- **Title: Scalable Multi-Agentic Sentiment Analysis with A2A, MCP, and ML Agents**

Subtitle: A Cloud-Native Deployment for Twitter and iPhone Review Understanding
Group: Project Team - 9

Team Members:

Gangadhar Singh Shiva

Ananya Chandraker

Mohd Sharik

Date: June 2025

Business Goals and Objectives

- **Goal:** Classify user sentiment from Twitter posts and iPhone reviews at scale for both enterprises and consumer customers.

- **Need:** Organizations struggle with scattered ML pipelines and lack of intelligent routing.

- **Solution:** Develop an intelligent, modular system that classifies sentiment across multiple sources.
- Enable scalability and low-latency inference through distributed deployment.
- Empower agents to operate independently while routing intelligently via a central orchestrator.
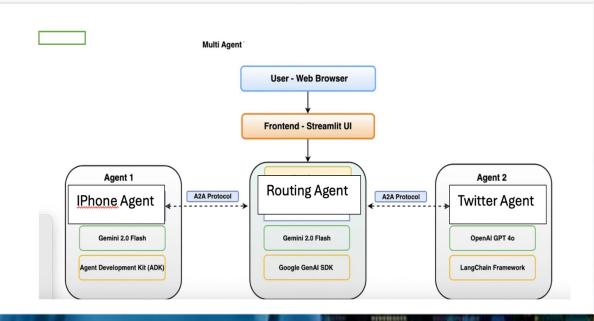- Deliver a fully deployable system on Docker/Kubernetes cloud infrastructure.

Project Dataset Summary

Twitter Dataset

- Source: Pre-labeled tweets with sentiment classification (Positive, Negative, Neutral).
- Size: ~75,000 samples
- Fields: Tweet ID, Entity, Sentiment, Tweet Content
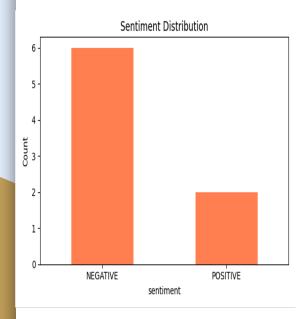
iPhone Review Dataset

- Source: Amazon product reviews
- Fields: reviewDescription, ratingScore
- Label derivation: Rating 1–2 → Negative, 3 → Neutral, 4–5 →
- Positive Neutral, Negative

System Architecture

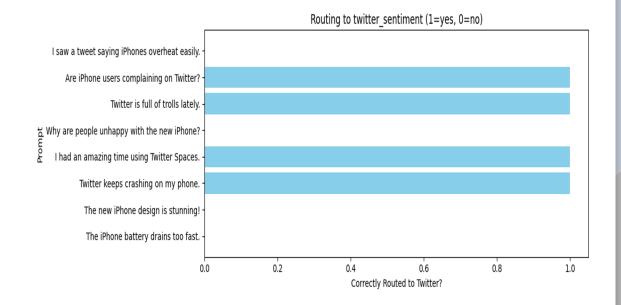| Agent	Model	Input	Output
iPhone Agent	Random Forest / TextBlob	Review text	Sentiment
Twitter Agent	BERTweet	Tweet content	Sentiment
Router Agent	Keyword/Embedding	Query	Routes to Agent
MCP Server	FastMCP	Agent requests	Manages routing/Sentiments

Performance Metrics

Sentiment Distribution

Routing to twitter_sentiment (1=yes, 0=no)

Agent	Accuracy	Notes
iPhone ML Agent	~86%	RF + TF–IDF
Twitter BERT Agent	~88%	BERTweet fine-tuned
Routing Accuracy	100%	On test queries

System Architecture (ML Model)

ML Models

Twitter Agent: Fine-tuned BERTweet Transformer (HuggingFace)

- ○ `a2a_twitter_sentiment_agent.py` handles loading the model, tokenizing input, and returning prediction
- ○ Uses Hugging Face's `AutoModelForSequenceClassification`
- ○ Optimized for short-text sentiment classification
- ○ Example Query: *"Twitter keeps crashing on my phone."*
- ○ Output: **Negative** sentiment with ~88% classification confidence

System Architecture (ML Model)

ML Models

- **iPhone Agent**: Random Forest with TF-IDF vectorization and optional TextBlob sentiment analysis
  - `a2a_iphone_sentiment_agent.py` loads review CSVs, vectorizes reviews, applies RandomForestClassifier
  - Sentiment is derived from Amazon star ratings and review content
  - Alternative lightweight path uses `TextBlob` polarity scoring
  - Example Query: *"The iPhone battery drains too fast."*
  - Output: **Negative** sentiment (probability from RF classifier or polarity < -0.1 from TextBlob)
  - **Routing Agent** Uses keywo
- `a2a_main.py` uses `mcp.call(tool_name, input)` to route user query
- Routing Accuracy observed: **100%** in test samples

- Uses keyword classification or sentence embeddings to identify target agent
- `a2a_main.py` uses `mcp.call(tool_name, input)` to route user query
- Routing Accuracy observed: **100%** in test samples

System Architecture (ML Model)

ML Models

- ○ **Routing Agent** Uses key...
- ● `a2a_main.py` uses `mcp.call(tool_name, input)` to route user query
- ● Routing Accuracy observed: **100%** in test samples

- ● Uses keyword classification or sentence embeddings to identify target agent
- ● `a2a_main.py` uses `mcp.call(tool_name, input)` to route user query
- ● Routing Accuracy observed: **100%** in test samples
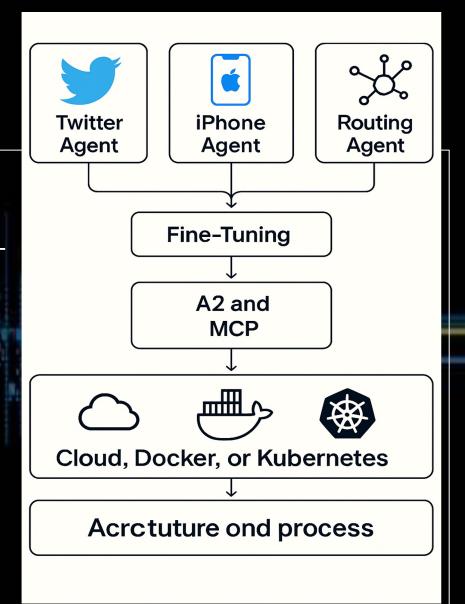
System Architecture (MCP) Workflow

MCP Client and Server Integration

The system is structured such that the MCP server listens for function calls from agent clients. MCP clients act as wrappers for the agents and route communication using either standard input/output (for terminal-based tools) or HTTP. The client uses schema definitions and registered MCP decorators (`@mcp.tool`) to facilitate execution.

- **MCP Server**: Hosts service registration, manages message routing, handles execution lifecycle.
- **MCP Client**: Sends query via CLI or embedded API (e.g., FastMCP.run("tool_name", input))
- **Embedding Support**: For routing, sentence transformers or keyword-based vector encoders are used to transform queries into semantic space and match the most appropriate agent.

Deployment Strategy

- Containerized with Docker

- Microservices on Kubernetes

- Option for FastAPI–based REST

- Deployable via Helm or

- Docker Compose

Summarize

Modular agents for domain-specific sentiment
MCP for orchestration

A2A for clean tool communication

ML, LLM for Sentimental Analysis

● Architecture can power scalable, domain-aware AI assistants — whether for enterprise analytics, customer service, or real-time monitoring.

Thank you