Empowering Learners to Build Production - Ready AI Agents

Applied Agentic AI for SWEs

Capstone Project : AI-Powered Email Assistant

Problem Statement

1. Business Use Case

Professionals spend significant time composing emails, leading to lost productivity and inconsistent tone or quality. This assistant uses agentic AI to generate, personalize, and validate email drafts in seconds.

The Business Opportunity

- **Productivity Boost**: Reduce email drafting from 15–20 mins to under 2 mins
- **Context-Aware Writing**: Tailor tone and intent to recipient and situation
- **Multi-Purpose**: Supports outreach, follow-ups, internal updates, and more
- **Scalability**: Enable consistent, on-brand communication at team/org level

---

2. Technical Architecture

The system features a multi-agent pipeline using LLMs to convert user intent and context into personalized, polished emails.

Component	Primary Tech	Alternatives	Purpose
Multi-Agent System	LangGraph / CrewAI	AutoGen, Semantic Kernel	Modular task-specific agent orchestration
Language Models	GPT-4, Claude	Gemini, Cohere	Language understanding and generation
Email Template Engine	LangChain + Function Calling	PromptLayer	Controls tone, length, and formatting
User Profile Store	JSON / MongoDB	Postgres, Weaviate	Stores preferences and prior context

Web Interface	Streamlit	Gradio, Flask	UI for composing, editing, and exporting emails
Memory Layer	LangGraph Memory	Redis, Pinecone	Remembers user tone and previous drafts
Control Plane (MCP)	LangSmith / LiteLLM	Helicone, Custom Router	Optimizes model selection and routing

---

3. Agent Architecture

Agent Name	Function
Input Parsing Agent	Validates prompt, extracts intent, recipient, tone, constraints
Intent Detection Agent	Classifies intent: outreach, follow-up, apology, info, etc.
Tone Stylist Agent	Adjusts tone (formal, friendly, assertive, etc.) using tokenized prompts
Draft Writer Agent	Generates main body text with structure and clarity
Personalization Agent	Injects user profile data and prior messages
Review & Validator Agent	Checks grammar, tone alignment, and ensures contextual coherence
Routing & Memory Agent	Manages fallback flow, logs drafts, and stores profile memory

---

4. Deliverables / Objectives

Week 1: Core Agent Workflow

- Implement Input Parsing Agent for context normalization
- Build Intent Detection Agent using classification prompts
- Set up Draft Writer Agent with tone-aware templates
- Integrate Tone Stylist for 3 tone modes (e.g., formal, casual, assertive)
- Store personalization data (company, name, style) in local JSON

Week 2: Finalization + UI + Memory

- Add Review Agent to check tone and grammar
- Implement Router Agent for fallback and retry logic using LangGraph
- Build Streamlit UI:
    - Context + tone selector
    - Email preview and editor
    - Export to email/PDF (optional)
- Log user draft edits and personalize subsequent suggestions
- Optional: Dockerize and deploy via Streamlit Cloud or local

---

5. Learning Goals

Agentic AI Skills

- LangGraph / CrewAI: Build cooperative multi-agent workflows
- Prompt Engineering: Create tone-flexible, context-rich prompts
- Function Calling: Use structured outputs for consistent draft formatting

- Personalization + Memory: Persist preferences and reuse tone context
- MCP Routing: Dynamically route across models (fallback or optimization)

System Design Skills

- Scalable agent modules (intent → tone → draft → review)
- Reliable fallback/validation layers for output safety
- Customizable UI components for human-in-the-loop feedback

- Modular design for easy tone or template extension

---

6. Submission Guidelines

Working Prototype

- Accepts a user prompt, tone, and optional metadata
- Returns a complete, editable email draft
- UI with:
    - Tone & intent dropdowns
    - Real-time preview
    - Editable textbox with copy/export option

Demo Video

- Show user prompt → email generation pipeline
- Tone change & validation demo
- Optional: fallback recovery or MCP routing visualization

Codebase Structure

email_assistant/
├── src/

```
│   ├── agents/
│   │   ├── input_parser_agent.py
│   │   ├── intent_detection_agent.py
│   │   ├── tone_stylist_agent.py
│   │   ├── draft_writer_agent.py
│   │   ├── personalization_agent.py
│   │   ├── review_agent.py
│   │   └── router_agent.py
│   ├── workflow/
│   │   └── langgraph_flow.py
│   ├── ui/
│   │   └── streamlit_app.py
│   ├── memory/
│   │   └── user_profiles.json
│   ├── integrations/
│   │   ├── openai_client.py
│   │   └── cohere_client.py
├── data/
│   └── tone_samples/
├── config/
│   └── mcp.yaml
├── Dockerfile
└── README.md
```

---

7. Evaluation Criteria

Category	Weight	Details
Functionality	30%	Drafts are accurate, relevant, tone-aligned
Agentic Architecture	25%	Distinct modular agents, LangGraph routing, fallback handling
User Experience	20%	Intuitive UI, live preview, editable content
Routing & MCP	10%	Shows fallback, logs usage or model switching
Innovation	10%	Custom tones, template libraries, personalization memory
Documentation	10%	Readme, agent flows, prompt logic, deployment