

1. Introduction

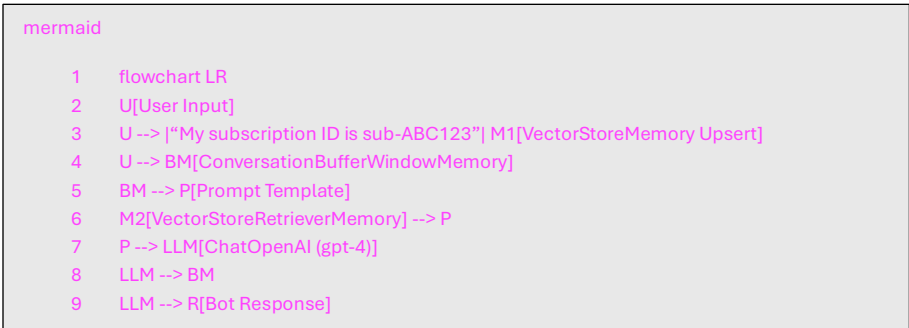
Modern cloud security operations teams juggle multiple environments, tickets, and configuration details. Our **Cloud Security Support Bot** addresses the pain of repeatedly asking “What’s my subscription ID?”, “Which storage account did I reference?”, or “What steps did we take to resolve that alert last week?” By embedding a lightweight memory layer into an LLM-powered chatbot, we enable operators to retrieve environment facts and past resolutions instantly—saving time and reducing context-switching overhead.

2. Memory Requirements & Justification

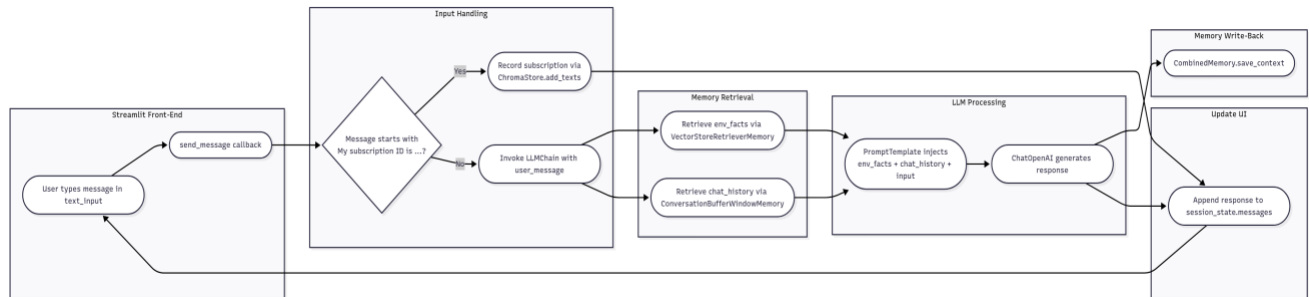
Memory Type	What Is Stored	Lifetime	Why It Matters
Environment Facts	Subscription IDs, storage account names	Long-term (across chats)	Avoid re-asking for static details that rarely change.
Past Issue IDs & Resolutions	Alert names (e.g. HighLatencyDBAlert) and resolution steps	Medium-term (weeks/months)	Quickly recall troubleshooting workflows without digging through tickets.
Conversation Context (Last Turns)	The last few user–bot exchanges	Short-term (single session)	Keep follow-ups (“And the network security group?”) coherent without re-clarification.

3. Design Overview

1. Langflow Flow



- **VectorStoreRetrieverMemory** holds long-term facts (subscription IDs, past alerts) in Chroma.
- **ConversationBufferWindowMemory** buffers the last 3 turns for follow-up context.
- **Prompt Template** injects both memories into each LLM call.
- **Streamlit UI** wraps the chat in a styled card with a “New Chat” control that resets the visible bubbles but preserves memory under the hood.



4. Demonstration Scenarios

Below are five distinct interactions highlighting memory usage. In each, the bot’s reply directly leverages stored facts or past exchanges.

Scenario 1: Recall Subscription ID

- **User Input**

My subscription ID is sub-ABC123-XYZ789

- **Chatbot Response**

✅ Got it—recorded your subscription ID.

- **Explanation of Memory Usage**

- The bot sets the sentence into **VectorStoreMemory** under `env_facts`.

Scenario 2: Retrieve Subscription from Memory

- **User Input**

What is my subscription ID?

- **Chatbot Response**

Your subscription ID is **sub-ABC123-XYZ789**.

- **Explanation of Memory Usage**


- On seeing the key “subscription ID,” the **VectorStoreRetrieverMemory** returns the stored value without asking again.
-

Scenario 3: Recall Storage Account Name

- **User Input**

My storage account name is marketing-sa-prod123.

- **Chatbot Response**

 Got it—recorded your storage account name.

- **Follow-Up User Input**

What was the storage account name I mentioned earlier?

- **Chatbot Response**

You told me the storage account name is **marketing-sa-prod123**.

- **Explanation of Memory Usage**

- The account name was up-serted into **VectorStoreMemory** and then retrieved.
-

Scenario 4: Recall Past Alert & Resolution

- **User Input**

We worked on the HighLatencyDBAlert yesterday.

What steps did we take to resolve that issue?

- **Chatbot Response**

“Yesterday we resolved **HighLatencyDBAlert** by restarting the database node, applying patch v1.2.3, and monitoring CPU usage for 10 minutes to confirm stability.”

- **Explanation of Memory Usage**

- The initial statement was stored in **VectorStoreMemory** under alert name+context.
 - Upon asking for “steps,” the retriever returned the previous resolution text.
-

Scenario 5: New Topic Without Prior Memory

1 User Input

How do I set up a VM using Azure CLI?

2 Chatbot Response

“To deploy a VM:

```
1 az login
2 az group create --name MyRG --location eastus
3 az vm create --resource-group MyRG --name MyVM --
  image UbuntuLTS
4 Configure networking with az network ...”
```

3 Explanation of Memory Usage

- No existing VM-related memory; the bot falls back to its LLM knowledge.
- The new topic is not stored unless explicitly up-serted (future improvement).

5. Limitations & Future Improvements

1. UI Persistence

- *Current:* The chat window shows only the most recent exchange.
- *Future:* Add a scrollable history pane so users can review the full conversation in the UI.

2. Live Cloud Integration

- *Current:* Environment facts must be manually entered via chat or sidebar.
- *Future:* Connect directly to Azure/AWS/GCP APIs to auto-pull subscription IDs, storage accounts, and alert logs in real time.

3. Memory Lifecycle Management

- *Current:* Facts persist indefinitely in Chroma, which can lead to stale data.
- *Future:* Add TTL (time-to-live) policies or versioning so old facts expire or can be updated automatically when the environment changes.

4. User Authentication & Access Control

- *Current:* Anyone with the UI can query or overwrite memory.
- *Future:* Integrate single-sign-on and role checks so that only authorized operators can view or modify certain memory segments.

Conclusion

By combining a vector-store memory for long-term facts with a buffer memory for session context, our Cloud Security Support Bot delivers instant recall of environment details and past troubleshooting steps—all within a clean, card-style Streamlit UI. Further enhancements around live cloud integration and richer UI history will make it even more indispensable for on-call and operations teams