

Software &
Digital Platforms

AI Agent Workshop

Building AI Agents with Azure

May 05 – May 12

Contents

Introduction to Agentic AI

Day 1: May 5, 2025

01

Introduction to Agentic AI

02

Agentic System Design Patterns

03

Azure Agentic Services & Frameworks

- Azure AI Agent Service
- Sematic Kernel
- Autogen

Workshop Assignments

Day 2: May 7, 2025

04

Workshop Objectives

05

Business Scenario

06

Workshop Challenges & Team Assignments for Guided Exercises

07

Agentic Solutions

08

Demo

Workshop Results

Day 3: May 12, 2025

09

Presentation of Team Solutions

10

Wrap-up & Final Discussion



Workshop Overview

By the end of the workshop you will learn

How to build AI Agents and explore the Agent [tools](#)

What are the AI Agent Platforms and how to use them

Hands-on experience building single and multi-agent systems

Getting Started

[README](#) - <https://github.com/microsoft/OpenAIWorkshop/blob/int-agentic/README.md>

Target Audience

Day 1: Executives, Product Management, Customer Success, Developers

Day 2: Developers, Product Management, Customer Success

Day 3: Developers



Pre-requisites

Day 1: Basic understanding of AI and LLMs

Day 2 & Day 3: Access to an Azure subscription with \$50 budget, [GitHub](#) account, VSCode, basic familiarity with python

Please complete before next meeting if you haven't already:

Questionnaire

[Microsoft Forms](#)

Agents quickstart

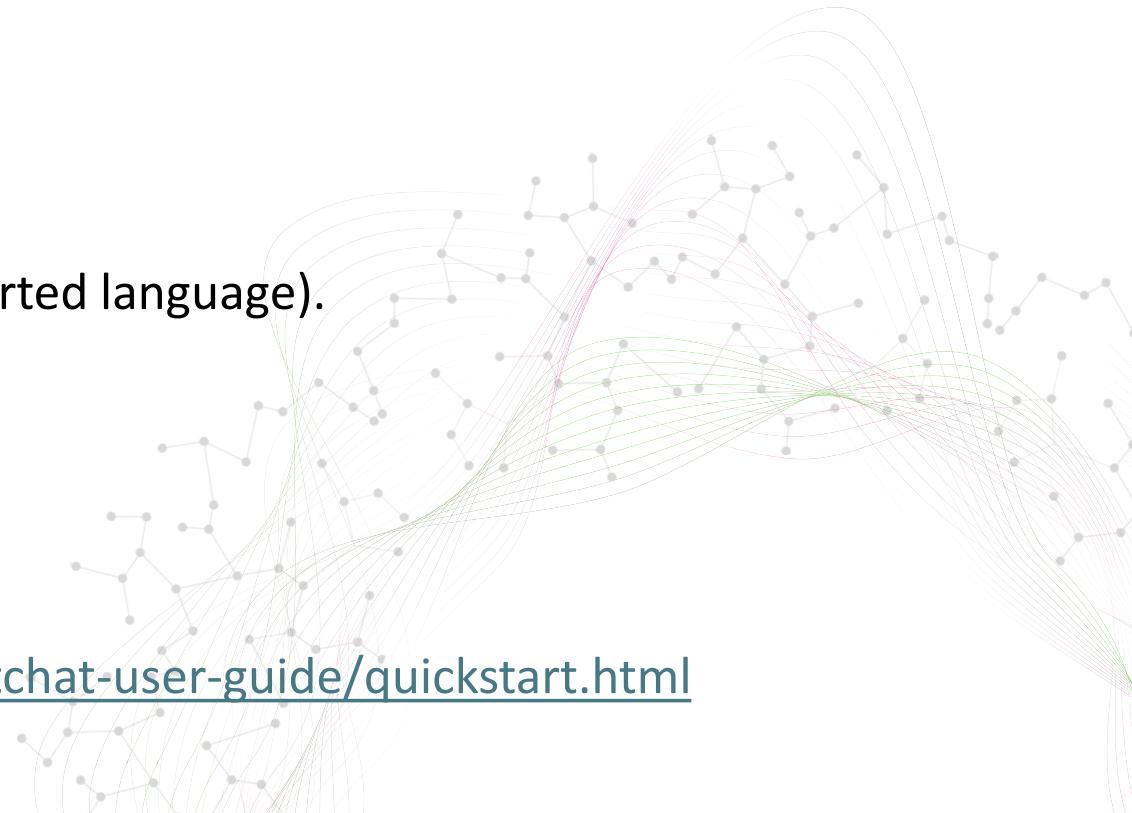
[Azure Agents Quickstart Guide](#) (Python or any other supported language).

