



id: SI\_005

model: openai/gpt-5.4-mini

seeds:

04\_risiko: mulighet

03\_relasjon: makt

05\_handling: implementer

01\_perspektiv: strategisk

02\_tidhorisont: kortsiktig

## Instans SI\_005

---

**Seeds:** strategisk | kortsiktig | makt | mulighet | implementer

For en **200-personers, strengt regulert helseorganisasjon** med **cloud-only** og **batch-toleranse**, er det strategisk riktige svaret: **bygg swarm som en Azure-native kontrollplan, ikke som et eksternt “agent framework”-eksperiment**. Microsoft-investment bevares best ved å bruke **Azure OpenAI + Python-orkestrering** (evt. Azure Functions/Container Apps) som kjerne, og bare bruke **Copilot Studio/Power Automate** der arbeidsflyten er enkel og human-in-the-loop. Proprietære swarm-rammeverk gir fortere demo, men dårligere **makt over governance, logging, data residency og modellvalg**.

### 1) Arkitekturvalg

- **Azure OpenAI + Python agents:** best for kontroll, parallelisme, observability, policy enforcement. Krever mer implementering, men høyest operasjonell verdi.
- **Copilot Studio + Power Automate:** raskest for M365-automatisering, men svak for 100-agent parallellanalyse og dynamisk agent-allokering.
- **CrewAI/Claude Swarm osv.:** rask prototyping, men dårligere fit for streng helse-GRC; risiko for dataflyt utenfor Microsoft-perimeter.

**Anbefaling:** Azure OpenAI + service-bus/queue + Python workers + Microsoft Purview/DLP + Entra ID.



---

## 2) Perspektivanalyse med 100 agenter

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

- 20 domeneagenter (klinisk, juridisk, drift, økonomi)
- 20 kritikk-/red team-agenter
- 20 signal-ekstraktorer
- 20 scenario/alternativ-agenter
- 20 synthesizers som bare leser andre agents output

Hver agent får: {role, objective, constraints, unique\_context\_slice, scoring\_rubric}

Output må være **strukturert JSON** med felter som: claims, evidence, risks, confidence, recommended\_actions .

Samle via en **meta-agent** som rangordner, dedupliserer og lager consensus + dissent. Kaos unngås ved at hver agent vurderes på **signalverdi**, ikke “mening”.

---

## 3) Case-processing inbox

Bruk **queue-driven worker pool**:

- Innboks i Dataverse/SQL
- Queue per case-type
- Workers henter ledig case, kjører passende prompt-template
- Dynamisk “complexity router” sender enkle saker til billig modell, tunge til sterkere modell

System-prompt kan være delt template; rolleparametre varierer per case. Konsistens sikres med **schema validation**, retry-policy og confidence thresholds.

---

## 4) Data og kontekst-injection

Bruk **RAG internt**: SharePoint/Dataverse/e-post indekseres i Microsoft-økosystemet.

Ikke kopier rådata til eksterne API-er. Kritisk:

- Entra ID auth
- Managed Identity
- Key Vault
- Purview, DLP, sensitivity labels
- Private networking / Azure OpenAI med enterprise-godkjenning



---

## 5) Observerbarhet og kost

Logg per agent: `token_in/out`, `latency`, `model`, `retry_count`, `confidence`, `cost_estimate`.

Bruk batching, caching av retrieval, og fallback til billig modell for klassifisering/sammendrag. ROI kommer ofte fra **routing**, ikke fra “bedre prompt”.

---

## 6) Transition

Realistisk: **6-10 uker** til første produksjons-POC, **3-6 måneder** til robust drift.

### Quick wins:

1. 10-agent pilot for én komplekst case-type
2. Queue-basert micro-case pipeline i Dataverse
3. Sentral observability dashboard for kost/latency/feil

**Når ikke swarm?** Når problemet er entydig, lav risiko og godt løst av én modell + enkel workflow. Swarm gir makt først når beslutningsrommet er stort nok til å fortjene det.