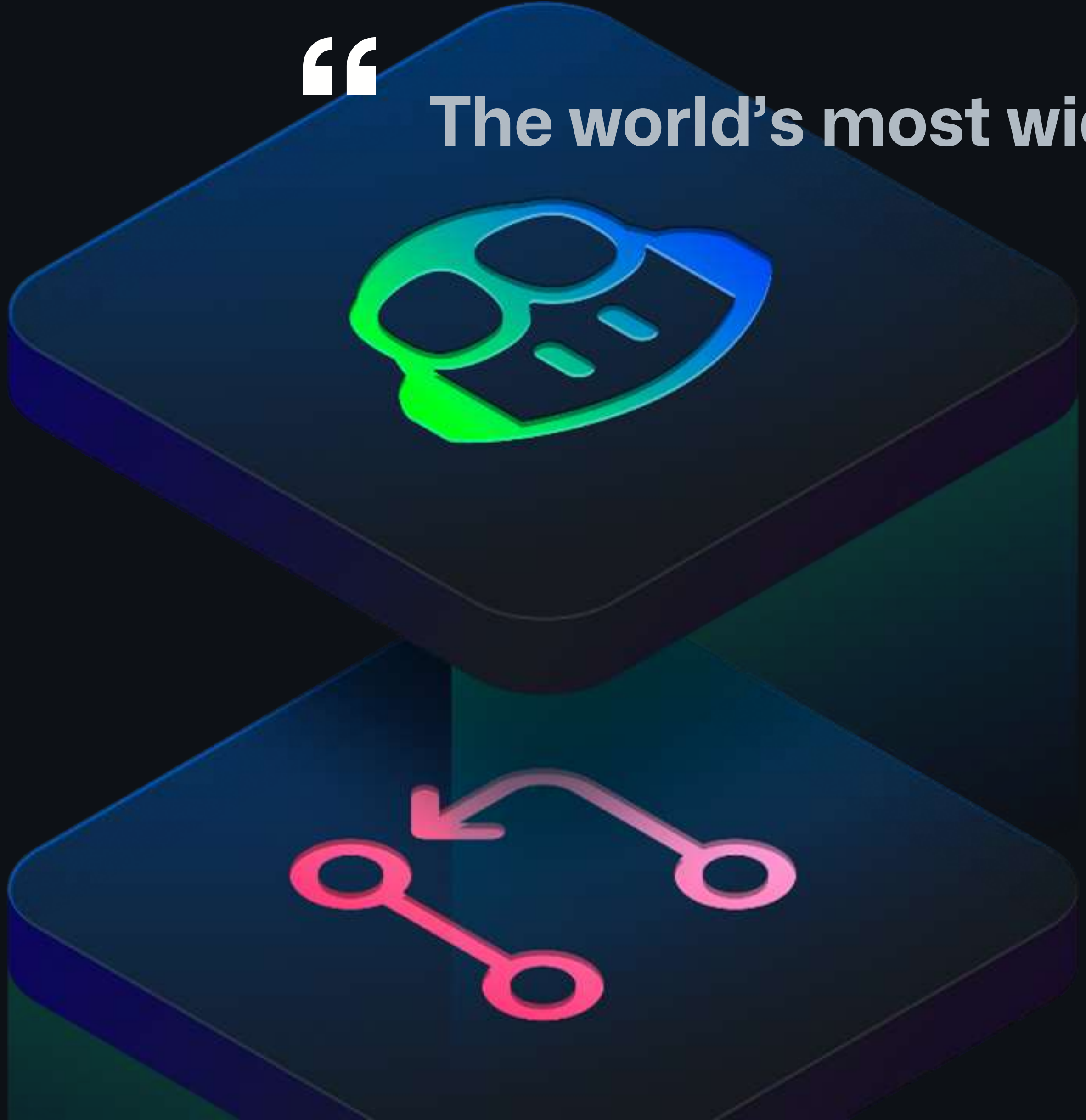
GitHub

Multi-Agents Application Development With GitHub Copilot Workshop

Shinya Yanagihara / 柳原伸弥
Developer Productivity GBB
Microsoft Corporation

Do you know GitHub Copilot?

