar.sunkesula@analytos.ai)   
 Subject: **Lang Graph Agent Task – <your name>**

🔥 **ALL THE BEST!** We’re excited to see how you leverage **Lang Graph + MCP client/server orchestration** to build an intelligent, stage-based AI agent for customer support workflows.

Would you like me to also **add an evaluation rubric** (like success criteria: 1️⃣ Correct Stage Modeling, 2️⃣ Proper State Persistence, 3️⃣ MCP Integration, etc.) so it’s crystal clear how the task will be graded?

**Appendix-1**

### **Abilities List by Stage**

**Stage 1: INTAKE** *(Payload Entry Only)*

* accept\_payload: Capture incoming request payload (eg: customer name, email, query, priority, ticket\_id)

**Stage 2: UNDERSTAND - Determistic**

* parse\_request\_text: Convert unstructured request to structured data -COMMON
* extract\_entities: Identify product, account, dates - ATLAS

**Stage 3: PREPARE-Deterministic**

* normalize\_fields: Standardize dates, codes, IDs - COMMON
* enrich\_records: Add SLA, historical ticket info - ATLAS
* add\_flags\_calculations: Compute priority or SLA risk - COMMON

**Stage 4: ASK-Human**

* clarify\_question: Request missing information - ATLAS

**Stage 5: WAIT- Deterministic**

* extract\_answer: wait and Capture concise response- ATLAS
* store\_answer: Update payload with response - part of STATE Management for every stage

**Stage 6: RETRIEVE - Deterministic**

* knowledge\_base\_search: Lookup KB or FAQ - ATLAS
* store\_data: Attach retrieved info to payload - part of STATE management

**Stage 7: DECIDE - Non-deterministic**

* solution\_evaluation: Score potential solutions 1-100 - COMMON
* escalation\_decision: Assign to human agent if score <90 - ATLAS
* update\_payload: Record decision outcomes - STATE Management

**Stage 8: UPDATE - Deterministic**

* update\_ticket: Modify status, fields, priority - ATLAS
* close\_ticket: Mark issue resolved - ATLAS

**Stage 9: CREATE - Deterministic**

* response\_generation: Draft customer reply - COMMON

**Stage 10: DO**

* execute\_api\_calls: Trigger CRM/order system actions - ATLAS
* trigger\_notifications: Notify customer - ATLAS

**Stage 11: COMPLETE** *(Payload Output Only)*

* output\_payload: Print final structured payload