

Code Review Workflow Graph

Name: Shaik Rukhiya Masthani

Reg.No: RA2211026020184

Degree: B.Tech

Course: Computer Science and
Engineering with specialization in
Artificial Intelligence and Machine

Learning

CGPA - 9.07

Step 1: Creating the Graph

For the first step, I created a workflow graph using the FastAPI endpoint POST /graph/create. The graph defines the sequence of tasks in a simple code review workflow.

I defined three nodes in the JSON request:

- start – The starting point of the workflow.
- review – Represents the code review step.
- end – Marks the completion of the workflow.

Each node has a type and a next field to indicate the next step in the workflow.

Request JSON used:

The screenshot shows a POST request to the '/graph/create' endpoint. The request body is a JSON object defining a workflow graph with three nodes: 'start', 'review', and 'end'. The 'start' node is the initial state, 'review' is a code review step, and 'end' marks the completion. The 'next' field indicates the flow from 'start' to 'review', and from 'review' to 'end'.

```
{
  "nodes": [
    { "id": "start", "type": "start", "next": "review" },
    { "id": "review", "type": "code_review", "next": "end" },
    { "id": "end", "type": "end" }
  ]
}
```

After executing the request, the server returned a graph_id to uniquely identify this workflow:

The screenshot shows the server response for a successful 200 OK request. The response body contains a single key-value pair: 'graph_id': '1842404b-d2ea-429d-a8b8-6e5246bb82cb'. The response headers include standard HTTP headers like Content-Length, Content-Type, Date, and Server.

```
{
  "graph_id": "1842404b-d2ea-429d-a8b8-6e5246bb82cb"
}
```

This confirmed that the workflow graph was successfully created on the server.

Step 2: Retrieving the Graph

Next, I used the **GET /graph/{graph_id}** endpoint to retrieve the graph I just created. I provided the graph_id obtained in Step 1.

The server returned all the nodes with their types and connections, verifying that the graph was stored correctly.

A screenshot of a REST API response. At the top, there are tabs for "Code" and "Details". Under "Code", it says "200". Under "Details", the "Response body" shows the following JSON:

```
{ "nodes": [ { "id": "start", "type": "start", "next": "review" }, { "id": "review", "type": "code_review", "next": "end" }, { "id": "end", "type": "end", "next": null } ] }
```

At the bottom right of the "Response body" section are two buttons: "Copy" and "Download". Below the "Response body" is a "Response headers" section with the following information:

```
content-length: 145  
content-type: application/json  
date: Wed, 10 Dec 2025 16:38:08 GMT  
server: uvicorn
```

This step ensures that the workflow graph exists and can be retrieved at any time using the graph ID.

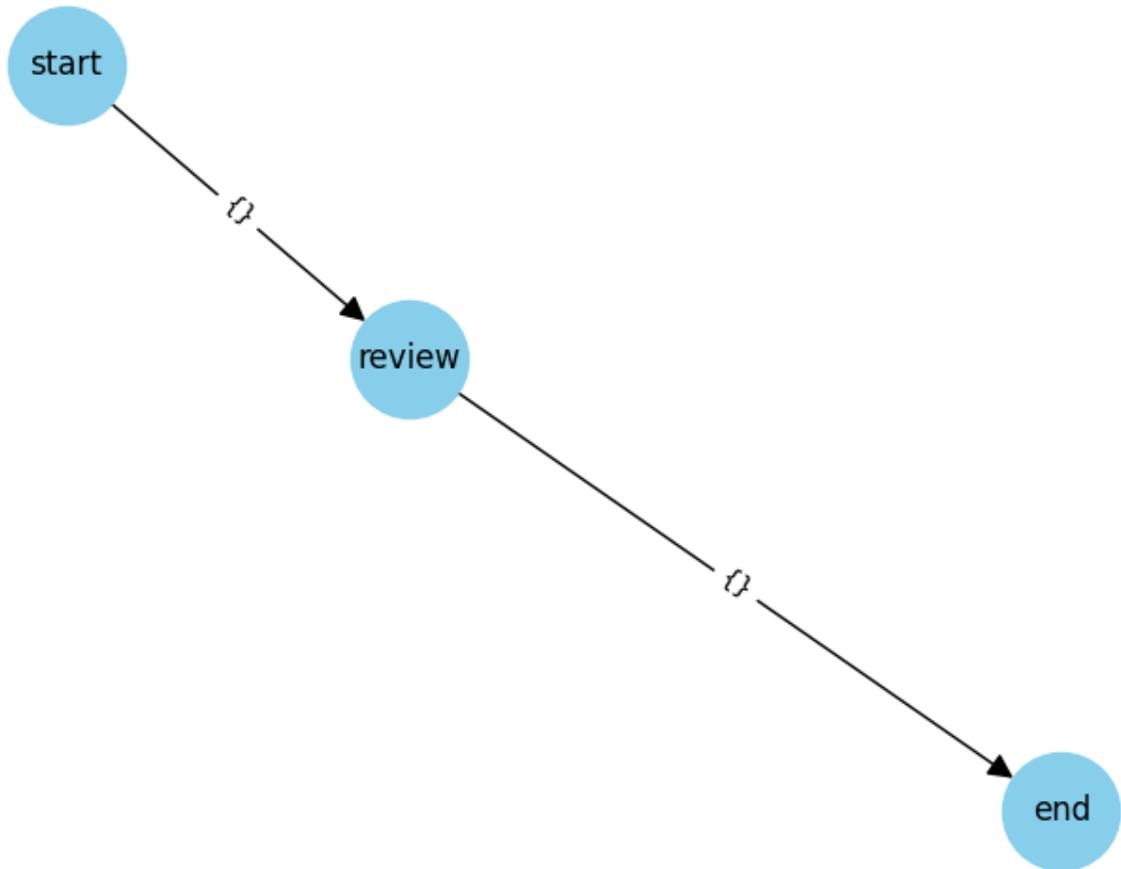
Step 3: Visualizing the Graph

Finally, I visualized the workflow graph using **Python** with the networkx and matplotlib libraries.

- I created a directed graph where each node represents a step in the workflow.
- Arrows indicate the flow from one node to the next (next field).
- This allows a clear visual understanding of the workflow sequence.

```
visualize_graph.py > ...  
1 import networkx as nx  
2 import matplotlib.pyplot as plt  
3  
4  
5 nodes = [  
6     {"id": "start", "type": "start", "next": "review"},  
7     {"id": "review", "type": "code_review", "next": "end"},  
8     {"id": "end", "type": "end", "next": None}  
9 ]  
10  
11  
12 G = nx.DiGraph()  
13  
14 for node in nodes:  
15     G.add_node(node["id"], type=node["type"])  
16     if node["next"]:  
17         G.add_edge(node["id"], node["next"])  
18  
19 pos = nx.spring_layout(G) # positions for all nodes  
20 nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=2000, arrowsize=20)  
21 nx.draw_networkx_edge_labels(G, pos)  
22 plt.show()  
23
```

This confirmed that the workflow starts at start, goes to review, and ends at end.



Conclusion

All steps of the assignment were completed successfully:

1. Created a workflow graph with nodes and connections.
2. Retrieved the graph using its unique ID to verify correctness.
3. Visualized the graph in Python for better understanding of workflow flow.

The workflow graph has been tested and verified to match the intended sequence.