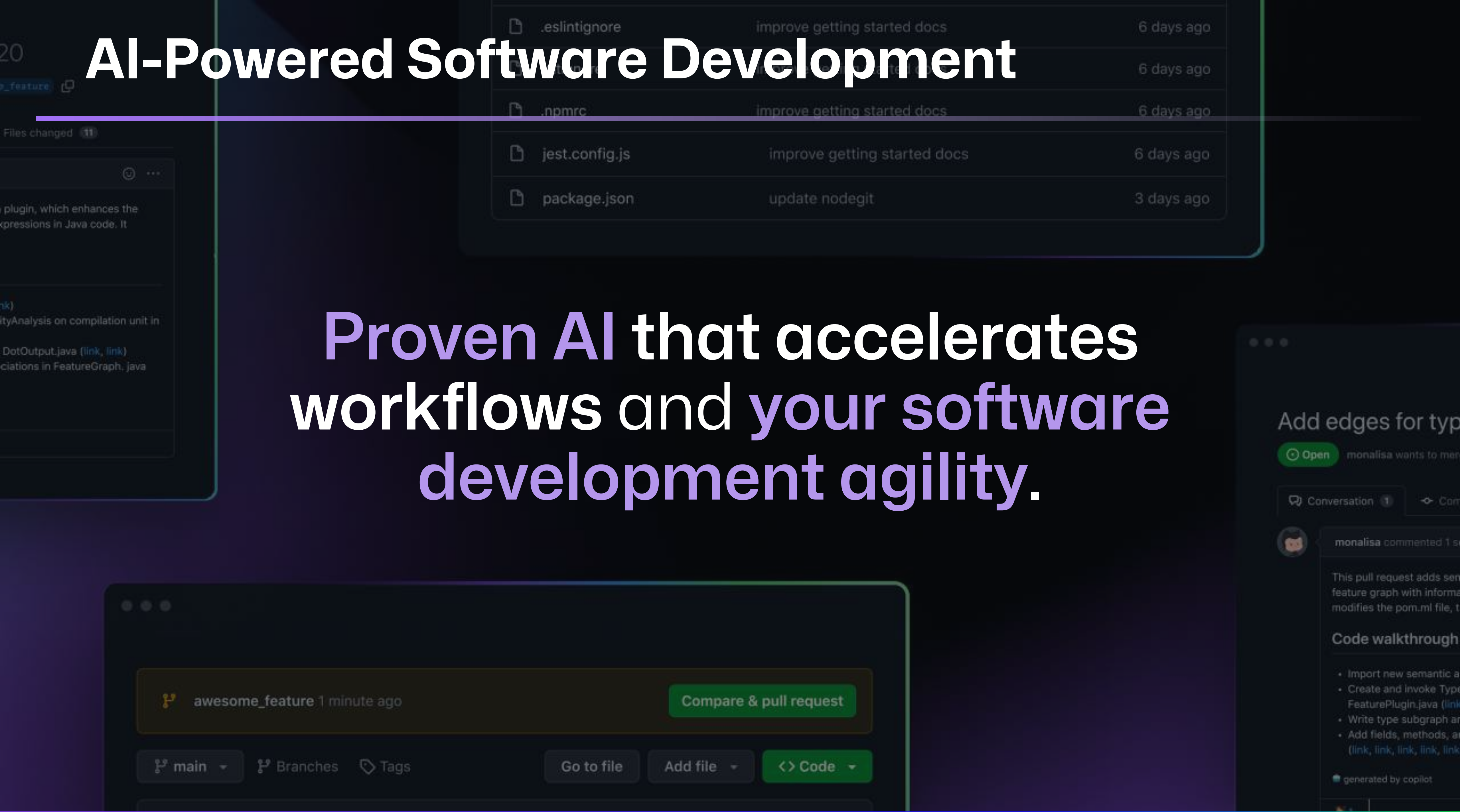
The Copilot Effect



> "The world's most widely adopted AI developer tool."

- Maximize developer happiness
- Increase developer productivity
- Accelerate software development

AI-Powered Software Development



Proven AI that accelerates workflows and your software development agility.

