# Microsoft

# GH-600

## Developing in Agentic AI Systems

# Latest Version: 6.0

## Topic 1, Contoso Ltd,

**Overview**
Contoso Ltd. is a software development company located in the United States.
**Existing Environment**
**GitHub Environment**
Contoso uses GitHub Enterprise and assigns GitHub Copilot Pro+ licenses to its developers.
The developers use Microsoft Visual Studio Code as their IDE.
Contoso has a customer portal. The code for the portal is stored in a GitHub repository named repo1 that contains the following:
A custom agent named agent1 that includes instructions to review specs related to best practices
A custom instruction file named validate-instructions.md that is used to validate tone of voice and applies to all .md and .txt files
A custom instruction file named codereview.instructions.md that is used by the Copilot coding agent but is excluded for use by the Copilot code review repo1 has the following structure:
The front-end is stored in the /frontend folder.
The API logic is stored in the /api folder.
Contoso has a second repository named repo2 that contains a legacy .NET application named App1 built by using .NET 6. repo2 has a multi-agent workflow for modernization tasks.
Contoso enables the Model Context Protocol (MCP) registry and allows the Microsoft Learn MCP Server. Every developer must configure their own connection to the Learn MCP Server.
**Problem Statements**
The developers working in repo1 report that the Microsoft Learn documentation is NOT being retrieved when they attempt to validate a design by using agent1.
The testing team at Contoso identifies that the customer portal uses inconsistent UI styles, which leads to customer confusion and branding issues. The UI inconsistencies stem from variations in the folder structure.
**Agent Logs**
You have the following logs for the multi-agent workflow used in repo2.
**Requirements**
**Planned Changes**
Contoso plans to have all agents and developers in repo1 use the Microsoft Learn MCP to ensure that reviews are validated by using the appropriate documentation. This must be implemented centrally.

Contoso plans to leverage AI-powered coding agents to implement new portal features and pages.

**Technical Requirements**

App1 must be upgraded to .NET 10. A previous upgrade attempt was started by using the Copilot modernization agent, but the attempt was never finalized.

You plan to retry the upgrade. You must first analyze App1 by using AI, and then generate a report that contains breaking changes and deprecated patterns before retrying the upgrade.

All AI-generated code for UI styling must adhere to a predefined folder structure.

The architects at Contoso need help building implementation plans for repo1. The company wants to implement a new agent named agent2 to analyze the code base and the code requirements, and then respond with a detailed plan. The agent must NOT be able to edit files or run local commands.

The developers must be able to delegate work to the Copilot coding agent by assigning issues to the agent.

## Question: 1

HOTSPOT

You need to implement agent2 to meet the technical requirements.

How should you complete the YAML configuration? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

name: implementation-planner

description: Creates detailed implementation plans and technical specifications in markdown format

tools: [

<Dropdown 1>,

<Dropdown 2>,

'microsoftdocs/mcp/docs_search',

'microsoftdocs/mcp/docs_fetch'

]

The accompanying image includes empty dropdown controls and recreated practice alternatives.

Answer Area

```
---
name: implementation-planner
description: Creates detailed implementation plans and technical specifications in markdown format
tools: [
        [ 'agent', 'edit', 'search' ]
        [ 'execute', 'read', 'web' ]
  'microsoftdocs/mcp/docs_search',
  'microsoftdocs/mcp/docs_fetch'
]
---
```

**Answer:**

Answer Area

```
---
name: implementation-planner
description: Creates detailed implementation plans and technical specifications in markdown format
tools: [
        [ 'agent', 'edit', 'search' ]
        [ 'execute', 'read', 'web' ]
  'microsoftdocs/mcp/docs_search',
  'microsoftdocs/mcp/docs_fetch'
]
---
```

## Question: 2

Before App1 is upgraded, you need to verify each individual upgrade step and whether all tests have passed.
Which file should you use?

