1.DEEPSEEK-CODER:1.3B using ollama pull command

```
import streamlit as st
import ollama
st.set_page_config(page_title="DeepSeek Chatbot", page_icon="🤖", layout="wide")
st.title("💬 DeepSeek Chatbot (Ollama)")
# --- Initialize session state ---
if "chats" not in st.session_state:
    st.session_state.chats = []  # list of chat sessions (each is a list of messages)
if "current_chat" not in st.session_state:
    st.session_state.current_chat = None
# --- Sidebar ---
with st.sidebar:
    st.header("💬 Chat History")
    # 🔍 Search bar
    search_query = st.text_input("Search chats")
    # ➕ New Chat button
    if st.button("➕ New Chat", use_container_width=True):
        if len(st.session_state.chats) >= 15:
            st.session_state.chats.pop(0)  # remove oldest
        st.session_state.current_chat = len(st.session_state.chats)
        st.session_state.chats.append([])
    st.markdown("---")
    # Show chat titles
    for i, chat in enumerate(st.session_state.chats):
        if not chat:
            continue
        # Title = first user message
        title = next((m["content"] for m in chat if m["role"] == "user"), "Untitled Chat")
        # Apply search filter
        if search_query and search_query.lower() not in title.lower():
            continue
        # Truncate for sidebar
        short_title = title[:25] + ("..." if len(title) > 25 else ""
```

```
        # Select chat button
        if st.button(short_title, key=f"chat_{i}", help=title, use_container_width=True):
            st.session_state.current_chat = i
# --- Main Chat Area ---
if st.session_state.current_chat is not None:
    chat = st.session_state.chats[st.session_state.current_chat]
    # Display messages
    for msg in chat:
        with st.chat_message(msg["role"]):
            st.markdown(msg["content"])
    # Chat input
    if prompt := st.chat_input("Type your message..."):
        # Save user message
        chat.append({"role": "user", "content": prompt})
        # Get response from DeepSeek (via Ollama)
        response = ollama.chat(
            model="deepseek-coder:1.3b",
            messages=chat
        )
        reply = response["message"]["content"]
        # Save assistant reply
        chat.append({"role": "assistant", "content": reply})
        # Refresh UI
        st.rerun()
else:
    st.info("👉 Start a new chat from the sidebar.")
```

2. DEEPSEEK-R1:1.5B using ollama pull command

```
import streamlit as st
import ollama
st.set_page_config(page_title="DeepSeek R1 1.5B Chatbot", page_icon="🤖", layout="wide")
st.title("🤖 DeepSeek-R1 1.5B Chatbot (Ollama)")
# ---------------------------
# Initialize session state
# ---------------------------
if "conversations" not in st.session_state:
    st.session_state.conversations = {
        "Chat 1": [
            {"role": "system", "content": "You are a helpful AI assistant."}
        ]
    }
if "active_chat" not in st.session_state:
    st.session_state.active_chat = "Chat 1"
# ---------------------------
# Sidebar - chat history
# ---------------------------
with st.sidebar:
    st.header("📁 Chats")
    # Button for new chat
    if st.button("➕ New Chat"):
        new_name = f"Chat {len(st.session_state.conversations)+1}"
        st.session_state.conversations[new_name] = [
            {"role": "system", "content": "You are a helpful AI assistant."}
        ]
        st.session_state.active_chat = new_name
        st.rerun()

    # Show list of chats
    for name in st.session_state.conversations.keys():
```

```
        if st.button(name):
            st.session_state.active_chat = name
            st.rerun()
    # Clear chat button
    if st.button("🗑️ Clear Current Chat"):
        st.session_state.conversations[st.session_state.active_chat] = [
            {"role": "system", "content": "You are a helpful AI assistant."}
        ]
        st.rerun()
# ----------------------------
# Main chat window
# ----------------------------
chat_name = st.session_state.active_chat
messages = st.session_state.conversations[chat_name]
st.subheader(f"💬 {chat_name}")
# Show past messages
for msg in messages:
    if msg["role"] == "user":
