



id: SI_002

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

seeds:

04_risiko: sårbarhet

03_relasjon: makt

05_handling: forhandle

01_perspektiv: strategisk

02_tidhorisont: langsiktig

Instans SI_002

Seeds: strategisk | langsiktig | makt | sårbarhet | forhandle

Arkitektonisk maktbalanse: Azure OpenAI vs. Low-code

For en helseorganisasjon er **strategisk** suverenitet knyttet til data-residency. Å velge Copilot Studio/Power Automate er en kortsiktig gevinst i hastighet, men en **sårbarhet** i skalering. For 100 agenter vil Power Automate kollapse under orkestreringskostnader og latency.

Anbefalt arkitektur: En "Hybrid Sovereign Swarm". Bruk **Azure OpenAI Service** (med Managed Identities/Entra ID) orkestrert via **Python (Semantic Kernel eller LangGraph)** deployet på **Azure Container Apps**. Dette bevarer Microsoft-investeringen (data blir i din tenant), men gir deg den tekniske **makten** til å styre tokens og logikk utenfor SaaS-begrensningene.

Operasjonalisering av casene

- Case 1 (Perspektivanalyse):** Unngå duplisering via en Hierarchical Orchestration. Ikke send 100 prompts; send 10 "Lead Agents" som hver styrer 10 "Specialist Agents". Bruk en **Synthesizer Agent** (Reasoning model som o1/GPT-4o) for å vekte motstridende input. Dette transformerer kaos til et beslutningsgrunnlag du kan **forhandle** ut fra.
- Case 2 (Micro-processing):** Implementer en **Asynchronous Message Queue (Azure Service Bus)**. Hver case er en melding. Agenter er "consumers". Bruk en Factory Pattern i Python: En `AgentFactory` injiserer en spesifikk `SystemPrompt` basert på case-metadata fra Dataverse.



Sikkerhet og kontroll (Sårbarhetsanalyse)

- **Data Injection:** Bruk **Azure AI Search** (RAG) med Security Trimming. Agenten får aldri rådata, kun søkeresultater som er filtrert av brukerens Entra ID-rettigheter.
- **Kostnad/Observabilitet:** Implementer en **Token Proxy**. Alle agenter må rute forespørsler gjennom en sentral Python-wrapper som logger `user_id`, `agent_role`, og `token_count` til **Azure Monitor/Log Analytics**.

Failure Modes

Swarm-arkitektur er ikke rett for lineære oppgaver. Ved lav kompleksitet vil overhead (latency/kostnad) utradere ROI.

Neste steg:

1. **PoC (Veke 1-4):** Bygg en Python-basert "Skeleton Swarm" på Azure Container Apps som prosesserer 5 cases via Service Bus.
2. **Governance-audit:** Verifiser at Azure AI Search respekterer eksisterende SharePoint-tilgangsnivåer (ACLs).
3. **Evaluerings:** Definer "Decision Quality Score" for Case 1 før du skalerer fra 5 til 100 agenter.