perplexity

Implementing Real-Time Agent Reasoning Visualization in Streamlit with DeerFlow Integration

Overcoming Streamlit's UI Limitations for Dynamic Agent Feedback

When implementing AI research workflows with Streamlit and DeerFlow, displaying real-time agent reasoning presents unique challenges due to Streamlit's execution model. This analysis explores technical solutions grounded in Streamlit's API capabilities and DeerFlow's multi-agent architecture, supported by 20 research sources.

Streamlit Execution Model Constraints

Streamlit's script rerun-on-interaction paradigm creates three core challenges for real-time agent feedback:

1. **State Persistence**: Agent reasoning steps require preservation across script executions[3][15]
2. **Concurrent Processing**: Long-running research tasks block UI updates[4][9]
3. **Dynamic Rendering**: Traditional widgets don't support incremental updates[17][19]

```
# Baseline implementation showing blocking execution
def research_flow():
    st.write("Starting research...")
    result = deerflow_agent.execute()  # Blocks UI
    st.write(result)
```

Solution Architecture

Hybrid Execution Model

Implement a producer-consumer pattern with Streamlit's session state:

```
if 'research_queue' not in st.session_state:
    st.session_state.research_queue = []
    st.session_state.current_step = None

def agent_callback(step):
    st.session_state.research_queue.append(step)

def ui_consumer():
    while st.session_state.research_queue:
        step = st.session_state.research_queue.pop(0)
```

```
        st.session_state.current_step = step
        # Rerun to update UI
        st.rerun()

# DeerFlow integration
deerflow_agent.set_callback(agent_callback)
```

Real-Time Visualization Techniques

1. Streaming Output with write_stream

Utilize Streamlit's native streaming API for incremental updates[7][10]:

```
def thinking_generator():
    yield "Starting research...\n"
    for step in deerflow_agent.steps():
        yield f"Processing: {step}\n"
        time.sleep(0.1)

with st.status("Research Progress", expanded=True):
    st.write_stream(thinking_generator())
```

2. Fragment-Based Partial Updates

Isolate dynamic components using experimental fragments[3][19]:

```
@st.experimental_fragment
def thinking_fragment():
    placeholder = st.empty()
    while deerflow_agent.is_running():
        placeholder.markdown(f"""
            **Current Step**: {deerflow_agent.current_step}
            **Elapsed**: {deerflow_agent.elapsed_time}s
        """)
        time.sleep(0.1)
```

3. Animated Status Indicators

Combine spinner with progress visualization[8][9]:

```
with st.spinner("Analyzing data..."):
    progress_bar = st.progress(0)
    for i in range(100):
        deerflow_agent.process_chunk(i)
        progress_bar.progress(i + 1)
        time.sleep(0.05)
```

DeerFlow Agent Integration

Custom Callback Handlers

Implement LangChain-style callbacks for step capture[6][10]:

```
from langchain.callbacks import BaseCallbackHandler

class StreamlitThinkingHandler(BaseCallbackHandler):
    def on_agent_action(self, action, **kwargs):
        st.session_state.research_steps.append(action.log)

    def on_llm_new_token(self, token, **kwargs):
        st.session_state.current_token = token
        st.rerun()

# Agent configuration
deerflow_agent.add_callback(StreamlitThinkingHandler())
```

Multi-Agent State Management

Leverage DeerFlow's architecture for distributed reasoning tracking[11][14]:

```
class ResearchCoordinator:
    def __init__(self):
        self.agents = {
            'planner': PlannerAgent(),
            'researcher': ResearchAgent(),
            'analyst': AnalysisAgent()

<div style="text-align: center">⁂</div>

[^1]: https://github.com/bytedance/deer-flow
[^2]: https://www.youtube.com/watch?v=bbupey9_UAQ
[^3]: https://discuss.streamlit.io/t/placeholder-keyword-argument/49905
[^4]: https://discuss.streamlit.io/t/how-to-run-a-background-task-in-streamlit-and-notify
[^5]: https://docs.streamlit.io/develop/api-reference/chat
[^6]: https://python.langchain.com/docs/concepts/callbacks/
[^7]: https://docs.streamlit.io/develop/api-reference/write-magic/st.write_stream
[^8]: https://docs.streamlit.io/develop/api-reference/status/st.status
[^9]: https://docs.streamlit.io/develop/api-reference/status/st.spinner
[^10]: https://python.langchain.com/docs/integrations/callbacks/streamlit/
[^11]: https://aiagent.marktechpost.com/post/meet-deerflow-an-open-source-deep-research-a
[^12]: https://github.com/bytedance/deer-flow/activity
[^13]: https://discuss.streamlit.io/t/input-test-placeholder/53576
[^14]: https://www.linkedin.com/pulse/deerflow-modular-multi-agent-framework-deep-researc
[^15]: https://discuss.streamlit.io/t/is-there-any-way-to-use-random-placeholder-for-text
[^16]: https://www.marktechpost.com/2025/05/09/bytedance-open-sources-deerflow-a-modular-
[^17]: https://franekjemiolo.pl/how-handle-streaming-data-streamlit/
[^18]: https://learnopoly.com/bytedand-open-source-deerflow-a-modular-multi-aging-frame-f
[^19]: https://discuss.streamlit.io/t/any-workaround-for-placeholder-of-interactive-widge
[^20]: https://www.reddit.com/r/LocalLLaMA/comments/1chkl62/langchain_vs_llamaindex_vs_cr
[^21]: https://www.linkedin.com/posts/saeed-kasmani_github-bytedancedeer-flow-deerflow-is
```