Developer Productivity with GitHub Copilot

87 %

less **mental effort** on repetitive tasks

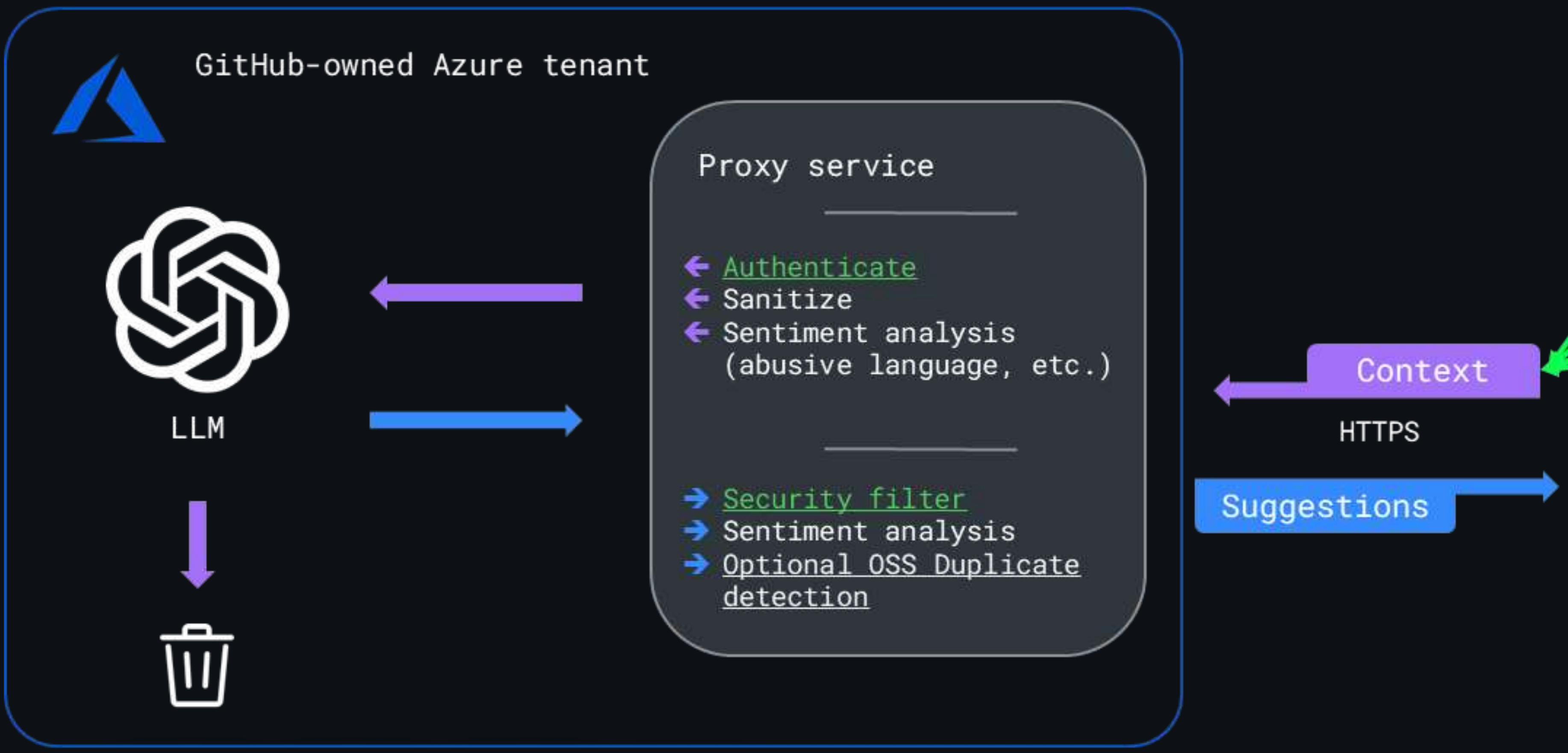
74 %

focus on more **satisfying** work

73 %

say they are more **in the flow**

GitHub Copilot and its Dataflow



GitHub-owned Azure tenant

LLM

Proxy service

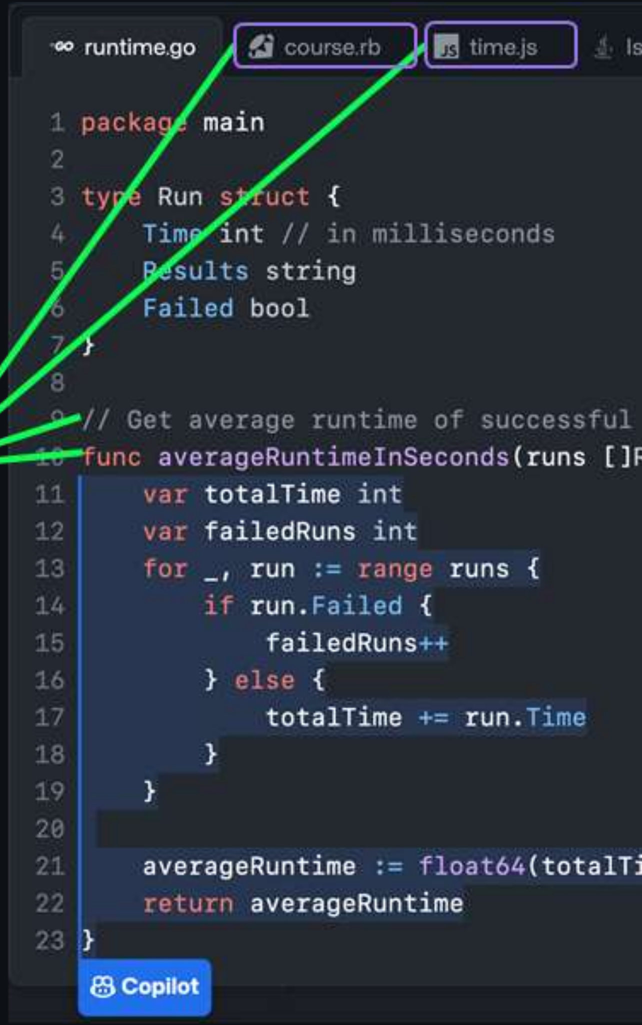
- Authenticate
- Sanitize
- Sentiment analysis (abusive language, etc.)

- Security filter
- Sentiment analysis
- Optional OSS Duplicate detection

Context

HTTPS

Suggestions



```go
package main

type Run struct {
    Time int // in milliseconds
    Results string
    Failed bool
}

// Get average runtime of successful
func averageRuntimeInSeconds(runs []R
    var totalTime int
    var failedRuns int
    for _, run := range runs {
        if run.Failed {
            failedRuns++
        } else {
            totalTime += run.Time
        }
    }

    averageRuntime := float64(totalTi
    return averageRuntime
}
```

Visual Studio

Neovim

VS Code

JetBrains IDEs

Xcode

Eclipse

GitHub Copilot RAG

Copilot-instructions.md

Coding Response Instructions
- if responding with code that include a class, function or method definition, add doc-string comments.
- review for possible exceptions and add exception handling.

Coding Standards
Naming Standards
- C# and Java: Camel Case
- Python: Snake Case
- Other Languages:
  Follow the naming convention of the code context provided.

Magic Strings
- Avoid using magic strings. Either parameterize or create constants.

Additional Instructions
Revalidate before responding. Think step by step.

+

Prompt File

contoso-dev.prompt.md

Context

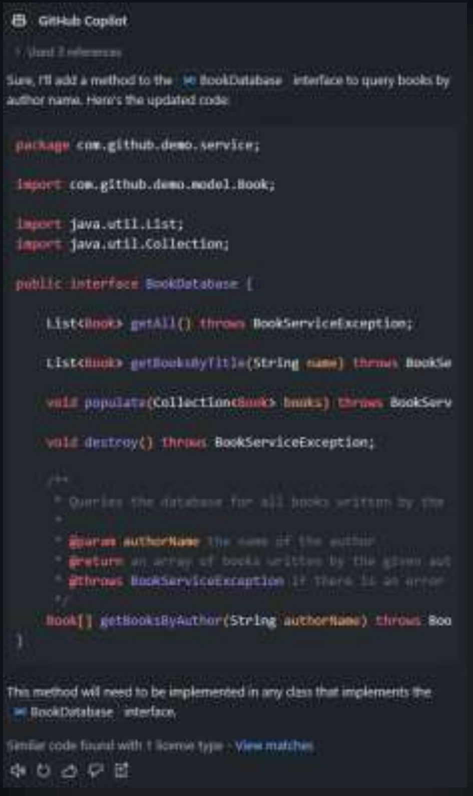
BookUtils.java
BookDatabaseImpl.java
...

Prompt

Generate a method that queries the database for all books that was written by the given author.
- input: author
- output: Book array

=

Response



Sure, I'll add a method to the BookDatabase interface to query books by author name. Here's the updated code:

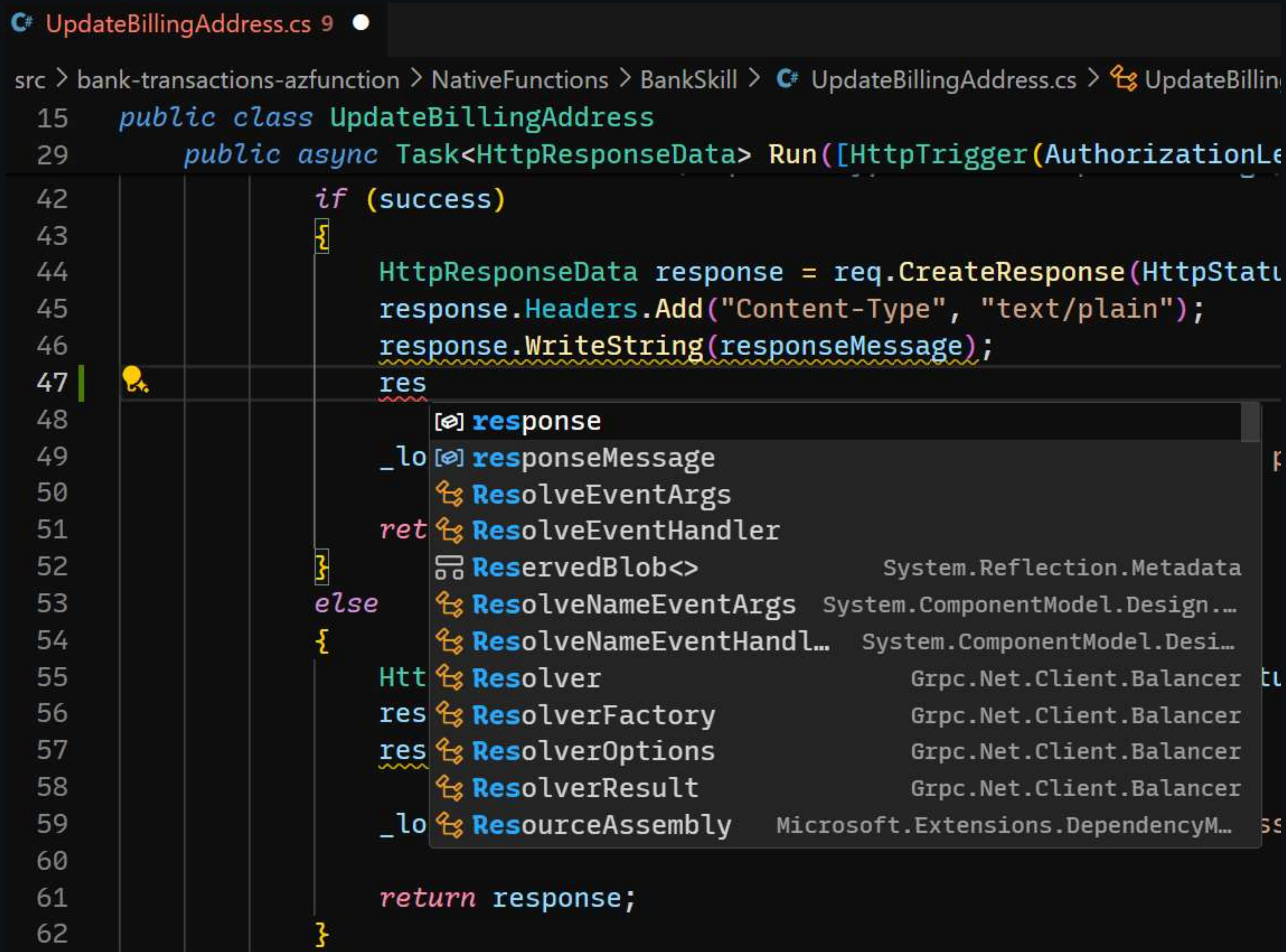
```
package com.github.demo.service;

import com.github.demo.model.Book;

import java.util.List;
import java.util.Collection;

public interface BookDatabase {

    List<Book> getAll() throws BookServiceException;

    List<Book> getBooksByTitle(String name) throws BookSe

    void populate(Collection<Book> books) throws BookServ

    void destroy() throws BookServiceException;

    /**
     * Queries the database for all books written by the
     *
     * @param authorName the name of the author
     * @return an array of books written by the given aut
     * @throws BookServiceException if there is an error
     */
    Book[] getBooksByAuthor(String authorName) throws Boo
}
```

This method will need to be implemented in any class that implements the BookDatabase interface.

Similar code found with 1 license type - View matches

GitHub Copilot
IDE Features

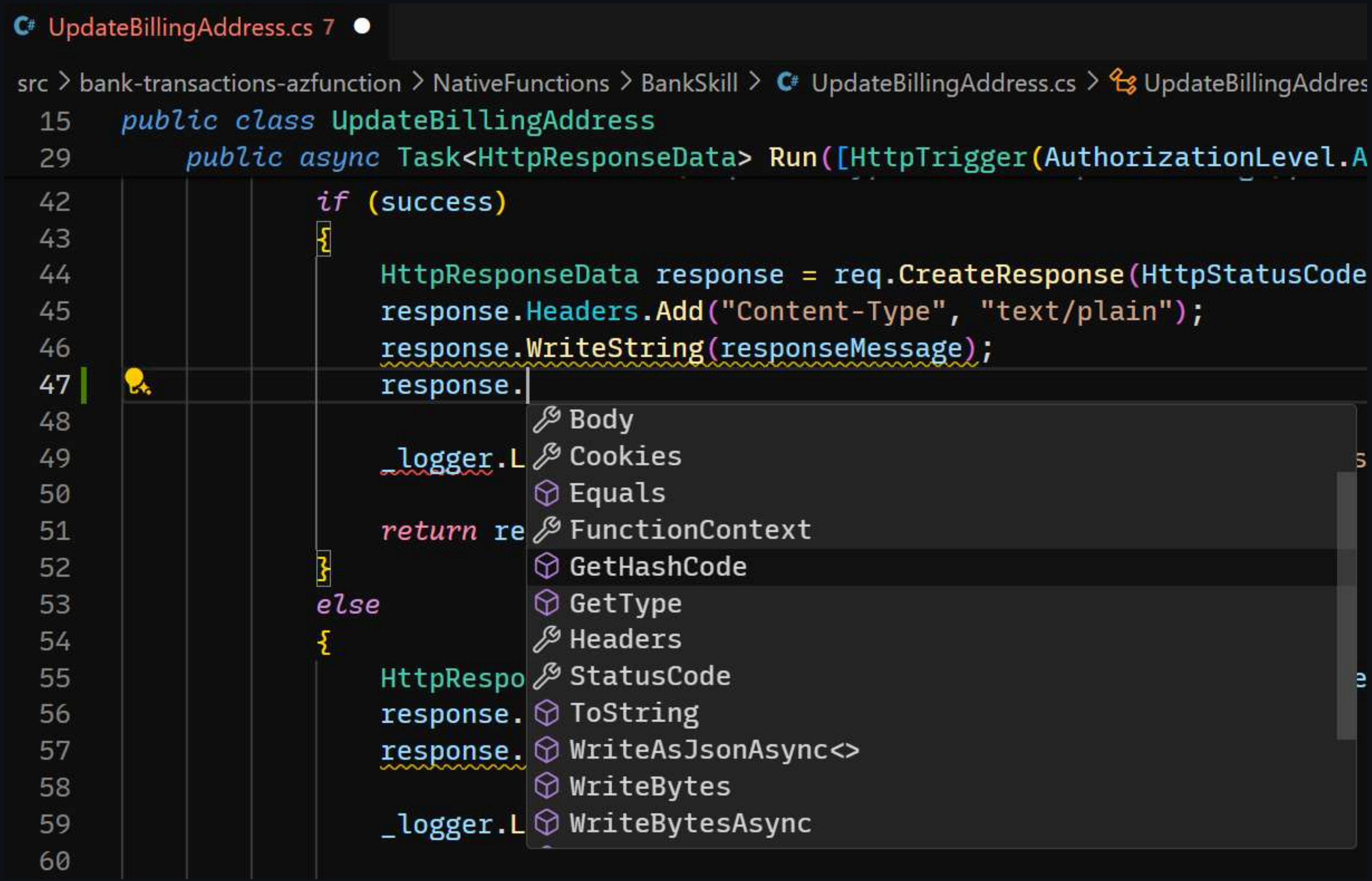
IDE Features - Autocomplete


UpdateBillingAddress.cs 9
src > bank-transactions-azfunction > NativeFunctions > BankSkill > UpdateBillingAddress.cs > UpdateBillin

```
public class UpdateBillingAddress
    public async Task<HttpResponseData> Run([HttpTrigger(AuthorizationLe
            if (success)
            {
                HttpResponseData response = req.CreateResponse(HttpStatu
                response.Headers.Add("Content-Type", "text/plain");
                response.WriteString(responseMessage);
                res

                _lo

                ret
            }
            else
            {
                Htt
                res
                res

                _lo

                return response;
            }
```

