Simple LangGraph App with Streamlit UI

A basic LangGraph application with tools, RAG, and memory - now with a beautiful Streamlit web interface!

Files

- `streamlit_app.py` - Main Streamlit web interface
- `app.py` - Original CLI version
- `tools.py` - Simple custom tools (calculator, time, search)
- `rag.py` - Basic RAG with FAISS vector store
- `config.py` - Simple configuration
- `run_streamlit.py` - Easy launcher for the Streamlit app
- `requirements.txt` - Dependencies

Setup

1. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

2. Create `.env` file:

   ```
   OPENAI_API_KEY=your_openai_key_here
   TAVILY_API_KEY=your_tavily_key_here
   ```

   Get API Keys:

   - OpenAI: https://platform.openai.com/api-keys
   - Tavily: https://app.tavily.com/sign-in (free tier available)

3. Run the Streamlit app:

   ```
   streamlit run streamlit_app.py
   ```

   Or use the launcher:

   ```
   python run_streamlit.py
   ```

Features

🌟 Streamlit Web Interface

- **Chat Interface**: Clean, modern chat UI
- **Example Queries**: Click buttons to try sample questions
- **Session Management**: Clear chat history, unique session IDs
- **Sidebar Info**: Feature explanations and technical details
- **Real-time**: Instant responses with loading indicators

🛠️ Core Functionality

- **Tools**: Calculator, current time, **real web search** (via Tavily)
- **RAG**: Retrieves docs when you ask "what is", "explain", etc.
- **Memory**: Remembers conversation history across the session
- **Simple**: Minimal classes, functional approach

Usage Examples

In the Streamlit UI:

- Click example buttons in the sidebar
- Type in the chat input at the bottom
- See responses with thinking indicators
- Clear history with the sidebar button

Programmatic usage:

```python
from streamlit_app import chat_with_assistant

# Use tools
response = chat_with_assistant("Calculate 15 * 23")

# Trigger RAG
response = chat_with_assistant("What is LangGraph?")

# Normal conversation with memory
response = chat_with_assistant("My name is Alice", thread_id="user1")
response = chat_with_assistant("What's my name?", thread_id="user1")
```

Streamlit Features

🎯 Interactive Elements

- **Example Buttons**: Quick-start with pre-made queries
- **Session Management**: Clear chat, view session ID
- **Expandable Details**: Technical information panel
- **Responsive Design**: Works on desktop and mobile

🔧 Performance Optimizations

- **Caching**: LLM and graph initialization cached with `@st.cache_resource`
- **Session Persistence**: Chat history maintained in session state
- **Unique Sessions**: Each browser session gets its own memory thread

🎨 UI Components

- **Chat Messages**: Native Streamlit chat interface
- **Sidebar**: Feature info and controls
- **Status Indicators**: Loading spinners and success messages
- **Error Handling**: Graceful error display

How it Works

1. **Graph Structure**: Assistant → Tools (if needed) → Assistant → End
2. **RAG Trigger**: Keywords like "what is", "explain" trigger document retrieval
3. **Memory**: Uses MemorySaver with unique session IDs for persistence
4. **Tools**: LLM decides when to use calculator, time, or search tools
5. **Streamlit**: Provides web interface with chat history and interactive elements

The app automatically:

- Retrieves relevant documents for knowledge questions
- Uses tools when needed (math, time, search)
- Remembers conversations within the browser session
- Handles errors gracefully with user-friendly messages
- Provides example queries to get started quickly

Running Options

Streamlit Web UI (Recommended):

```
streamlit run streamlit_app.py
```

Command Line Interface:

```
python app.py
```

Easy Launcher:

```
python streamlit run week-04-langgraph-chatbot/app.py
```