Semantic Kernel quickstart

[Semantic Kernel Quickstart Guide](#)

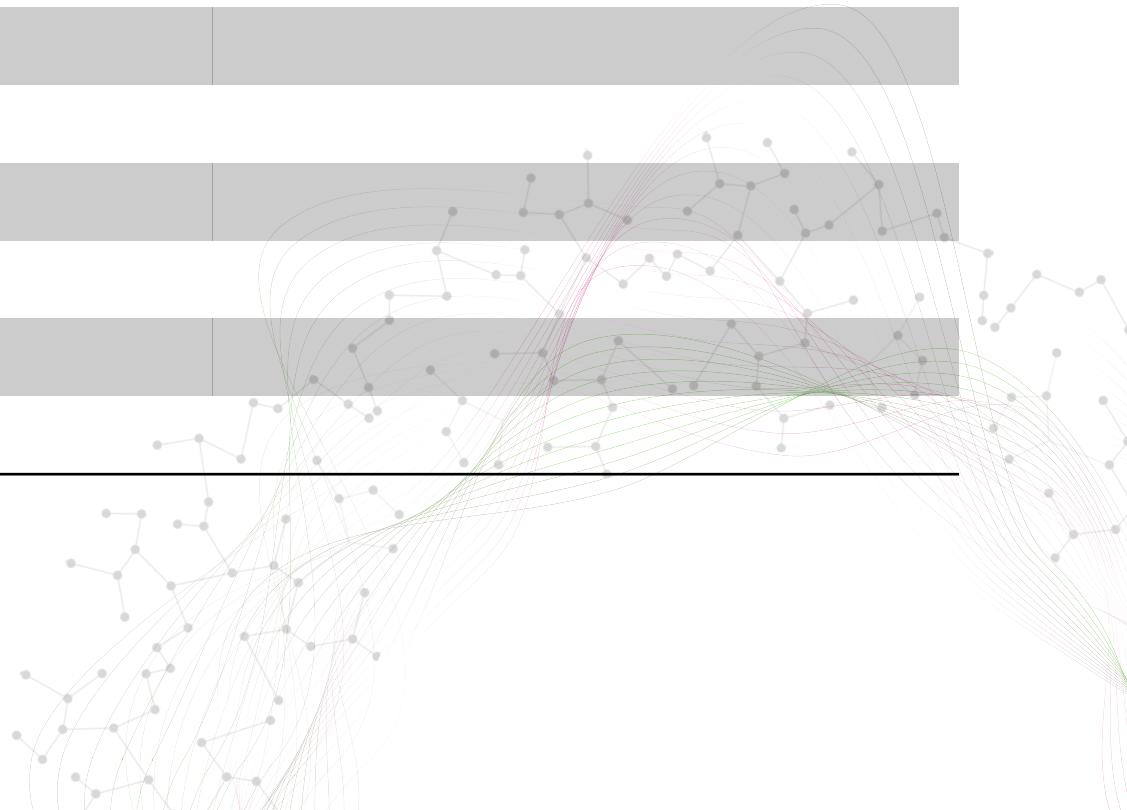
AutoGen quickstart

<https://microsoft.github.io/autogen/dev/user-guide/agentchat-user-guide/quickstart.html>



Workshop Support Team

Name	Contact	Notes
James Nguyen		
Anil Dwarkanath		
Nicole Serafino		
Patrick O'Malley		
Heena Ugale		
Aditya Agrawal		
Claire Rehfuss		
Kirby Repko		



Day 2

Workshop Assignments

Objectives

Business Scenario

Exercises

Solution Framework

Demo

Workshop Objectives

Hands-on exercises to build Agents using multiple platforms

Demonstrate using example use case / application scenario

Advance from single-agent setups to multi-agent architectures

The following cutting edge frameworks are used:

1. Azure AI Agent Service
2. Semantic Kernel
3. Autogen

Business Scenario

"Contoso Communications" provides telecom and internet services.

