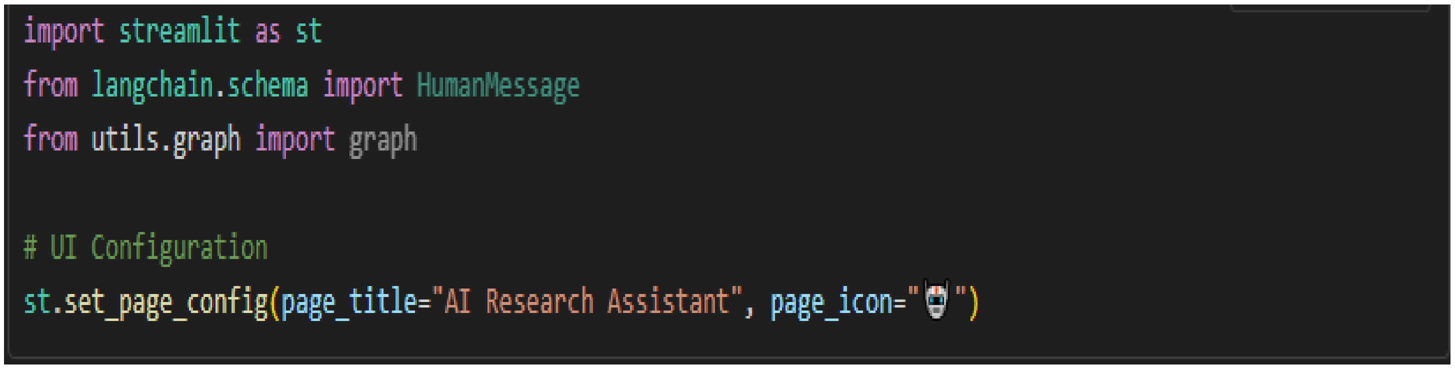
DEEP RESEARCH AI - DETAILED IMPLEMENTATION

Project Overview

This is an AI-powered research assistant that combines LangGraph for workflow management with Streamlit for the user interface. The system uses a sophisticated two-agent architecture to process queries and generate informed responses.

Core Components

1. **Main Application (main.py)**



```python
import streamlit as st
from langchain.schema import HumanMessage
from utils.graph import graph

# UI Configuration
st.set_page_config(page_title="AI Research Assistant", page_icon="🤖")
```

- Creates a Streamlit web interface
- Manages chat history using session state
- Implements real-time streaming responses
- Processes user input through the LangGraph workflow

2. LangGraph Workflow (utils/graph.py)

The workflow is defined as a directed acyclic graph (DAG) with two main agents:

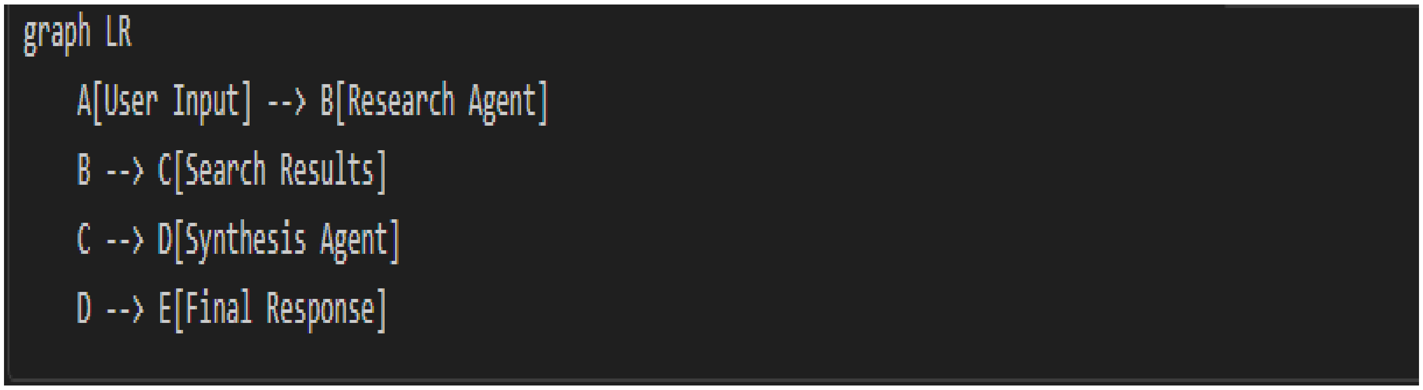
Research Agent

- Uses Tavily API for web searches
- Processes user queries to gather relevant information
- Returns structured search results

Synthesis Agent

- Powered by Google's Gemini model
- Processes search results
- Generates coherent, contextual responses

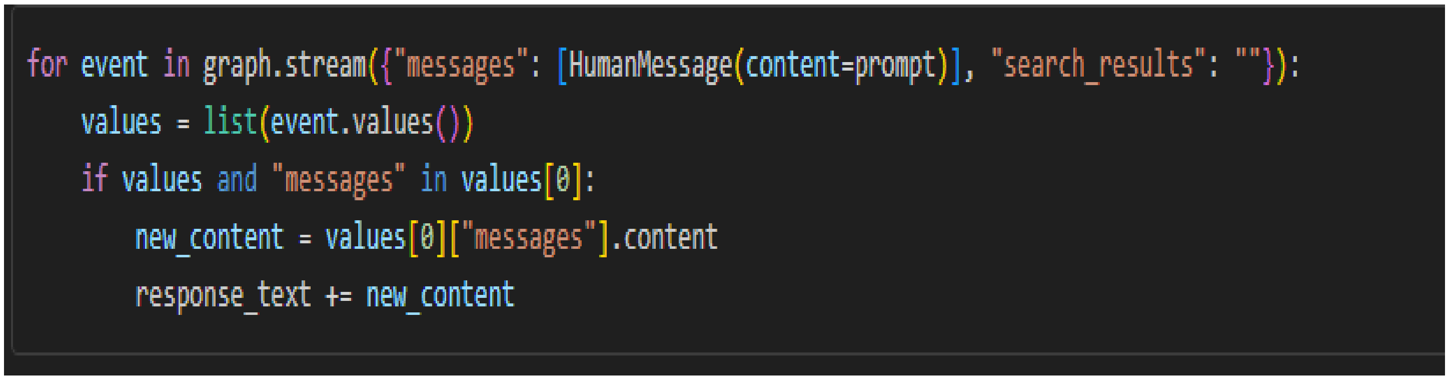
Workflow Architecture



```
graph LR
    A[User Input] --> B[Research Agent]
    B --> C[Search Results]
    C --> D[Synthesis Agent]
    D --> E[Final Response]
```

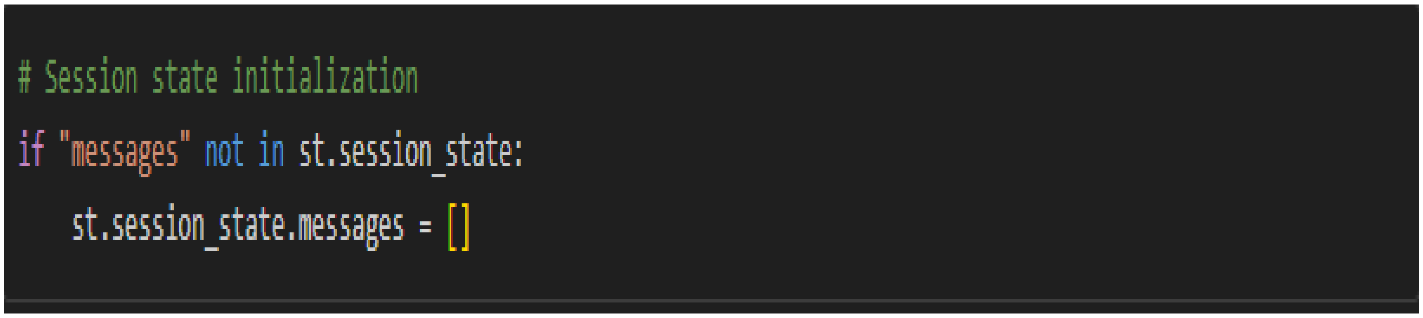
Key Features

1. **Real-time Response Streaming**



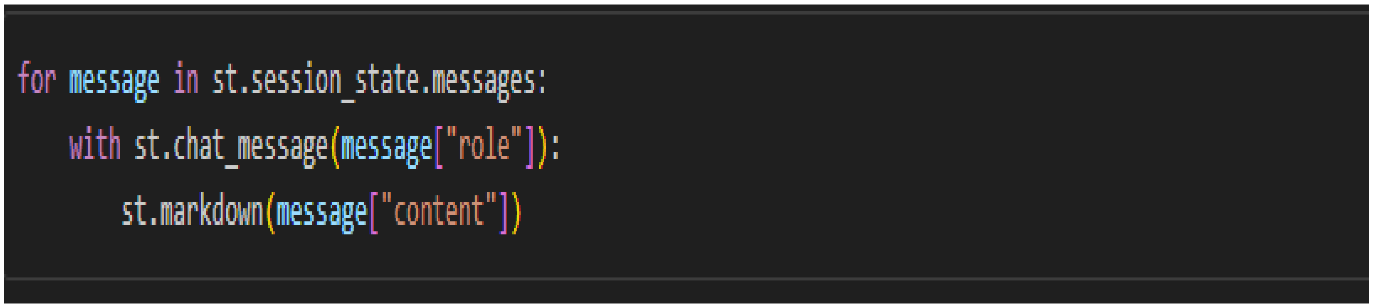
```
for event in graph.stream({"messages": [HumanMessage(content=prompt)], "search_results": ""}):
    values = list(event.values())
    if values and "messages" in values[0]:
        new_content = values[0]["messages"].content
        response_text += new_content
```

2. **State Management**



```python
# Session state initialization
if "messages" not in st.session_state:
    st.session_state.messages = []
```

3. **Chat History**



```python
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])
```

Technical Details

Dependencies

- Streamlit: Web interface
- LangGraph: Workflow management
- Tavily API: Web search functionality
- Google Gemini: Language model
- LangChain: LLM framework integration

Data Flow

1. User submits query through Streamlit interface
2. Query processed by Research Agent using Tavily API
3. Search results passed to Synthesis Agent

4. Gemini model generates response
5. Response streamed back to user interface
6. Chat history updated in session state

This implementation creates a powerful research assistant that combines web search capabilities with advanced language understanding to provide comprehensive, real-time responses to user queries.