Agentic AI Chatbot Project Report

1. Objectives

- ➔ Develop an intelligent, agent-based AI chatbot capable of natural, human-friendly conversations.
- ➔ Leverage LangGraph for orchestrating chatbot logic and seamless tool integration workflows.
- ➔ Enable the bot to use multiple APIs (e.g., Yelo API, The Movie Database (TMDB))
- ➔ Design the architecture to be scalable, allowing easy addition of new tools/APIs.

2. Design

- ❖ LangGraph-Driven Architecture:
  - ➢ Central orchestration is handled by LangGraph, which defines and manages the flow between the chatbot, tools, and external APIs.
  - ➢ Graph-based execution allows clear and modular task management.
- ❖ Tool Integration Layer:
  - ➢ Custom nodes/tools represent API connectors (e.g., Yelo for services/business listing, TMDB for movie info).
  - ➢ Tools are dynamically pluggable into the LangGraph workflow.
- ❖ Conversational Flow:
  - ➢ Maintains multi-turn conversation context.
  - ➢ Trained prompts and structured responses ensure human-friendly, engaging interactions.
- ❖ Scalable Code Structure:
  - ➢ New tools can be integrated simply by defining them as LangGraph-compatible components and registering them without breaking the existing code flow.

3. Implementation

- ❖ Technology Stack:
  - ➢ Python as the core development language.
  - ➢ LangGraph for agent orchestration and graph-based execution logic.
  - ➢ LLM API (e.g., OpenAI/Anthropic) for conversational intelligence.
  - ➢ Yelo API for local service/business data.
  - ➢ The Movie Database API for movie search, metadata, and recommendations.
- ❖ Module Structure:

Agentic AI Chatbot Project Report

- ➢ Main Agent Node – Handles user messages, context storage, and intent recognition.
- ➢ Tool Nodes – Represent API calls (Yelo, TMDB) and can easily be extended for new APIs.
- ➢ Interface Layer – Web or CLI-based interface to handle chat interactions.
- ➢ LangGraph Workflow – Defines execution order:
  User Query -> LLM Reasoning ->Tool Selection ->API Execution -> Response Formatting.

4. Challenges

- ★ API Integration Complexity:
  Ensuring each API returns consistent, usable data while handling authentication, rate limits, and error responses.
- ★ Maintaining Natural Conversation:
  Balancing structured tool calls with an unstructured, friendly chat experience.
- ★ Extensibility Without Overhead:
  Designing tools in a modular way that fits cleanly into LangGraph's node-based model.
- ★ Latency Management:
  Minimizing delays caused by sequential API calls and LLM reasoning.

5. Lessons Learned

- ➔ LangGraph Simplifies Orchestration:
  Graph-based design makes it intuitive to manage multi-step workflows and add/remove tools.
- ➔ Pluggable Architecture Boosts Scalability:
  A clear interface for tools means new APIs can be added with minimal code changes.
- ➔ APIs Improve Chatbot Utility:
  Integrating Yelo and TMDB showed that users value actionable, up-to-date information.
- ➔ User Friendliness = Engagement:
  Even complex agent reasoning feels approachable when responses are conversational and empathetic.
- ➔ Error Tolerance is Critical:
  Providing fallback responses keeps the experience smooth if an API fails.