



id: SI\_007

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

seeds:

04\_risiko: mulighet

03\_relasjon: tillit

05\_handling: implementer

01\_perspektiv: operasjonell

02\_tidhorisont: kortsiktig

## Instans SI\_007

**Seeds:** operasjonell | kortsiktig | tillit | mulighet | implementer

### 1. Arkitektur: Hybrid Azure-Native Orchestration

For å bevare **tillit** i helsesektoren er Copilot Studio/Power Automate for rigid for 100 agenter, mens rene rammeverk som CrewAI skaper friksjon mot din governance. **Valg:** Azure OpenAI Service + Azure Functions (Python) orkestrert via en Message Queue (Azure Service Bus).

- **Pros:** Full kontroll over data residency og Entra ID-autentisering.
- **Cons:** Høyere utviklingskostnad enn low-code.
- **Implementer:** Bruk Semantic Kernel (Microsofts SDK) for å bygge agenter som kan kalle dine eksisterende API-er sikkert.

### 2. Case 1: Spesialisering uten kaos

For å unngå duplisering må du gå fra “Prompt Engineering” til “Persona Architecting”.

- **Metode:** Bruk en Hierarchical Swarm. En “Master Orchestrator” mottar problemet, dekomponerer det i 100 sub-oppgaver, og tildeler hver oppgave en unik System Message generert dynamisk basert på en taksonomi (f.eks. Medisinsk Etikk, Logistikk, Juridisk Compliance).
- **Aggregering:** Bruk en “Synthesis Agent” som ikke oppsummerer, men utfører en Cross-Examination av de 100 svarene for å identifisere konsensus og outlier-risiko.



### 3. Case 2: Distribuert Inbox-modell

- **Arkitektur:** Database → Azure Service Bus → Azure Functions (Agent Workers) .
- **Consistency:** Bruk en Base Template (System Prompt) som er uforanderlig, men injiser Context Snippets via RAG. Ved variabel kompleksitet: Implementer en “Complexity Classifier” agent som først kategoriserer caset og velger mellom GPT-4o (kompleks) eller GPT-4o-mini (standard) for å sikre **operasjonell** kostnadskontroll.

### 4. Data, Sikkerhet og Monitorering

- **Data:** Bruk Azure AI Search med OneLake integrasjon. Data forblir i din tenant; agenter får kun kontekst via RAG-vektorer.
- **Observability:** Implementer Azure Monitor/Application Insights. Du må tracke `Token_Cost_Per_Case` og `Agent_Failure_Rate` i sanntid.
- **Failure Mode:** Hvis 100 agenter i Case 1 ender i en “hallusinasjons-loop”, må du ha en Circuit Breaker som terminerer kjøringen ved overskridelse av token-budsjett eller lav semantisk diversitet.

#### Konkrete neste steg:

1. **PoC (Kortsiktig):** Sett opp en Azure Function som trigger 5 spesialiserte agenter via Service Bus for å validere latency og auth-flow.
2. **Data-audit:** Kartlegg hvilke SharePoint-områder som kan indekseres i Azure AI Search uten å bryte helsesektorens personvernregler.
3. **Prototype Synthesis:** Test “Master Orchestrator”-logikken på et historisk datasett for å se om de 100 perspektivene faktisk gir ny innsikt.