



id: SI_006

model: google/gemma-4-26b-a4b-it

seeds:

04_risiko: mulighet

03_relasjon: makt

05_handling: vent

01_perspektiv: strategisk

02_tidhorisont: historisk

Instans SI_006

Seeds: strategisk | historisk | makt | mulighet | vent

Analysen skifter fra brukersentrert assistanse til systemisk kontroll. I helsesektoren er makt ikke bare beslutninger, men kontroll over dataintegritet og regulatorisk etterlevelse.

1. Arkitektur: Det strategiske valget

Å bruke Copilot Studio for 100 agenter er en **historisk blindvei**; det er bygget for lineær logikk, ikke sverm-entropi. For å bevare Microsoft-investering uten å kveles av SaaS-begrensninger, er den eneste veien **Azure OpenAI Service (AOAI) + Python-basert orkestrering (f.eks. LangGraph)** deployet på Azure Container Apps.

- **Trade-off:** Du ofrer “no-code” enkelhet for total kontroll over state management og concurrency.
- **Makt:** Ved å eie orkestreringen i Python, eier du logikken bak beslutningshierarkiet, ikke Microsoft.

2. Spesialisering og Aggregering (Case 1)

For å unngå “støy”, må du slutte å tenke “prompts” og begynne å tenke **“Ontologiske roller”**.

- **Implementering:** Hver agent får en unik `SystemInstruction` injisert via en metadata-tabell i Dataverse.
- **Syntese:** Bruk en **Hierarkisk Konsensus-modell**. 100 agenter produserer rå-output → 5 “Synthesizer-agenter” (høyere modell, f.eks. GPT-4o) aggregerer → 1 “Final Auditor” (regulatorisk kontroll) leverer sluttrapport.



3. Distribusjon og Kontekst (Case 2 & Data)

Bruk en **Message Queue (Azure Service Bus)**.

- **Inbox-modell:** Case lander i Queue → en Dispatcher-agent analyserer kompleksitet → allokere til agent-pool.
- **Data-injeksjon:** Bruk **RAG via Azure AI Search** med Managed Identities (Entra ID). Data forblir i din Azure-tenant; agentene “ser” kun konteksten via sikkerhetstunneler. Kopier aldri sensitive helsedata til prompt-kontekst; send kun relevante embeddings.

4. Operasjonell Realitet (Kost & Risiko)

Vent. Ikke rull ut 100 agenter samtidig. Start med en “Shadow Swarm” som kjører parallelt med manuelle prosesser for å kalibrere drift.

- **Cost:** Implementer Token Budgeting per agent-rolle.
- **Failure mode:** Swarm-arkitektur feiler når agentene går inn i “feedback loops”. Du trenger en Circuit Breaker-logikk som stopper eksekvering hvis redundans i svar overstiger 30%.

Neste steg:

1. **POC:** Sett opp en Azure Container App med LangGraph som håndterer 5 spesialiserte agenter mot en test-database.
2. **Governance:** Definer Entra ID-roller for “Agent-identiteter” (Service Principals).
3. **Benchmark:** Mål differansen i beslutningskvalitet mellom én Copilot-prompt og 5 spesialiserte agenter.