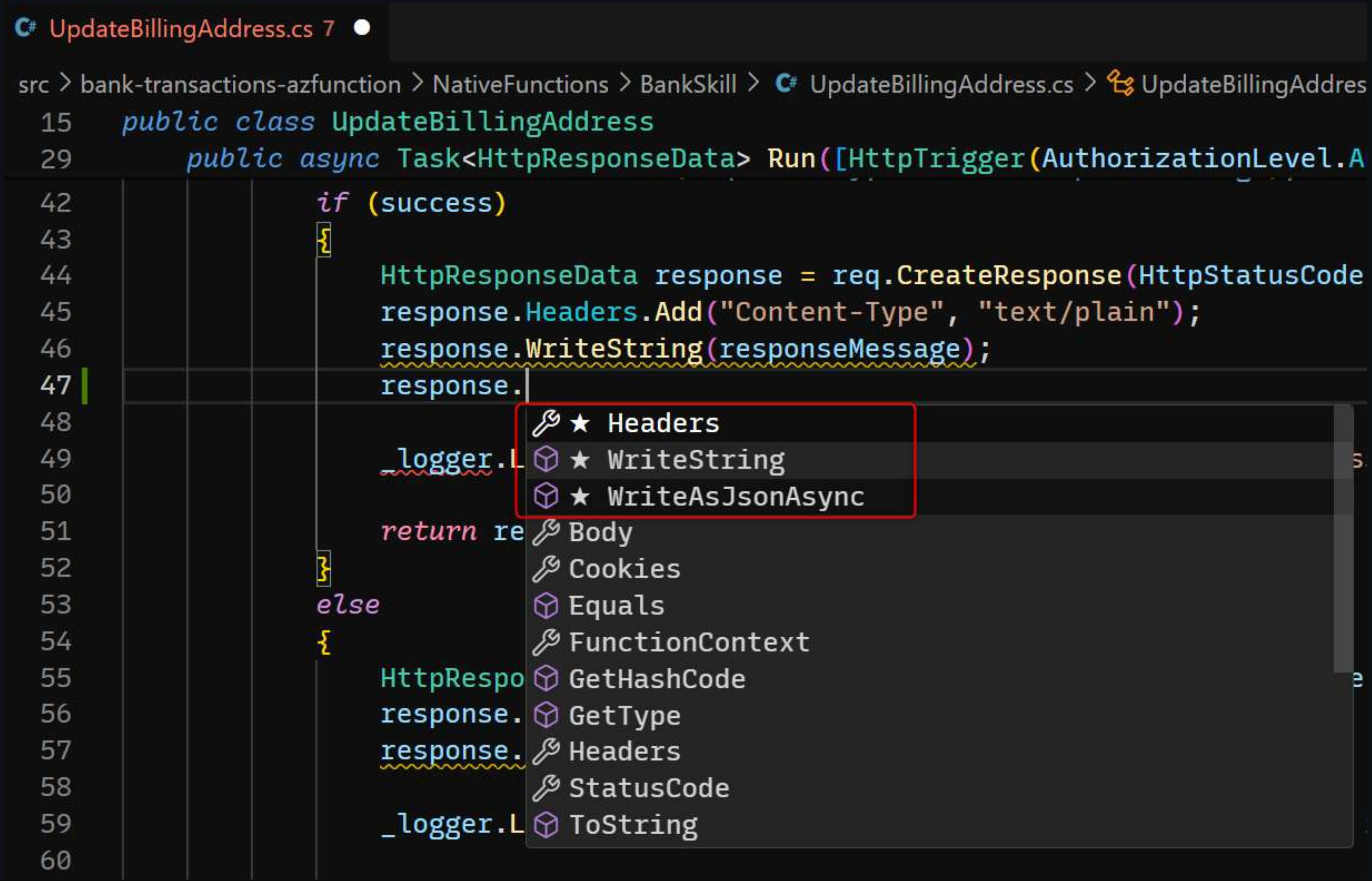
response
responseMessage
ResolveEventArgs
ResolveEventHandler
ReservedBlob<> System.Reflection.Metadata
ResolveNameEventArgs System.ComponentModel.Design....
ResolveNameEventHandl... System.ComponentModel.Desi...
Resolver Grpc.Net.Client.Balancer
ResolverFactory Grpc.Net.Client.Balancer
ResolverOptions Grpc.Net.Client.Balancer
ResolverResult Grpc.Net.Client.Balancer
ResourceAssembly Microsoft.Extensions.DependencyM...

IntelliSense


UpdateBillingAddress.cs 7
src > bank-transactions-azfunction > NativeFunctions > BankSkill > UpdateBillingAddress.cs > UpdateBillingAddres
Body
Cookies
Equals
FunctionContext
GetHashCode
GetType
Headers
StatusCode
ToString
WriteAsJsonAsync<>
WriteBytes
WriteBytesAsync

```
public class UpdateBillingAddress
    public async Task<HttpResponseData> Run([HttpTrigger(AuthorizationLevel.A
            if (success)
            {
                HttpResponseData response = req.CreateResponse(HttpStatusCode
                response.Headers.Add("Content-Type", "text/plain");
                response.WriteString(responseMessage);
                response.

                _logger.L                                                  s

                return re
            }
            else
            {
                HttpRespo                                                  e
                response.
                response.

                _logger.L

```

IntelliCode


UpdateBillingAddress.cs 7
src > bank-transactions-azfunction > NativeFunctions > BankSkill > UpdateBillingAddress.cs > UpdateBillingAddres
★ Headers
★ WriteString
★ WriteAsJsonAsync
Body
Cookies
Equals
FunctionContext
GetHashCode
GetType
Headers
StatusCode
ToString

```
public class UpdateBillingAddress
    public async Task<HttpResponseData> Run([HttpTrigger(AuthorizationLevel.A
            if (success)
            {
                HttpResponseData response = req.CreateResponse(HttpStatusCode
                response.Headers.Add("Content-Type", "text/plain");
                response.WriteString(responseMessage);
                response.

                _logger.L

                return re
            }
            else
            {
                HttpRespo
                response.
                response.

                _logger.L

```

Other features

Generate unit tests

Copilot Chat can help write unit test cases by generating code snippets based on the existing code in the editor or the code snippet highlighted in the editor by the user.