[^22]: https://www.instagram.com/reel/DJscxLMhwPS/
[^23]: https://github.com/langchain-ai/langchain/issues/3263
[^24]: https://python.langchain.com/docs/how_to/streaming/
[^25]: https://www.youtube.com/watch?v=fni87YjoQt8
[^26]: https://www.cnblogs.com/smartloli/p/17796744.html
[^27]: https://github.com/whitphx/streamlit-webrtc
[^28]: https://www.youtube.com/watch?v=crEszDkC-O8
[^29]: https://discuss.streamlit.io/t/progress-bar-issue-in-cached-function/47011
[^30]: https://blog.streamlit.io/how-to-build-a-real-time-live-dashboard-with-streamlit/
[^31]: https://www.reddit.com/r/LangChain/comments/1eduao5/how_can_i_stream_only_the_fina
[^32]: https://python.langchain.com/docs/integrations/memory/streamlit_chat_message_histo
[^33]: https://discuss.streamlit.io/t/spinner-doesnt-show-and-then-everything-updates-onc
[^34]: https://docs.streamlit.io/develop/concepts/custom-components/intro
[^35]: https://discuss.streamlit.io/t/how-to-make-placeholder-for-components-iframe/18659
[^36]: https://js.langchain.com/docs/concepts/callbacks/
[^37]: https://docs.chainlit.io/api-reference/integrations/langchain
[^38]: https://api.python.langchain.com/en/latest/callbacks/langchain_core.callbacks.stre
[^39]: https://www.reddit.com/r/LangChain/comments/18ogw3p/how_do_callbacks_for_streaming
[^40]: https://python.langchain.com/docs/how_to/sequence/
[^41]: https://python.langchain.com/api_reference/core/callbacks.html
[^42]: https://js.langchain.com/docs/how_to/qa_streaming
[^43]: https://python.langchain.com/api_reference/community/callbacks/langchain_community
[^44]: https://api.python.langchain.com/en/latest/community/callbacks/langchain_community
[^45]: https://python.langchain.com/docs/how_to/custom_callbacks/
[^46]: https://www.53ai.com/news/langchain/280.html
[^47]: https://api.python.langchain.com/en/latest/_modules/langchain_openai/chat_models/b
[^48]: https://discuss.streamlit.io/t/st-write-stream-with-formatted-html-text/63405
[^49]: https://stackoverflow.com/questions/78091501/streamlit-st-write-stream-function-re
[^50]: https://github.com/streamlit/streamlit/issues/8166
[^51]: https://www.aidoczh.com/streamlit/develop/api-reference/write-magic/st.write_strea
[^52]: https://www.youtube.com/watch?v=qOn1vUvA5iA
[^53]: https://js2iiu.com/2025/01/13/streamlitst-write_stream/
[^54]: https://docs.streamlit.io/develop/api-reference/status
[^55]: https://discuss.streamlit.io/t/how-to-use-st-progress-to-show-progress-of-function
[^56]: https://discuss.streamlit.io/t/progress-bar/22664
[^57]: https://discuss.streamlit.io/t/progress-bar-not-incrementally-updating/79721
[^58]: https://www.youtube.com/watch?v=26uhRaQ2IG0
[^59]: https://stackoverflow.com/questions/75065231/realtime-data-update-to-streamlit-fro
[^60]: https://discuss.streamlit.io/t/does-streamlit-support-real-time-data/12687
[^61]: https://discuss.streamlit.io/t/custom-write-stream/71366
[^62]: https://discuss.streamlit.io/t/multiple-simultaneous-write-stream-elements-running
[^63]: https://dev.to/jamesbmour/streamlit-part-8-status-elements-mc1
[^64]: https://discuss.streamlit.io/t/streaming-in-chat-but-without-typewriter-effect/631
[^65]: https://discuss.streamlit.io/t/using-write-stream-with-langchain-llm-streaming-sho
[^66]: https://alejandro-ao.com/how-to-use-streaming-in-langchain-and-streamlit/
[^67]: https://www.youtube.com/watch?v=zKGeRWjJlTU
[^68]: https://blog.streamlit.io/langchain-streamlit/
[^69]: https://github.com/hwchase17/langchain-streamlit-template/issues/3
[^70]: https://github.com/streamlit/StreamlitLangChain/blob/main/streaming_demo.py