Customers frequently submit requests about billing, account management, service status, and promotions.

Requests range from simple queries (single data source) to complex questions requiring multiple backend systems interactions.

Source systems

- **CRM System (e.g., Salesforce or Dynamics CRM)**
 - Central repository for customer profiles, subscriptions, contract details, and interaction history.
 - Customer account data, subscription plans, contract dates, customer identification data.
 - Structured API queries (REST API).
- **Billing Database System**
 - Containing structured invoice and subscription data
 - Invoice amounts, billing history, plan details, payment status.
 - Structured SQL-style queries or REST API.
- **Product and Promotion Database**
 - Contains structured data about available products, upgrades, promotions, discounts, and eligibility criteria.
 - Promotion details, eligibility criteria, product/service details.
 - Structured API queries (REST API).
- **Security & Authentication Database**
 - Manages structured information about account security status, login attempts, and authentication issues, Account lockout reasons, authentication logs, security flags, Structured API queries or REST API.
- **Knowledge Base (Confluence or SharePoint-like system)**
 - Centralized repository for documentation, FAQs, troubleshooting guides, internal policies, and procedural guidelines.
 - Unstructured/semi-structured text documents, policy documents, troubleshooting procedures.
 - Semantic search API, keyword-based or embedding-based retrieval.

Technical Problem

By using AI Agents, "Contoso Communications" will significantly improve their customer service

Some scenarios can be sufficiently addressed using Single-agents

Other scenarios are better served using multi-agents

Workshop participants will try using single and multi-agents with different AI Agentic platforms to evaluate which scenarios and platforms provide the best customer experience

Agentic Platforms

- **AI Agent Service**
 - Central repository for customer profiles, subscriptions, contract details, and interaction history.
 - Customer account data, subscription plans, contract dates, customer identification data.
 - Structured API queries (REST API).
- **Semantic Kernel**
 - Containing structured invoice and subscription data
 - Invoice amounts, billing history, plan details, payment status.
 - Structured SQL-style queries or REST API.
- **Autogen**
 - Contains structured data about available products, upgrades, promotions, discounts, and eligibility criteria.
 - Promotion details, eligibility criteria, product/service details.
 - Structured API queries (REST API).
- **Security & Authentication Database**
 - Manages structured information about account security status, login attempts, and authentication issues, Account lockout reasons, authentication logs, security flags, Structured API queries or REST API.
- **Knowledge Base (Confluence or SharePoint-like system)**
 - Centralized repository for documentation, FAQs, troubleshooting guides, internal policies, and procedural guidelines.
 - Unstructured/semi-structured text documents, policy documents, troubleshooting procedures.
 - Semantic search API, keyword-based or embedding-based retrieval.