Explain code

Copilot Chat can help explain selected code by generating natural language descriptions of the code's functionality and purpose.

Propose code fixes

Copilot Chat can help propose a fix for bugs in your code by suggesting code snippets and solutions based on the context of the error or issue.

Answer coding questions

Copilot Chat allows you to ask for help or clarification on specific coding problems and receive responses in natural language format or in code snippet format.

Translate languages

Copilot Chat can help you translate code from one programming language to a different language.

Increase readability

Copilot Chat can help you increase code readability by adding missing comments and suggesting such as better variable names.

GitHub Copilot Agent mode

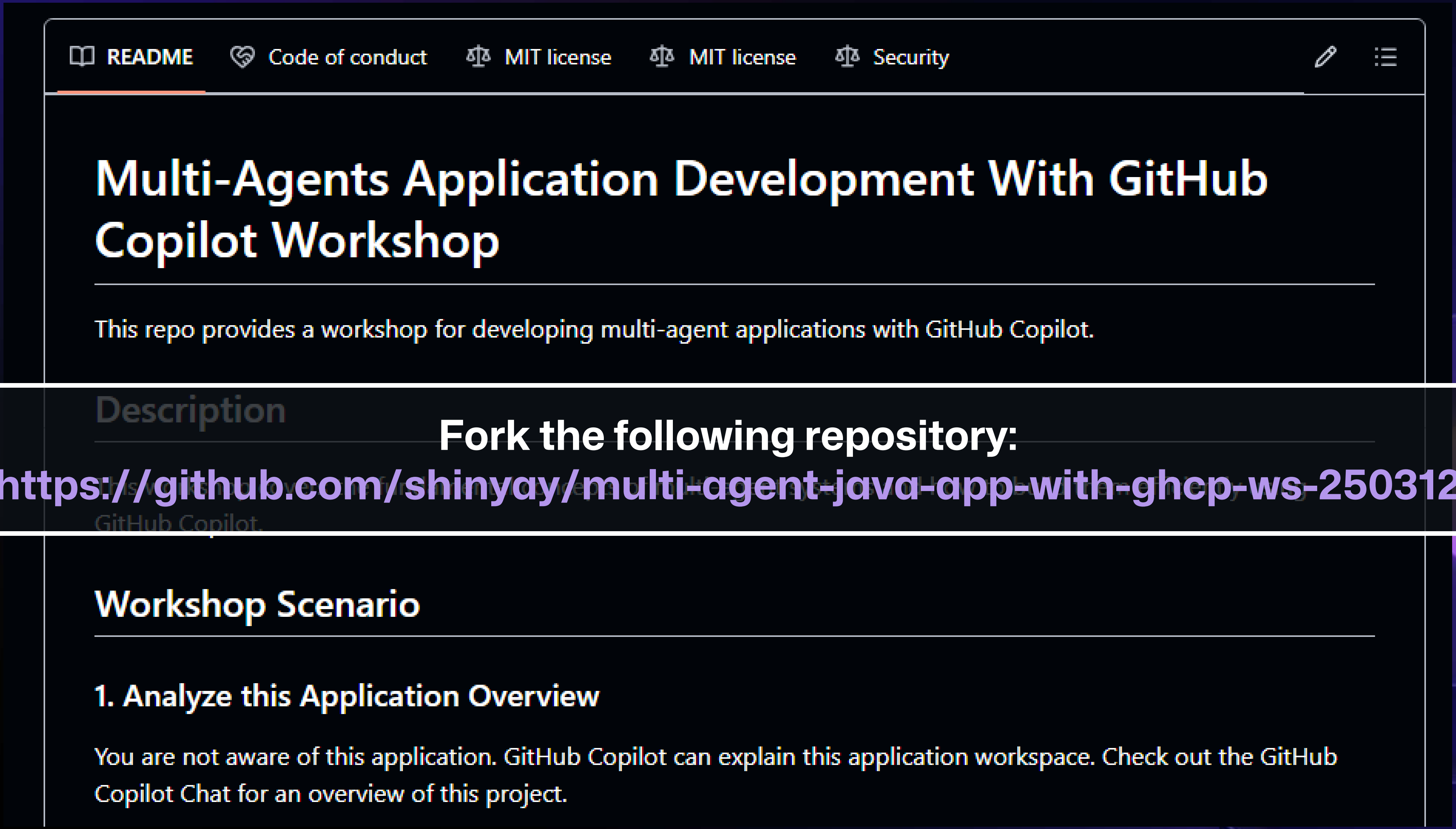
Agent mode is an editing feature that automatically:

- Searches your codebase
- Identifies and reads relevant files
- Executes shell commands (with your confirmation)
- Processes errors
- Applies edits to multiple files
- Insiders

All in one streamlined workflow

Workshop

0. Prepare for the workshop



Multi-Agents Application Development With GitHub Copilot Workshop

This repo provides a workshop for developing multi-agent applications with GitHub Copilot.