        st.chat_message("user").write(msg["content"])
    elif msg["role"] == "assistant":
        st.chat_message("assistant").write(msg["content"])
# Input box
if prompt := st.chat_input("Type your message..."):
    # Save user message
    messages.append({"role": "user", "content": prompt})
    with st.chat_message("user"):
        st.write(prompt)
    with st.spinner("Thinking..."):
        try:
          response = ollama.chat(
              model="deepseek-r1:1.5b",
              messages=messages
```

```
        )
        reply = response["message"]["content"]
        # Save assistant reply
        messages.append({"role": "assistant", "content": reply})
        with st.chat_message("assistant"):
            st.write(reply)
    except Exception as e:
        st.error(f"Error: {e}")
```

3. DEEPSEEK-R1-DISTILL-LLAMA-70B using groq API key

```
import streamlit as st
from groq import Groq
from datetime import datetime
# ------------------------------
# Page Configuration
# ------------------------------
st.set_page_config(
    page_title="DeepSeek R1 Chatbot",
    layout="wide"
)
# ------------------------------
# Session State Initialization
# ------------------------------
if "messages" not in st.session_state:
    st.session_state.messages = []
# ------------------------------
# Helper Functions
# ------------------------------
def append_message(role, content):
    """Append a message to the session state with timestamp."""
    st.session_state.messages.append({
        "role": role,
```

```
        "content": content,
        "timestamp": datetime.now().strftime("%H:%M:%S")
    })
def display_last_message():
    """Display the last user and bot messages in the sidebar."""
    if st.session_state.messages:
        user_msgs = [m for m in st.session_state.messages if m["role"] == "user"]
        bot_msgs = [m for m in st.session_state.messages if m["role"] == "assistant"]
        if user_msgs:
            last_user = user_msgs[-1]
            st.markdown(f"**You ({last_user['timestamp']}):** {last_user['content']}")
        if bot_msgs:
            last_bot = bot_msgs[-1]
            st.markdown(f"**Bot ({last_bot['timestamp']}):** {last_bot['content']}")
    else:
        st.info("No messages yet.")
# ------------------------------
# Sidebar
# ------------------------------
with st.sidebar:
    st.title("Settings & Last Message")
    # API Key Input
    api_key = st.text_input("Groq API Key", type="password")
    st.markdown("[Get Groq API Key](https://console.groq.com/keys)")
    # Clear Chat
    if st.button("Clear Chat"):
        st.session_state.messages = []
        st.experimental_rerun()
    st.markdown("---")
    st.subheader("Last Messages")
    display_last_message()
```

```
# ------------------------------
# Main Chat Panel
# ------------------------------
st.title("DeepSeek R1 Chatbot")
st.caption("Powered by Groq API")
# Chat Input
prompt = st.chat_input("Type your message here...")
if prompt:
    if not api_key:
        st.info("Please enter your Groq API key to continue")
        st.stop()
    # Append and display user message
    append_message("user", prompt)
    with st.chat_message("user"):
        st.markdown(prompt)
        st.caption(f"_{st.session_state.messages[-1]['timestamp']}_")
    # Call Groq API
    try:
        client = Groq(api_key=api_key)
        response = client.chat.completions.create(
            model="deepseek-r1-distill-llama-70b",
            messages=[{"role": m["role"], "content": m["content"]} for m in st.session_state.messages],
            # Optional parameters for customizing responses:
            # temperature=0.7, max_tokens=1024, top_p=1.0, frequency_penalty=0.0,
            stream=False
        )
        ai_response = response.choices[0].message.content
        # Append and display bot message
        append_message("assistant", ai_response)
        with st.chat_message("assistant"):
            st.markdown(ai_response)
            st.caption(f"_{st.session_state.messages[-1]['timestamp']}_")
```

```
except Exception as e:
    st.error(f"Error generating response: {str(e)}")
```