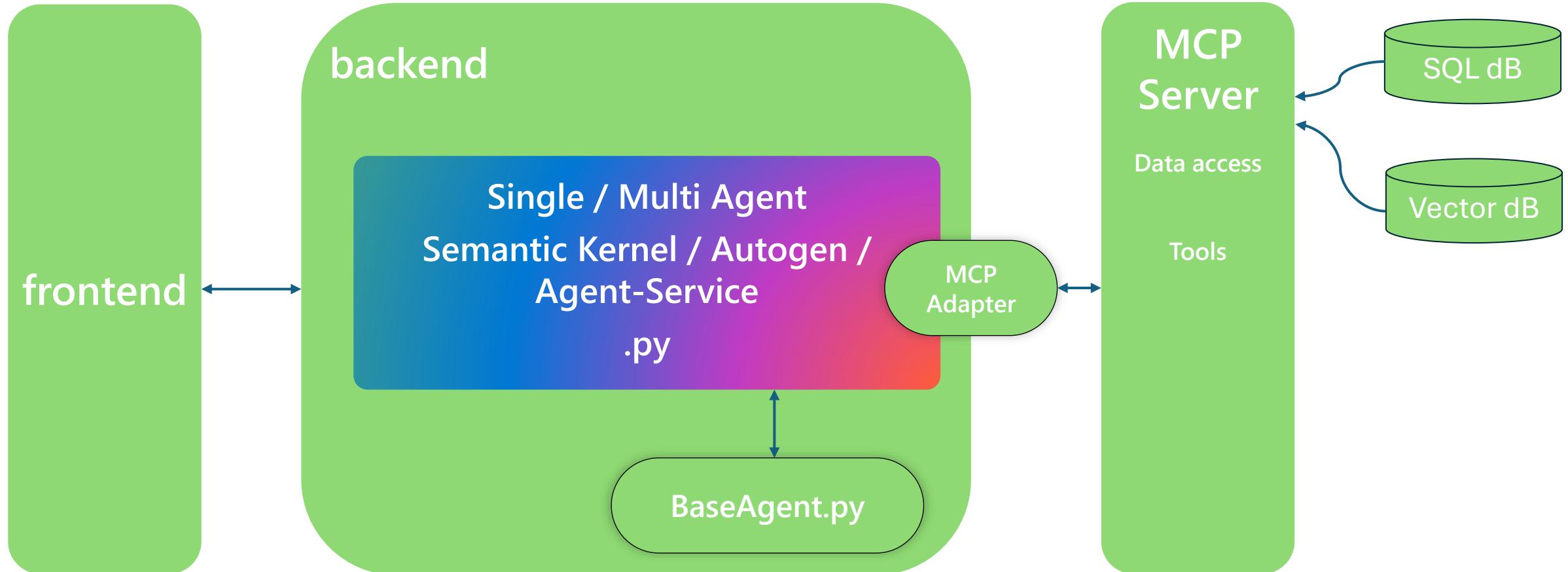
Exercises – Sample customer queries to try with your Agent

No.	Customer Query	Agent Analysis & Plan	Systems Accessed
1	I noticed my last invoice was higher than usual—can you help me understand why and what can be done about it?	<ul style="list-style-type: none">Check billing history and recent changes in subscriptions or charges.Review policy for invoice adjustments	CRM Billing Database Knowledge Base
2	My internet service seems slower than before—can you check what's happening?	<ul style="list-style-type: none">Review the customer's current subscription plan and service status.- Check for recent service incidents or network usage history.- Search knowledge base for troubleshooting guidance or policies on service quality guarantees.	CRM Service Monitoring and Diagnostic Systems Knowledge Base
3	I'm traveling abroad next month. What should I do about my phone plan?	<ul style="list-style-type: none">Check the customer's current mobile subscription, roaming capabilities, and international charges.- Search knowledge base for suitable international roaming options or temporary international plans.- Update the plan after verifying eligibility and availability.	CRM (Current Subscription). - Product & Service Database (Available Plans). - Knowledge Base (International Roaming Policies).
4			
5			

Solution Framework

Workshop
Development
Focus

Workshop Fixed
Resources



Azure AI Foundry



Azure OpenAI Service LLM Deployment

GPT-4o

Model Context Protocol (MCP) Benefits & Tools Used

MCP Benefits

- Make agent development accessible to more people
- Reduce the cost and complexity of integrating data sources
- Simplify adapting to changes made in data sources
- Standardize security and access control for data sources

The following services are exposed as tools for AI Agents:

- `get_all_customers` : List all customers with basic info.
- `get_customer_detail` : Get a full customer profile including their subscriptions.
- `get_subscription_detail` : Detailed subscription view including invoices (with payments) and service incidents.
- `get_invoice_payments` : Return invoice-level payments list.
- `pay_invoice` : Record a payment for a given invoice and get new outstanding balance.
- `get_data_usage` : Daily data-usage records for a subscription over a date range (optional aggregation).
- `get_promotions` : List every active promotion (no filtering).
- `get_eligible_promotions` : Promotions eligible for a given customer (evaluates basic loyalty/date criteria).
- `search_knowledge_base` : Semantic search on policy/procedure knowledge documents.
- `get_security_logs` : Security events for a customer (newest first).
- `get_customer_orders` : All orders placed by a customer.
- `get_support_tickets` : Retrieve support tickets for a customer (optionally filter by open status).
- `create_support_ticket` : Create a new support ticket for a customer.
- `get_products` : List or search available products (optional category filter).
- `get_product_detail` : Return a single product by ID.
- `update_subscription` : Update one or more mutable fields on a subscription.
- `unlock_account` : Unlock a customer account locked for security reasons.
- `get_billing_summary` : What does a customer currently owe across all subscriptions?