Description

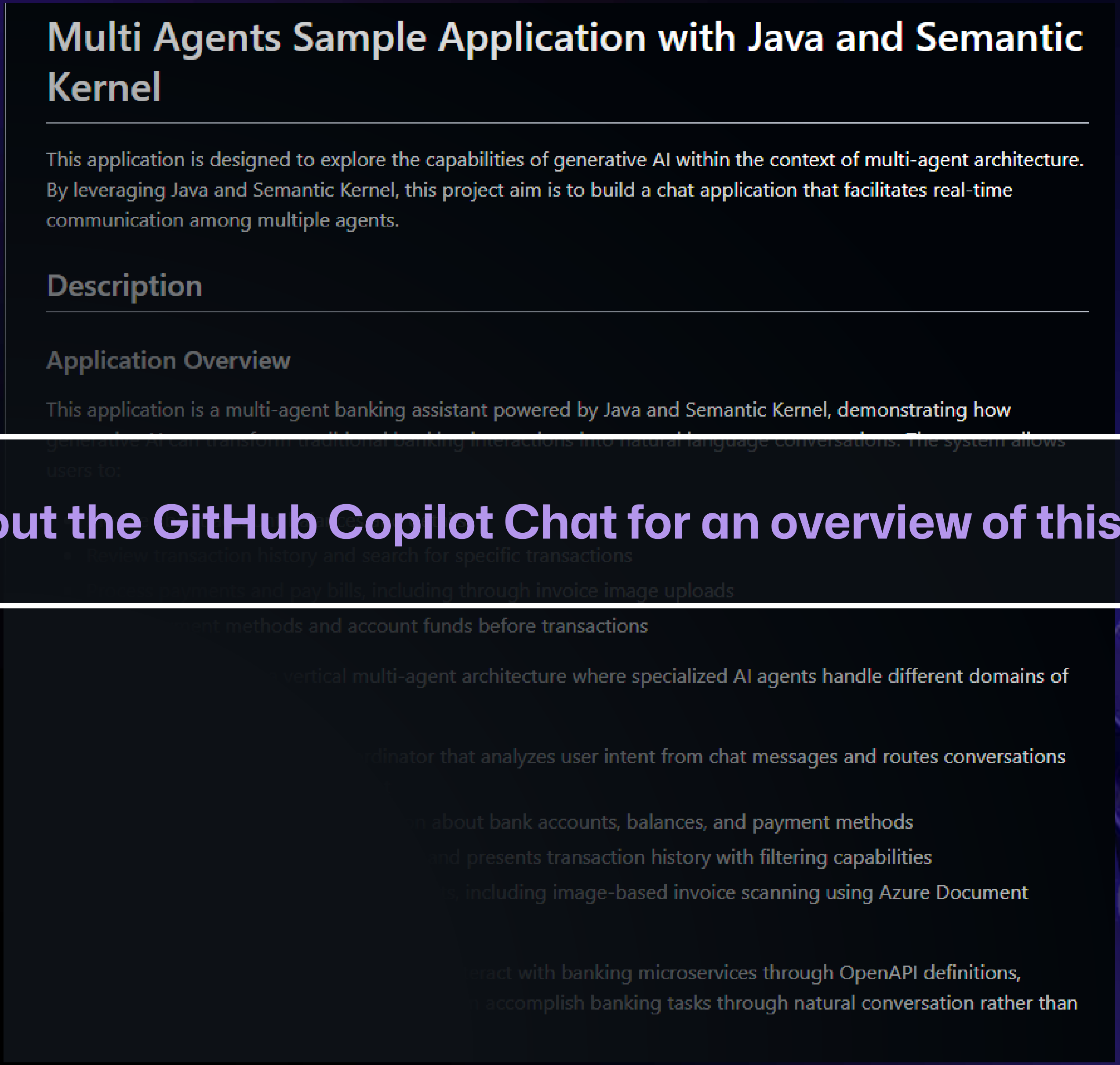
Workshop Scenario

1. Analyze this Application Overview

You are not aware of this application. GitHub Copilot can explain this application workspace. Check out the GitHub Copilot Chat for an overview of this project.

Fork the following repository:
https://github.com/shinyay/multi-agent-java-app-with-ghcp-ws-250312

1. Analyze this Application Overview



Multi Agents Sample Application with Java and Semantic Kernel

This application is designed to explore the capabilities of generative AI within the context of multi-agent architecture. By leveraging Java and Semantic Kernel, this project aim is to build a chat application that facilitates real-time communication among multiple agents.

Description

Application Overview

This application is a multi-agent banking assistant powered by Java and Semantic Kernel, demonstrating how generative AI can transform traditional banking interactions into natural language conversations. The system allows users to:

- Review transaction history and search for specific transactions
- Process payments and pay bills, including through invoice image uploads
- ment methods and account funds before transactions

vertical multi-agent architecture where specialized AI agents handle different domains of

dinator that analyzes user intent from chat messages and routes conversations

about bank accounts, balances, and payment methods

nd presents transaction history with filtering capabilities

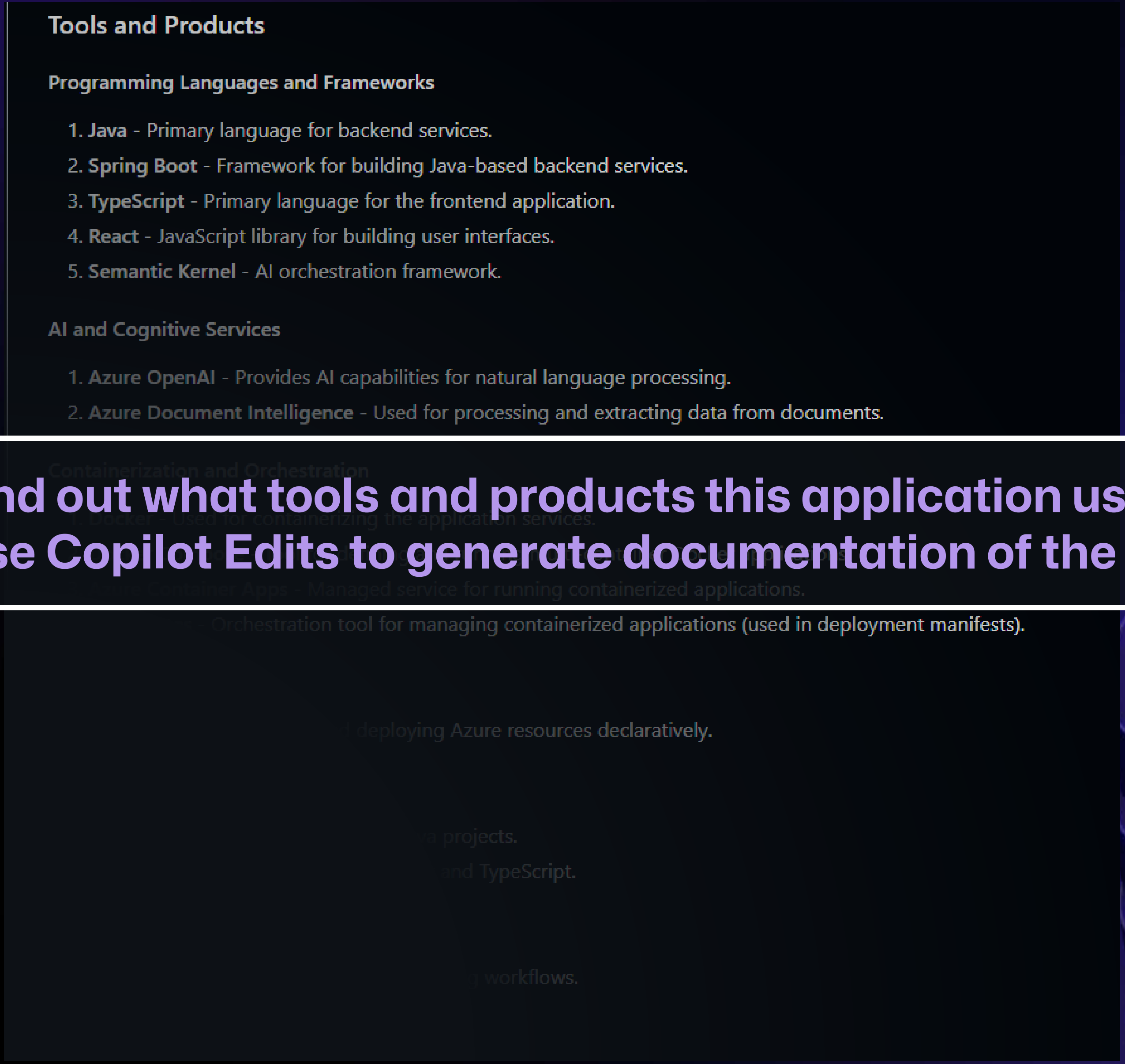
, including image-based invoice scanning using Azure Document

