



id: SI_009

model: openai/gpt-5.4-mini

seeds:

04_risiko: sårbarhet

03_relasjon: samarbeid

05_handling: implementer

01_perspektiv: operasjonell

02_tidhorisont: langsiktig

Instans SI_009

Seeds: operasjonell | langsiktig | samarbeid | sårbarhet | implementer

For en **helsevesen-organisasjon med veldig streng kontroll** ville jeg velge en **Microsoft-first hybrid: Azure OpenAI + Python-agenter i Azure Functions/ Container Apps**, styrt av **Entra ID, Key Vault, Private Link, Purview** og gjerne **Azure AI Search** for grounding.

Copilot Studio + Power Automate er best for **enkle, menneske-initierte workflows**; det blir fort for rigid for 100-agents parallelle analyser og distribusert case-processing.

Proprietære swarm-rammeverk kan gi fart, men svekker langsiktig kontroll, dataresidency og governance. For dere er det ofte feil å optimere for "swarm-hype" fremfor operasjonell kontroll.

1) Arkitekturvalg

- **Azure OpenAI + egne orkestratorer:** best for skalerbarhet, logging, policy, cost control.
- **Copilot Studio:** bruk som front-end/innsteg, ikke motor.
- **CrewAI/Claude Swarm:** kun hvis eksperimentell sandbox uten sensitiv data.

2) Perspektivanalyse med 100 agenter

Ikke dupliser samme prompt. Bruk en **rollematrise**:

- 10 hovedroller x 10 subroller (f.eks. regulatorisk, kost, drift, klinisk risiko, end-user impact).
- Hver agent får:

1. rolle-specifikk system prompt



2. **begrenset kontekstpakke**
3. **tildelt delspørsmål**
4. **output-skjema** (JSON med claims, evidence, confidence, risk)

Aggregator bør være en **kritisk meta-agent** som:

- dedupliserer,
 - vekter etter evidens,
 - flagger konflikt,
 - produserer 1 beslutningsnote + 1 risk register.
- Ellers får du "100 meninger", ikke innsikt.

3) Micro-case inbox-modell

Bruk **queue + worker pool**:

- Inngående cases i Dataverse/SQL → Service Bus/Storage Queue
- Orchestrator allokere til ledige agenter basert på case-kompleksitet
- Samme prompt-template, men med **case-type overrides**
- Variabel kompleksitet håndteres ved **routing rules** og **progressive enrichment** (start liten, hent mer ved behov)

4) Data og sikkerhet

Ikke kopier sensitiv data til eksterne API-er. Bruk:

- **Private networking**
- **Managed Identity**
- **RAG mot interne indekser**
- **DLP/retention policies**
- **prompt logging med masking**
- **per-agent least privilege**

5) Observability og kost

Mål per agent: tokens, latency, error, retrieval-hit-rate, review-rate.

ROI-tiltak:

- caching av grunnkontekst
- batching av små cases
- små modeller til triage, store modeller kun ved eskalering
- fallback til rule-based eller human review ved lav confidence



6) Transition

Realistisk: **8-12 uker pilot, 3-6 mnd operationalisering.**

Quick wins:

- 1. case triage i queue
- 1. 10-agent analyse av ett avgrenset domene
- 1. governance dashboard for kost/feil/etterlevelse

Failure mode: Hvis beslutningene er sjeldne, høyrisiko og krever sterk menneskelig godkjenning, kan swarm bli dyrere og mer sårbar enn en godt designet enkeltagent-løsning.

Neste steg:

1. Velg ett domene og definer 10 rolleprofiler + JSON-output.
2. Sett opp en Azure-basert pilot med queue, orchestrator og audit-logg.
3. Mål kost/nytte mot dagens Copilot-workflow før skalering.