Towards “Personalized” AI Assistant

Mia Mohammad Imran

So far we talked about

- Prompt Engineering
- How to get better results out of prompting
- CLI agent architecture
- AI agent Patterns
  - Reflection pattern
  - Tool use pattern
  - Planning pattern
  - Multi-agent pattern

So far we talked about

- Prompt Engineering
- How to get better results out of prompting
- CLI agent architecture
- AI agent Patterns
  - Reflection pattern
  - Tool use pattern
  - Planning pattern
  - Multi-agent pattern

Question: *How do we integrate everything into a real system that you could responsibly deploy? Aka how do we do AI system design? And Not just a simple chatbot.*

What is a “Personalized” AI Assistant?

It’s Not just a chatbot

The assistant:

- Operates on your local files
- Uses your project artifacts
- Follows your defined constraints
- Executes only allowed capabilities

Personalization = scoped execution over your environment

How do we build a secure, local, personalized AI assistant that we can actually trust and deploy?

Openclaw?

https://github.com/openclaw/openclaw

Goal: Build an assistant that can:

- Read your workspace
- Execute controlled actions
- Automate workflows

Openclaw as a reference

Concept:

An assistant that can:

- Read files
- Perform actions
- Automate workflows

But implemented safely

Design Principle

- **Security must be architectural**
- Do NOT rely on prompting: *“Please don’t execute dangerous commands”*
- Instead: Design layers that prevent it

Building OpenClaw-like assistant from scratch without the security issues?

https://composio.dev/blog/building-openclaw-from-scratch

A secure local assistant has 3 layers:

- LLM (brain)
- Interface
  - We will do Text-only interface (CLI or local web UI): [For now]
  - You can do
    - Speech features
    - Video features
- Controller layer (tools + automation)

Choose a Local LLM

DONT USE OLLAMA CLOUD VERSION

- Good open models you can run offline:
  - **Llama 3.2:3b (very small)**
  - **Mistral/Ministral/Qwen smaller models (<= 7b)**
  - **GPT-OSS:20b (if you have good enough)**
- For easy local inference:
  - **Ollama**
  - llama.cpp
  - Local vLLM

- **DO NOT GIVE Network privileges**

Interface Layer: Two Options

- **Option A: CLI (Most Secure)**
  - Direct terminal interaction
  - Minimal surface area
- **Option B: Local Web UI**
  - Runs only on localhost
  - No external binding

Controller Layer (Tools + Automation)

This is the only layer that can:

- Access filesystem
- Run subprocess
- Access network

And only under strict rules

Only layer allowed to access the network

- Examples:
  - Email (SMTP/IMAP)
  - Discord bot
  - External REST APIs

DON’T Do:

LLM → executes raw HTTP call

LLM → runs arbitrary API request

LLM → constructs dynamic shell

Summary of “Secure” Locally build Personal AI agent

Layer	Internet	File Access	Execution
LLM	None	None	None
Interface	None	None	None
Controller	Yes (restricted)	Yes (isolated)	Yes (whitelisted only)

Demo

https://github.com/imranraad07/CS-5001-AI-Augmented-SE

Let’s build “LocalClaw”

LocalClaw = a working local assistant skeleton to help with Emails:

- Local LLM inference via **Ollama**
- Email tool integration
- Interface:
  a. Terminal only
  b. A local dashboard with REST API (Flask)

Your Task

Evolve This Into “Secure Local Assistant”

- **Reflection pattern**: add a critique pass before sending
- **Planning pattern**: multi-step workflows (triage → draft → verify → send)
- **RAG**: ground replies in local artifacts (policies, templates, contacts)