Solution Framework

your-agent.py

Workshop participants will need to modify this file

- Agent file template provided for different scenarios

```
sys.path.insert(0, str(Path(__file__).resolve().parent.parent))
agent_module_path = os.getenv("AGENT_MODULE")
agent_module = importlib.import_module(agent_module_path)
Agent = getattr(agent_module, "Agent")
```

```
SESSION_STORE = {}

app = FastAPI()
```

```
@app.post("/chat", response_model=ChatResponse)
async def chat(req: ChatRequest):
    # Lookup or create agent for this session
    agent = Agent(SESSION_STORE, req.session_id)
    # Run chat
    answer = await agent.chat_async(req.prompt)

    return ChatResponse(response=answer)
```

Solution Framework

backend.py

Workshop participants do not need to modify this file

- Common backend that calls "Agent"
- Create session store
- Which agent is being called is configured in .env file
- Uses FastAPI to
 - post ChatResponses
 - get ConversationHistory

```
sys.path.insert(0, str(Path(__file__).resolve().parent.parent))
agent_module_path = os.getenv("AGENT_MODULE")
agent_module = importlib.import_module(agent_module_path)
Agent = getattr(agent_module, "Agent")
```

```
SESSION_STORE = {}

app = FastAPI()
```

```
@app.post("/chat", response_model=ChatResponse)
async def chat(req: ChatRequest):
    # Lookup or create agent for this session
    agent = Agent(SESSION_STORE, req.session_id)
    # Run chat
    answer = await agent.chat_async(req.prompt)

    return ChatResponse(response=answer)
```

Solution Framework

frontend.py

Workshop participants do not need to modify this file

- Common app frontend
- Uses streamlit
- BACKEND_URL setup in .env file
- Communicates with backend and presents chat interface to user

```
BASE_BACKEND_URL = os.getenv("BACKEND_URL", "http://localhost:7000")
CHAT_URL = f"{BASE_BACKEND_URL}/chat"
HISTORY_URL = f"{BASE_BACKEND_URL}/history"
SESSION_RESET_URL = f"{BASE_BACKEND_URL}/reset_session"
```

```
# ----- Chat interaction -----
prompt = st.chat_input("Type a message...")
if prompt:
    with st.chat_message("user"):
        st.markdown(prompt)

    with st.spinner("Assistant is thinking..."):
        r = requests.post(
            CHAT_URL,
            json={"session_id": st.session_state["session_id"], "prompt": prompt},
        )
        r.raise_for_status()
        answer = r.json()["response"]

    with st.chat_message("assistant"):
        st.markdown(answer)
```

Solution Framework

basemodel.py

Workshop participants do not need to modify this file

- Common agent class

```
BASE_BACKEND_URL = os.getenv("BACKEND_URL", "http://localhost:7000")
CHAT_URL = f"{BASE_BACKEND_URL}/chat"
HISTORY_URL = f"{BASE_BACKEND_URL}/history"
SESSION_RESET_URL = f"{BASE_BACKEND_URL}/reset_session"
```

```
# ----- Chat interaction -----
prompt = st.chat_input("Type a message...")
if prompt:
    with st.chat_message("user"):
        st.markdown(prompt)

    with st.spinner("Assistant is thinking..."):
        r = requests.post(
            CHAT_URL,
            json={"session_id": st.session_state["session_id"], "prompt": prompt},
        )
        r.raise_for_status()
        answer = r.json()["response"]

    with st.chat_message("assistant"):
        st.markdown(answer)
```

DEMO & SETUP

Team Assignments

Name	Contact	1	2	3