### 🚩 **Detailed Bug Report: Anomalies in Large Context Agents & Context Grounding (GPT-4 / GPT-5 Preview)**

Component: UiPath Agents (Studio & Runtime)

Models: GPT-4, GPT-5 (Preview)

Feature: Context Grounding (DeepRAG)

Executive Summary:

We are conducting stress tests on Agents using high-volume inputs via Context Grounding. We have identified four distinct stability issues: (1) Premature generation stops without errors on GPT-4, (2) Silent failures (empty payload) on GPT-5 Preview with high context, (3) JSON serialization breaking down on long outputs, and (4) Intermittent retrieval failures (False Negatives) on known data.

#### 🔴 **1. GPT-4: Premature Completion & Logic Degradation (Non-Truncation)**

* **Scenario:** Running a use case where the input context is high, but below the hard limit.
* **Observed Behavior:** The Agent does not hit a "Max Token" error, nor does it cut off mid-sentence (truncation). Instead, the model acts as if it has "given up" on the complexity of the prompt. It provides a severely abbreviated, "lazy" response that ignores the depth required by the prompt instructions.
* **The Bug:** The system marks this transaction as **Successful**. There is no warning flag indicating that the output quality was compromised.
* **Impact:** **Silent Logic Failure.** The failure is **undetectable programmatically** because the Agent status returns Success. The degradation is only visible when manually reviewing the **Debug / Agent Traces**, where the output clearly fails to meet the prompt requirements despite the valid status code.

#### 🔴 **2. GPT-5 (Preview): Silent Failure / Empty Payload at High Context**

* **Scenario:** Switched to GPT-5 Preview (128k context) to accommodate larger datasets.
* **Telemetry:**
  + Total Token Consumption: ~87,000 / 128,000 (well within limits).
  + RAG Tool Status: Successful (Context retrieved).
* **Observed Behavior:** The Agent returns a valid status, but the output payload is literally output = "[]".
* **Analysis:** This appears to be a generation failure after the retrieval step. The model ingests the 87k tokens but fails to synthesize a response, defaulting to an empty array string.
* **Impact:** **False Positive Success.** Automated workflows cannot catch this exception via standard Try/Catch blocks because no exception is thrown. The issue is only detected by manually inspecting the **Output Payload** in the debugger or logs, which reveals the empty JSON array [] despite the successful transaction state.

#### 🟠 **3. JSON Serialization Drift / Schema Violation on Long Outputs**

* **Configuration:** The System Prompt explicitly instructs the Agent to return a serialized JSON String representing the data.
* **Observed Inconsistency:**
  + **Short Outputs:** The Agent correctly follows instructions and returns a valid String (e.g., "{...}").
  + **Long Outputs:** When the response requires more tokens (even if total consumption is low, e.g., <20k), the Agent ignores the "String" constraint. It forces the output into a raw JSONArray structure (or a file object), breaking the expected schema.
* **Workaround Implemented:** We implemented a "Chunking/Looping" logic to request data in small batches.
  + *Constraint:* **We are aware this looping architecture is not best practice.** However, given tight project time constraints and the blocking nature of the serialization bug, this was the only viable option to stabilize the output.
  + *Result:* This works "consistently" but increases execution time.
* **The Bug:** The Agent’s adherence to output formatting rules (System Prompt) degrades significantly as the output length increases, even when token limits are not threatened.

#### 🟡 **4. Context Grounding: Non-Deterministic Retrieval & "False Negative" Loops**

* **Scenario:**
  + The Agent is using **Context Grounding (DeepRAG)** with citations enabled.
  + **System Prompt Rule:** We have enforced a strict rule to prevent hallucinations: *"If the answer is not found in the context, return a specific 'No Answer' JSON object with Confidence: 0."*
  + **Input:** Questions that are known to have clear, verbatim answers in the source documents.
* **Observed Behavior (Intermittent):**
  + The behavior is **non-deterministic**. For the *exact same input*, the Agent sometimes fails to map the citation, leading it to trigger the "No Answer" fallback.
* **Root Cause Indicators (Internal ID Leakage):**
  + In the failure instances, we have observed the Agent momentarily hallucinating or leaking internal Vector/Chunk IDs (e.g., [xU4kBF, p.X], (GyxdSY)) in the raw text stream before it settles on the "No Answer" output.
  + It appears the Agent retrieves the correct context chunk but fails to map the internal Vector ID to a user-friendly Citation, subsequently failing the schema validation and defaulting to "No Answer."
* **Frequency Note:** This issue occurs **mostly on the first run** of the Agent. This suggests the failure is related to initialization or cold-start retrieval, allowing us to rule out the "looping workaround" (from Issue #3) as the root cause.
* Resulting Artifact (False Negative):  
  Instead of a partial answer, the system outputs the following "Failure JSON," effectively hiding valid data from the business process:  
  JSON  
  {  
   "CategoryName": "TestCategory",  
   "No." : "1",  
   "Question": "Test Question Goes Here",  
   "Information":{  
   "Answer":"No Answer was found.",  
   "Confidence":0,  
   "Sources":[],  
   "Exception":null  
   }  
  }