A. <project>/.mcp/agent.md
B. <project>/.github/plan.md
C. <project>/.github/upgrades/{scenarioId}/assessment.md
D. <project>/.github/upgrades/{scenarioId}/tasks.md

**Answer: D**

Explanation:
The tasks.md file is the execution-tracking document for the upgrade scenario. It records individual tasks, their validation criteria, and their completion status. Microsoft's upgrade walkthrough specifically directs users to review this file when checking the status of every upgrade step.
A task can include several concrete verification activities, such as restoring dependencies, building the solution, correcting compilation errors, running the test suite, and rerunning tests after fixes. Reviewing these entries reveals whether implementation has merely been attempted or has satisfied the defined checks.
The assessment identifies upgrade issues and scope; it does not serve as the live record of completed work. A plan describes intended actions and ordering, which also differs from evidence that execution and testing succeeded.
The phrase "before App1 is upgraded" requires temporal care. Before execution, tasks.md identifies the steps and checks that must be performed. It cannot prove that future tests have already passed. Completion and test results become available as the agent executes and updates the tasks.
Study-guide topics: execution validation, task status, and acceptance criteria. Microsoft Learn—Execute and verify a Copilot upgrade.
==========

## Question: 3

You need to make changes to repo1 to support the planned changes for the agents.
What should you modify?

A. <project>/.mcp/server.json
B. .vscode/settings.json
C. .github/agents/*.agent.md
D. .vscode/mcp.json

**Answer: C**

Explanation:
Repository custom agent profiles belong in .github/agents/ and use Markdown files with agent configuration and instructions. When the intended changes concern the agents' roles, available tools, or operating guidance, these profiles are the appropriate implementation point.

An agent profile provides a durable definition that can be shared with other repository contributors. Keeping that definition under version control makes changes reviewable and enables the team to associate a particular agent configuration with the code revision used during evaluation. This is useful when diagnosing why an agent's behavior changed after its instructions or capabilities were modified.
Editor settings and MCP connection settings address different concerns. They can affect the environment in which an agent operates, but they do not replace the agent's own profile. Changing a server connection is appropriate for transport or authentication requirements; changing an agent profile is appropriate for agent behavior and capability selection.
The source selects C. Its applicability depends on the omitted planned changes actually concerning custom agent configuration.
Study-guide topics: agent profiles, configuration management, and SDLC traceability. GitHub—Creating custom agents.
==========

## Question: 4

You need to troubleshoot the issue reported by Dev1.
What should you review?

A. The GitHub Actions usage metrics of repo1.
B. The agent session log in the Agents panel.
C. The GitHub Actions runner log for the session job.
D. The GITHUB_TOKEN permissions block in the agent1 workflow.

**Answer: B**

Explanation:
The agent session log is the appropriate starting point when the reported problem concerns an agent's interaction, execution sequence, or tool activity within the development environment. It provides information closer to the failing behavior than repository-level workflow usage statistics.
A useful investigation establishes what the agent was asked to do, which operations it attempted, what responses those operations returned, and where progress stopped or diverged from the intended result. Session diagnostics can help distinguish an instruction problem from an unavailable tool, a rejected operation, or an execution error. VS Code also provides detailed chat debugging facilities for inspecting interactions and diagnosing failures.
GitHub Actions usage metrics primarily describe workflow consumption. Runner logs are relevant when evidence points to a hosted workflow or runner problem. The token permissions block becomes relevant when the failure concerns authorization to a GitHub resource. None should be assumed to be the root cause before examining the reported execution.
The PDF selects B; the underlying Dev1 incident description is not included, so this selection assumes a session-level agent issue.
Study-guide topics: diagnostic evidence, tool-call inspection, and failure localization. VS Code—Debug chat interactions.

## Question: 5

You need to enable Copilot memory to support the planned changes.
What should you configure?

A. The personal Copilot settings of each developer
B. The Copilot settings of each repository
C. The agent profile of agent1
D. The Copilot settings of the organization

## Answer: D

Explanation:
Copilot memory is an organization-level capability. Configuring it through the organization's Copilot settings establishes the governance boundary, availability, and policy controls consistently for the developers and repositories covered by that organization.
Personal settings are unsuitable because they create inconsistent behavior between developers and do not provide centralized administration. A repository-level configuration can supply repository instructions and scoped context, but it does not replace the organization-level control that enables and governs the memory capability. An individual agent profile defines role behavior, tool access, prompts, and delegation settings; it is not the tenant-level location for enabling Copilot memory.
Organization configuration is particularly important when agents must retain approved context, recurring pReference, and workflow guidance across sessions without each developer configuring behavior independently. Central governance also allows administrators to maintain predictable controls over which organizational information is available to Copilot.
Study-guide topics: organizational governance, persistent context, and centralized agent configuration.

# Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

**15 USD Discount Coupon Code:**

**G74JA8UF**

# FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer