



id: SI_004

model: openai/gpt-5.4-mini

seeds:

04_risiko: resiliens

03_relasjon: makt

05_handling: implementer

01_perspektiv: operasjonell

02_tidhorisont: historisk

Instans SI_004

Seeds: operasjonell | historisk | makt | resiliens | implementer

Kort svar: I en **Microsoft-sentrisk, strengt regulert helseorg** er den mest robuste veien vanligvis **Azure OpenAI + egen lettvekts-orkestrering i cloud-only Azure/Functions/Logic Apps**, ikke Copilot Studio alene og ikke et eksternt swarm-rammeverk som primærplattform. Historisk sett har M365-investeringer gitt mest verdi når de **beholdes som system of record**, mens “makt” til å orkestrere flyttes kontrollert til Azure-laget.

1) Arkitektur-valg

- **Copilot Studio + Power Automate:** raskest for enkel automasjon, men svak for 100-agent parallellisme, finmasket observability og kompleks feilkontroll.
- **Azure OpenAI + Python/Functions:** best balanse for kontroll, sikkerhet, audit, batch-kjøring og skalering. Bruk Entra ID, Managed Identity, Private Endpoints, Purview.
- **CrewAI/Anthropic Swarm osv.:** ok for prototyping, men gir mer vendor-friksjon og svakere Microsoft-governance. Ikke primærvalg i deres setting.

Anbefaling: Microsoft-investment preservasjon + swarm-skalering = **Azure OpenAI som model backplane + Azure Functions/Logic Apps/Durable Functions som orchestrator + Dataverse/Blob/SQL som state.**



2) Case 1: 100-agent perspektivanalyse

Ikke dupliser samme prompt. Gi hver agent en **rolle + kontekstvinkel + vurderingsrubrikk**:

- Regulatory reviewer
- Operational cost analyzer
- Patient-safety risk analyst
- Stakeholder/power map analyst
- Counterfactual / red-team analyst

Hver agent får:

1. Felles problembrief
2. Rolle-spesifikk instruks
3. Begrenset kontekstpakke (RAG-uttrekk)
4. Standardisert output-schema (JSON: risk, evidence, confidence, recommendation)

Aggregator gjør **clustering + dedup + weighted scoring**. Ikke "100 meninger"; lag 8–12 perspektivklynger og en final synthesis-agent.

3) Case 2: micro-case inbox

Bruk **queue-first**:

- Inngående cases → Service Bus/Queue
- Worker-funksjoner henter case, scorer kompleksitet, velger agentklasse
- Standard template, men med **policy-driven prompt injection** per case-type
- Konsistens sikres via **schema validation** og fallback til "review queue" ved lav confidence

4) Data og sikkerhet

Ikke kopier sensitiv data til eksterne APler. Bruk:

- Private networking
- RAG mot SharePoint/Dataverse via indekser i Azure AI Search
- Entra ID-baserte tilgangsgrenser
- DLP, Key Vault, logging av prompt/response metadata uten PHI i klartekst



5) Observability og kost

Mål per agent: tokens, latency, retries, output quality, human override-rate. Bruk Azure Monitor/App Insights. Kost-ROI kommer mest fra:

- **batching**
- **cache av retrieval**
- **små modeller for triage, større for final synthesis**
- cutoff-regler ved lav verdi

6) Phasing

0-6 uker: 1 case-type, 5-10 agenter, batch.

6-12 uker: 20-30 agenter, observability, policy gates.

3-6 mnd: 100-agent perspektivanalyse + queue-basert micro-case pipeline.

Ikke rett løsning når:

- problemet kan løses med enkel RAG + ett svar
- krav til determinisme er absolutt
- datasettet er for lite til at parallellisme gir verdi

Neste steg:

1. Velg én pilot for hvert use case og definer JSON-output-schema.
2. Sett opp Azure OpenAI + Service Bus + Durable Functions med Entra ID/Private Link.
3. Design 8-12 faste agentroller og en synthesis-aggregator før du skalerer.