Lab 5 & Lab 6 – Agentic Underwriting Integration Challenge

Challenge Overview

In this challenge, you will **complete the backend configuration of the Agentic Underwriting solution** by integrating multiple Azure AI services and agents created in earlier labs.

Rather than following step-by-step instructions, you will **architect and implement the integration** using the provided solution repository as a reference point. This mirrors a real-world scenario where platform teams must wire together deployed services, agents, and observability components into a functioning system.

By the end of this challenge, you should be able to **run the underwriting backend service locally**, with all agent dependencies correctly configured and responding to requests.

Challenge Goal

Your goal is to **successfully run the Agentic Underwriting backend component** using:

- A **Fabric data agent** for structured FEMA claims insights
- A **Foundry knowledge agent** for underwriting and claims manual retrieval
- **Azure OpenAI** for reasoning and orchestration
- **Azure Maps** for location-based risk enrichment
- **Application Insights + OpenTelemetry** for observability

All required services must be deployed and configured using work completed from **Lab 1 through Lab 4**, plus Azure Maps.

What You Are Given

- A working **solution repository** containing:
    - Backend API and agent orchestration logic
    - Agent abstractions and service clients
- Deployed Azure resources from previous labs:
    - Microsoft Fabric workspace and Gold data
    - Fabric data agent
    - Azure AI Search–backed knowledge index

    - Foundry agents
- An example .env configuration file that defines **all required runtime dependencies**

You are expected to **map deployed resources to backend configuration** and ensure the system runs end-to-end.

Success Criteria

You have successfully completed this challenge when **all** of the following are true:

- The backend service starts successfully with no configuration errors
- The backend can:
    - Invoke the **Fabric data agent** for structured claims queries
    - Invoke the **Foundry knowledge agent** for underwriting manual retrieval
    - Use **Azure OpenAI** for reasoning and response synthesis
    - Call **Azure Maps** to enrich requests with location intelligence
- Application logs and traces appear in **Application Insights**
- The backend API responds successfully to test requests from the frontend or API client

Backend Configuration Challenge

Your primary task is to **complete and validate backend configuration** using environment variables.

Use the example below as a **reference checklist**. Every value must map to a real deployed service in your Azure environment.

```
# Allowed CORS origins (JSON array string)

CORS_ORIGINS=["http://localhost:3000"]


# Azure AI Foundry (Fabric Agent)

FOUNDRY_FABRIC_AGENT_ENDPOINT="https://<your-foundry-
resource>.services.ai.azure.com/api/projects/<your-project-id>"

FOUNDRY_FABRIC_AGENT_NAME="underwriting-fabric-agent"


# Azure AI Foundry (Knowledge Agent)

FOUNDRY_KNOWLEDGE_AGENT_ENDPOINT="https://<your-foundry-
resource>.services.ai.azure.com/api/projects/<your-project-id>"

FOUNDRY_KNOWLEDGE_AGENT_NAME="underwriting-knowledge-agent"
```

```
# Application Insights (use full connection string from Azure Portal)
APPLICATIONINSIGHTS_CONNECTION_STRING="InstrumentationKey=<your-instrumentation-key>;IngestionEndpoint=<your-ingestion-endpoint>;LiveEndpoint=<your-live-endpoint>;ApplicationId=<your-application-id>"

# OpenTelemetry service name
OTEL_SERVICE_NAME="agentic-underwriting-backend"

# Azure OpenAI
AZURE_OPENAI_ENDPOINT="https://<your-azure-openai-resource>.openai.azure.com/"
AZURE_OPENAI_DEPLOYMENT="<your-deployment-name>"

# Azure Maps
AZURE_MAPS_BASE="https://atlas.microsoft.com"
AZURE_MAPS_SCOPE="https://atlas.microsoft.com/.default"
AZURE_MAPS_CLIENT_ID="<your-azure-maps-client-id>"

# Azure AI Foundry auth scope
FOUNDRY_OPENAI_SCOPE="https://ai.azure.com/.default"

# OpenAI API version
OPENAI_API_VERSION="2025-05-15-preview"
```

What You Must Figure Out (Intentionally Not Prescribed)

- Where to obtain:
    - Foundry agent endpoints and names
    - Azure OpenAI deployment details
    - Azure Maps client ID and permissions
    - Application Insights connection string
- How authentication is handled (managed identity vs local credentials)
- How agent calls are wired into the backend request flow
- How failures and partial responses are handled

Use:

- Azure Portal

- Microsoft Fabric
- Azure AI Foundry
- The backend source code in the solution repo

Validation Checklist

Use this checklist to confirm completion:

- Backend service starts without errors
- Fabric data agent queries return results
- Knowledge agent retrieval returns grounded excerpts
- Azure Maps requests succeed
- Application Insights shows traces and dependencies
- Frontend can successfully call backend APIs

Outcome

By completing this challenge, you will have:

- Integrated **multiple AI agents** into a single underwriting backend
- Applied **agentic architecture patterns** towards a real business use case
- Built a **production-realistic configuration** spanning data, knowledge, reasoning, maps, and observability

This completes the **core agentic underwriting system** and prepares you for advanced scenarios such as multi-agent orchestration and human-in-the-loop approvals.