Activity 1: Writing and Enhancing API Code with Copilot

Activity Introduction

Generative AI tools like Microsoft Copilot are powerful assistants for writing code that generate code based on your instructions.

In this activity, you will use Copilot to write and enhance API code for a User Management API. This project will give you practice writing code yourself and using Copilot to help. You'll use Copilot to generate code and then enhance and test that code.

This is the first of three activities in which you will develop and code a back-end API project. The final output will be a working API project that you can use to demonstrate your understanding of back-end development.

Activity Instructions

Step 1: Review the scenario

To begin, review the following scenario to understand the intent of this API project.

You've been hired by TechHive Solutions to develop a User Management API for their internal tools. The HR and IT departments need an API that allows them to create, update, retrieve, and delete user records efficiently. Your task is to build the core functionality of the API, using Microsoft Copilot to scaffold, enhance, and test the code.

Step 2: Set up the project

Next, start setting up your project.

- Create a new ASP.NET Core Web API project named UserManagementAPI.
- Use Microsoft Copilot to scaffold the project setup, including adding boilerplate code to Program.cs.

Step 3: Generate API endpoints

Then use Copilot to help you generate API endpoints.

- Use Copilot to generate CRUD endpoints for managing users:
    - GET: Retrieve a list of users or a specific user by ID.
    - POST: Add a new user.
    - PUT: Update an existing user's details.

- DELETE: Remove a user by ID.

Step 4: Test API functionality

Finally, to complete this project phase, test the API.

- Use Postman or a similar tool to test all CRUD endpoints.
- Document the specific ways Microsoft Copilot assisted in improving the API code and functionality.

Step 5: Save your work

By the end of this exercise, you should have a functional User Management API with robust CRUD operations and documentation of how Copilot contributed to code generation and enhancement. When completed, save your work. You will use this code in later activities.

Activity 2: Debugging API Code with Copilot

Activity Introduction

Generative AI tools like Microsoft Copilot can streamline the debugging process and improve your final code output.

In this activity, you will use Copilot to debug the code you've started for your API project. This project will give you practice reviewing code, debugging, and implementing fixes. You'll use Copilot to discover issues and suggest changes to improve your code.

This is the second of three activities in which you will develop and code a back-end API project. The final output will be a working API project that you can use to demonstrate your understanding of back-end development.

Before you start this activity, make sure you have completed the first activity in this module, Writing and Enhancing API Code with Copilot [LINK].

Activity Instructions

Step 1: Review the scenario

To begin, review the following scenario:

After deploying the initial version of the User Management API, TechHive Solutions reported several bugs. For example:

- Users were being added without proper validation.
- Errors occurred when retrieving non-existent users.
- The API occasionally crashed due to unhandled exceptions.

Your task is to debug the API using Microsoft Copilot, ensuring it works reliably and meets the company's requirements.

Step 2: Identify bugs

Next, start finding issues with the code you created previously.

- Use Copilot to analyze the existing codebase and identify issues.Examples:
    - Missing validation for user input fields (e.g., empty names or invalid emails).
    - Lack of error handling for failed database lookups.
    - Performance bottlenecks in the GET /users endpoint.

Step 3: Fix bugs with Copilot

Then, use Copilot to fix any issues that you have identified.

- Use Copilot to:
    - Add validation to ensure only valid user data is processed.
    - Implement try-catch blocks to handle unhandled exceptions.
    - Optimize queries or logic to improve performance.

Step 4: Test and validate fixes

Finally, test the debugged code to complete this project phase.

- Test the API endpoints again, focusing on edge cases (e.g., invalid input, non-existent IDs).
- Document how Copilot's suggestions helped identify and resolve issues.

Step 5: Save your work

By the end of this exercise, you should have a debugged and optimized User Management API and a summary of bugs fixed and how Copilot streamlined the debugging process. When completed, save your work. You will use this code in later activities.

Activity 3: Implementing and Managing Middleware with Microsoft Copilot

Activity Introduction

Generative AI tools like Microsoft Copilot can help you implement and manage middleware in your back-end projects.

In this activity, you will use Copilot to create middleware for logging, authentication, and error handling and configure the middleware pipeline. This project will give you practice working with middleware, from implementation to management. Copilot can help with every step of this process.

This is the last of three activities in which you will develop and code a back-end API project. The final output will be a working API project that you can use to demonstrate your understanding of back-end development.

Before you start this activity, make sure you have completed the previous two activities in this module, Writing and Enhancing API Code with Copilot and Debugging API Code with Copilot

Activity Instructions

Step 1: Review the scenario

To begin, review the following scenario:

To comply with corporate policies, TechHive Solutions requires middleware in the User Management API to:

- Log all incoming requests and outgoing responses for auditing purposes.
- Enforce standardized error handling across all endpoints.
- Secure API endpoints using token-based authentication.

Your task is implementing these middleware components using Microsoft Copilot and configuring the middleware pipeline for optimal performance.

Step 2: Implement logging middleware

Next, start implementing middleware into your project that you've worked on in the previous activities.

- Use Copilot to write middleware that logs:
    - HTTP method (e.g., GET, POST).

    - Request path.
    - Response status code.

- Example Copilot prompt: *"Generate middleware to log HTTP requests and responses in ASP.NET Core."*

Step 3: Implement error-handling middleware

Then, create the next piece of middleware.

- Use Copilot to create middleware that:
    - Catches unhandled exceptions.
    - Return consistent error responses in JSON format (e.g., { "error": "Internal server error." }).

Step 4: Implement authentication middleware

Next, create another middleware.

- Use Copilot to write middleware that:
    - Validates tokens from incoming requests.
    - Allows access only to users with valid tokens.
    - Returns a 401 Unauthorized response for invalid tokens.

Step 5: Configure the middleware pipeline

With your middleware in place, make sure the middleware is configured correctly.

- Use Copilot to ensure middleware is configured in the correct order:
    - Error-handling middleware first.
    - Authentication middleware next.
    - Logging middleware last.

Step 6: Test middleware functionality

Finally, test the code to complete this project phase.

- Test the middleware by sending various requests (e.g., valid/invalid tokens, triggering exceptions).

- Validate that logs are accurate and errors are handled consistently.

Step 7: Save your work

By the end of this exercise, you should have middleware for logging, error handling, and authentication integrated into the User Management API and a middleware pipeline optimized for performance and security. When completed, save your work. You will submit this code project for review.