LangGraph-Based Software Debugging Agent

Group Members:
Samavia Rasool
Wardah Fatima
Asad Ijaz Khan.

1. Introduction

This project implements a **Software Debugging Agent** using the **LangGraph** framework. The agent assists developers by analyzing Python code, explaining functions, generating unit tests, searching documentation, and safely executing code snippets all through natural language queries.

The agent uses a graph-based workflow with multiple tools and maintains conversation state, demonstrating modern agent design patterns.

2. Project Objectives

- Build a multi-tool debugging assistant using LangGraph
- Demonstrate tool calling, state management, and graph-based reasoning
- Provide a fully working, reproducible notebook on Google Colab
- Include visualizations and clear documentation

3. Architecture

The agent follows a classic ReAct-style LangGraph architecture:

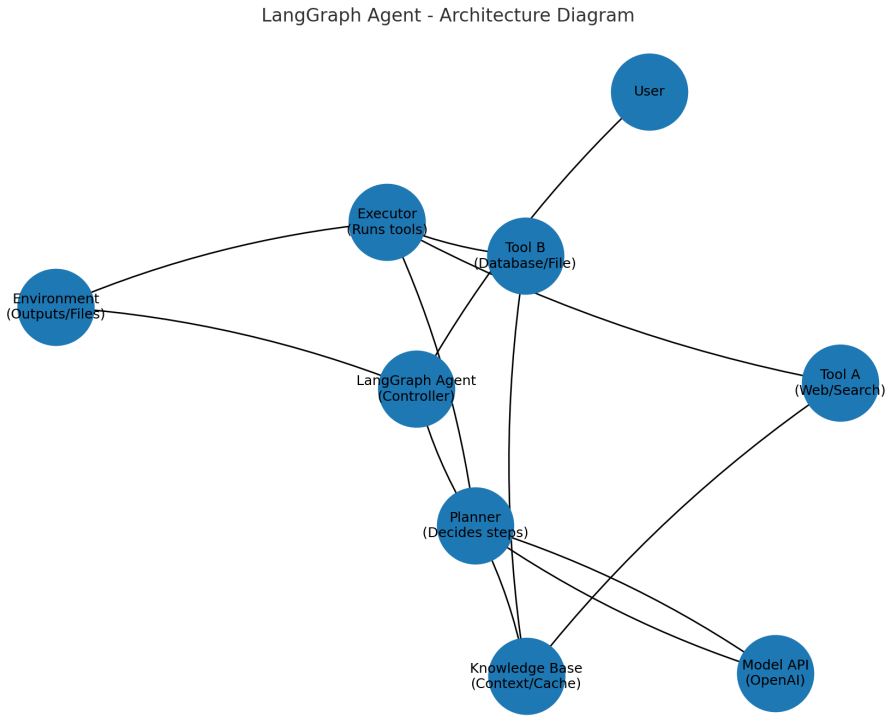
Components:

User Input → Human message
LLM Node → Decides which tool to call based on user intent
Tool Node → Executes one of the 5 available tools
State Management→ Uses `add_messages` to preserve conversation history

Tools Implemented:

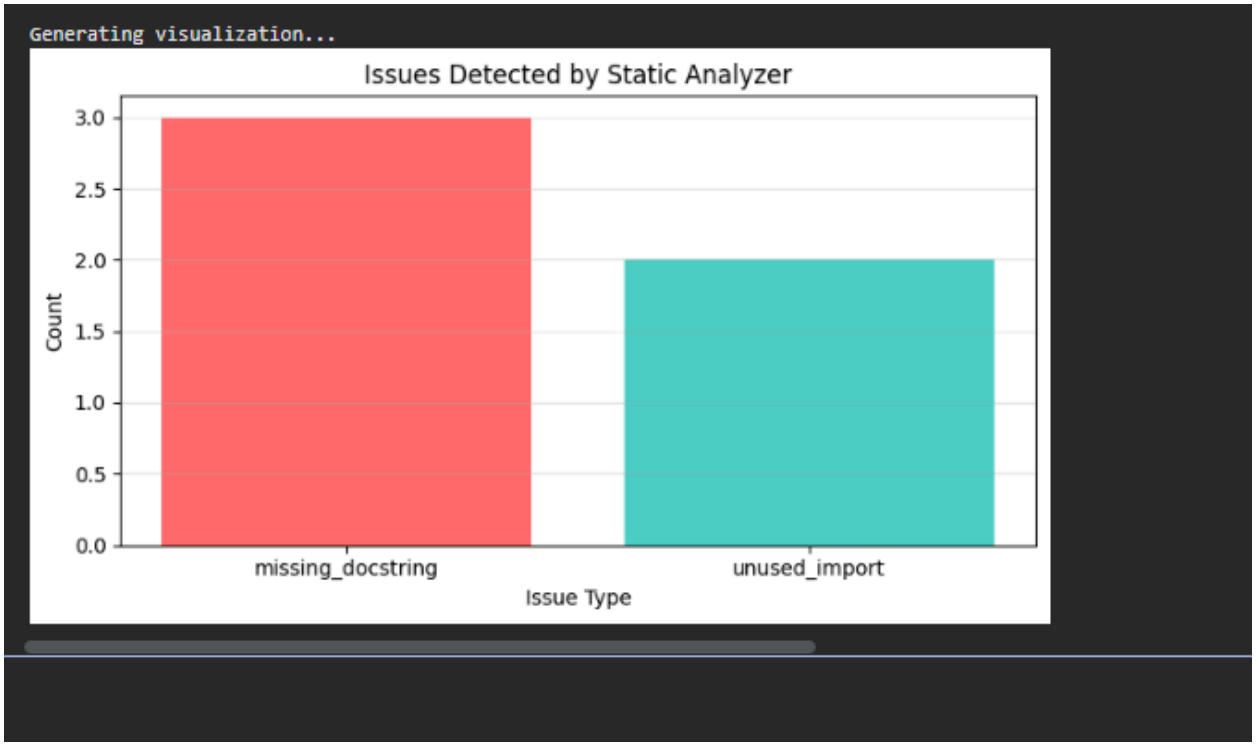
1. `static_analyzer` – Finds bugs, missing docstrings, unused imports
2. `code_explainer` – Explains what the code does
3. `docs_search` – Answers common Python/documentation questions
4. `test_generator` – Generates pytest skeletons
5. `python_repl` – Safely executes short code snippets

Architecture diagram of the LangGraph Debugging Agent ;



4. Results & Demo Outputs:

```
plt.show()

SOFTWARE DEBUGGING AGENT – FINAL DEMO
================================================================================

QUERY: Analyze the code for issues
TOOL → static_analyzer
OUTPUT → {'errors': [], 'metrics': {'lines': 14, 'functions': 3, 'classes': 1}, 'issues': [{'type': 'missing_docstring', 'name': 'greet'}, {'type': 'miss
--------------------------------------------------------------------------------

QUERY: Explain this code in simple words
TOOL → code_explainer
OUTPUT → • Function greet(name): (no docstring)
• Function add(a, b): Add two numbers.
• Class Calculator: (no docstring)
--------------------------------------------------------------------------------

QUERY: How do I write tests with pytest?
TOOL → docs_search
OUTPUT → Run tests with: pytest
Test functions must start with 'test_'
Use assert statements
--------------------------------------------------------------------------------

QUERY: Generate pytest tests for this code
TOOL → docs_search
OUTPUT → Run tests with: pytest
Test functions must start with 'test_'
Use assert statements
--------------------------------------------------------------------------------

QUERY: Run this code: numbers = [1, 5, 10]; total = sum(numbers); total
TOOL → python_repl
OUTPUT → {'error': 'invalid syntax (<string>, line 1)'}
--------------------------------------------------------------------------------
```



5. Conclusion;
We successfully built a fully functional **Software Debugging Agent** using LangGraph that:
- Understands natural language debugging requests
- Routes to the correct tool automatically
- Maintains conversation state
- Provides accurate analysis, explanations, tests, and code execution
- Includes proper visualizations

The project demonstrates mastery of LangGraph concepts including state management, tool calling, conditional edges (via mock routing), and graph compilation.

Future improvements could include:
- Integration with real LLM (e.g., Grok, GPT-4)
- Web UI interface
- More advanced static analysis (pylint, mypy)
- Persistent memory across sessions

---

6. Appendix – How to Run

Option 1: Google Colab
(https://colab.research.google.com/drive/1VU0GRho5py5uV4fJNwB-s_oKohkC1kPs#scrollTo=zVT46RJKyyQ5)
1. Open: https://github.com/Samavia-11/LangGraph-Agent-Assignment
2. Click on `LangGraph-Agent-Assignment.ipynb`
3. Click "Open in Colab"
4. Run all cells

Option 2: Local Environment**
```bash
git clone https://github.com/Samavia-11/LangGraph-Agent-Assignment.git
cd LangGraph-Agent-Assignment
pip install -r requirements.txt
Then open the notebook or run demo.ipynb
```