Banking Customer Support AI Agent using Multi-Agent Architecture

Course-end Project 2 — Multi-Agent GenAI System (LangGraph + LangChain)

1. Problem Scenario

Digital banking platforms handle high volumes of support messages across multiple channels. Traditional systems are often fragmented, leading to slow status updates, inconsistent responses, and limited personalization. An AI-driven multi-agent system can scale support operations by classifying messages, detecting sentiment, and integrating with service records to provide real-time ticket updates.

2. Objectives

This project implements a multi-agent AI assistant that:

• Classifies incoming messages into feedback (positive/negative) or queries

• Generates personalized responses based on classification and sentiment

• Tracks tickets and provides status updates via integration with a support database

3. Solution Overview

The system uses a multi-agent architecture orchestrated with LangGraph. Each agent/tool specializes in one responsibility:

• Classifier Agent — categorizes the message type

• Sentiment Agent — detects sentiment and intensity

• Ticket Agent — reads/updates/creates support tickets in a SQLite database

• Response Agent — crafts a final customer-facing message using outputs of other agents

4. Architecture (LangGraph State Machine)

Flow:

1) Classify → (2a) If ticket-related query: Ticket Agent → Sentiment → Respond

(2b) Otherwise: Sentiment → Respond

The orchestrator passes a shared state (classification, sentiment, ticket_action) between agents.

5. Ticket Database Integration

A lightweight SQLite database stores tickets with fields such as ticket_id, issue_type, status, timestamps, and notes. The Ticket Agent supports:

• Status lookup (e.g., “Ticket #1002 status?”)

• Status update (e.g., “Update ticket 1002 to resolved”)

• Ticket creation (e.g., “Create ticket for transfer failure”)

6. Personalization & Safety

Responses are personalized using sentiment: negative messages trigger empathetic language and clear next steps. The assistant avoids inventing ticket IDs or sensitive private information; it only reports what exists in the database.

7. UI & Demonstration

A Streamlit app provides:

• Assistant tab: enter a message and view classification, sentiment, ticket action, and final response

• Ticket dashboard: view recent tickets; a ‘Seed demo tickets’ button populates sample tickets for demonstration

8. Evaluation Approach

Quality can be assessed by testing representative scenarios:

• Complaint without ticket (negative feedback)

• Praise/thanks (positive feedback)

• Ticket status request

• Ticket status update

Responses should be correct, empathetic when needed, and consistent with ticket records.

9. Conclusion & Future Enhancements

This multi-agent approach improves maintainability (each agent is modular) and scales to real banking workflows. Future enhancements include:

• Authentication and role-based access

• Integration with real CRM systems (Zendesk/Freshdesk/Jira Service Management)

• More advanced intent extraction and entity resolution

• Analytics dashboards for support KPIs (SLA, resolution time, CSAT)