



id: SI_002

model: openai/gpt-5.4-mini

seeds:

04_risiko: usikkerhet

03_relasjon: tillit

05_handling: utforsk

01_perspektiv: operasjonell

02_tidhorisont: historisk

Instans SI_002

Seeds: operasjonell | historisk | tillit | usikkerhet | utforsk

For en **200-personers, strengt regulert helseorganisasjon** ville jeg ikke startet med “100-agents swarm” som et generisk rammeverk, men som en **kontrollert orkestreringsflate** over Microsoft-stacken. Historisk sett vinner organisasjoner som deres på **gradvis utvidelse av eksisterende kontrollplan**, ikke på å importere en ny AI-plattform.

1) Arkitektur-valg

Anbefalt vei: Azure OpenAI + egen lettvekts-orchestrator i Python (Azure Functions / Container Apps, evt. Durable Functions), med **Entra ID, Key Vault, Private Endpoints, Azure AI Search** / Dataverse connectors.

Hvorfor: Best kombinasjon av Microsoft-investment, data residency, auditability og fleksibilitet for ekte multi-agent skalering.

- **Copilot Studio + Power Automate:** bra for workflow-automatisering og lav kode, men blir raskt trangt når du vil ha dynamisk rolleallokering, 100 parallelle agents, eller avansert state management.
- **Proprietære swarm-rammeverk:** ofte rask prototyping, men svakere på governance, låsing til tredjepart og vanskeligere compliance i helse.
- **Azure OpenAI + Python:** mer arbeid, men gir kontroll på policy, observability og feilhåndtering.



2) Perspektivanalyse (100 agenter)

Ikke 100 kopier av samme prompt. Lag **rollefamilier**: regulatorisk, økonomi, drift, pasientsikkerhet, konkurranse, risiko, "red team". Hver agent får:

- egen **system prompt**
- egen **kontekstvindu-policy** (kun relevant dokumentpakke)
- egen **scorecard-output** (f.eks. risiko, evidensstyrke, anbefaling)

Aggregator bør ikke "stemme"; den bør **rangere, gruppere og flagge konflikt**. Bruk en siste **arbiter-agent** som kun oppsummerer høy-scorere og uenigheter.

3) Case-processing (100 små cases)

Bruk **queue-first**: Azure Service Bus / Storage Queue → worker pool → result store.

Hver agent bør helst dele **en felles baseline-template**, men få **case-type overrides**.

Variabel kompleksitet løses med **routing**: enkel case → billig modell; kompleks → sterk modell eller to-trinns behandling.

4) Data og kontekst

Hold data i Microsoft-økosystemet: **RAG mot SharePoint/Dataverse/Azure AI Search**, ikke kopier sensitiv info til eksterne API-er. Kritisk:

- Entra ID RBAC + minste privilegium
- Private networking
- DLP/retention policies
- prompt-injection-filtering
- full audit-logg per agentkall

5) Observerbarhet og kost

Mål per agent: tokens, latency, feilrate, retrieval-hit-rate, escalation-rate. Bruk OpenTelemetry + Azure Monitor. Kost-ROI kommer mest fra:

1. caching av retrieval-svar
2. batching av små jobs
3. billig modell først, dyr modell ved usikkerhet
4. hard cap på kontekststørrelse



6) Transition

Realistisk: **6-12 uker pilot, 3-6 måneder produksjonsklar** hvis governance er streng. Quick wins: batch-oppsummering av dokumentpakker, triage av innkommende saker, og parallelle risikovurderinger.

Når ikke swarm? Når oppgaven er enkel, lav-risiko, eller krever lite varians. Da er Copilot eller ett orkestrert LLM-kall ofte bedre.

Neste steg:

1. Velg én use case og definer scorecard + failure criteria.
2. Bygg en liten Azure Function-orchestrator med 5-10 roller.
3. Sett opp sikker RAG-pipeline i SharePoint/Dataverse med logging og kostmåling.