act with banking microservices through OpenAPI definitions,

accomplish banking tasks through natural conversation rather than

Check out the GitHub Copilot Chat for an overview of this project.

2. Analyze the tools and products



Tools and Products

Programming Languages and Frameworks

1. **Java** - Primary language for backend services.
2. **Spring Boot** - Framework for building Java-based backend services.
3. **TypeScript** - Primary language for the frontend application.
4. **React** - JavaScript library for building user interfaces.
5. **Semantic Kernel** - AI orchestration framework.

AI and Cognitive Services

1. **Azure OpenAI** - Provides AI capabilities for natural language processing.
2. **Azure Document Intelligence** - Used for processing and extracting data from documents.

Containerization and Orchestration

1. **Docker** - Used for containerizing the application services.
3. **Azure Container Apps** - Managed service for running containerized applications.

- Orchestration tool for managing containerized applications (used in deployment manifests).

deploying Azure resources declaratively.

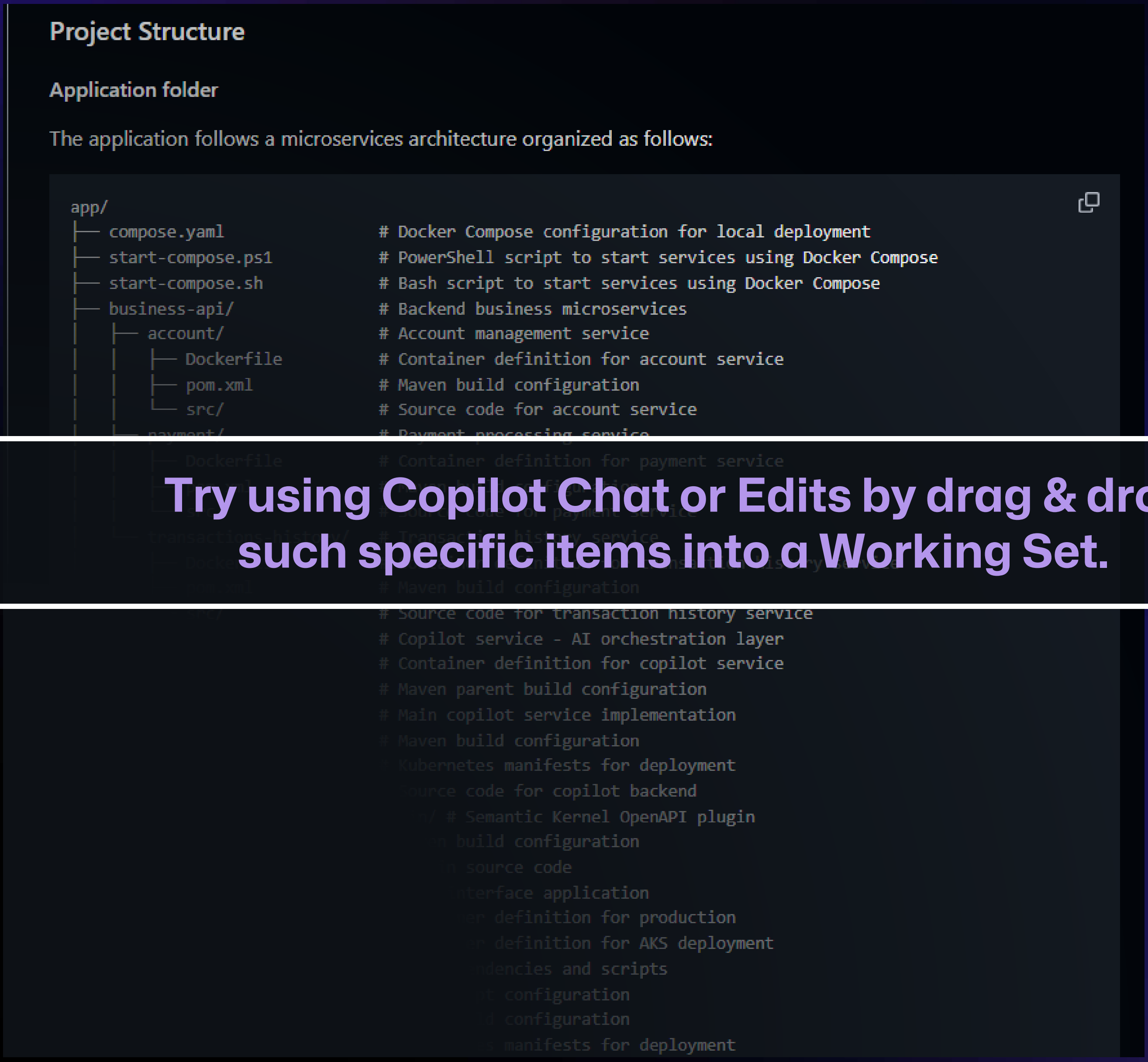
projects.

and TypeScript.

workflows.

Find out what tools and products this application uses.
Then use Copilot Edits to generate documentation of the results.

3. Explore the specific item



Project Structure

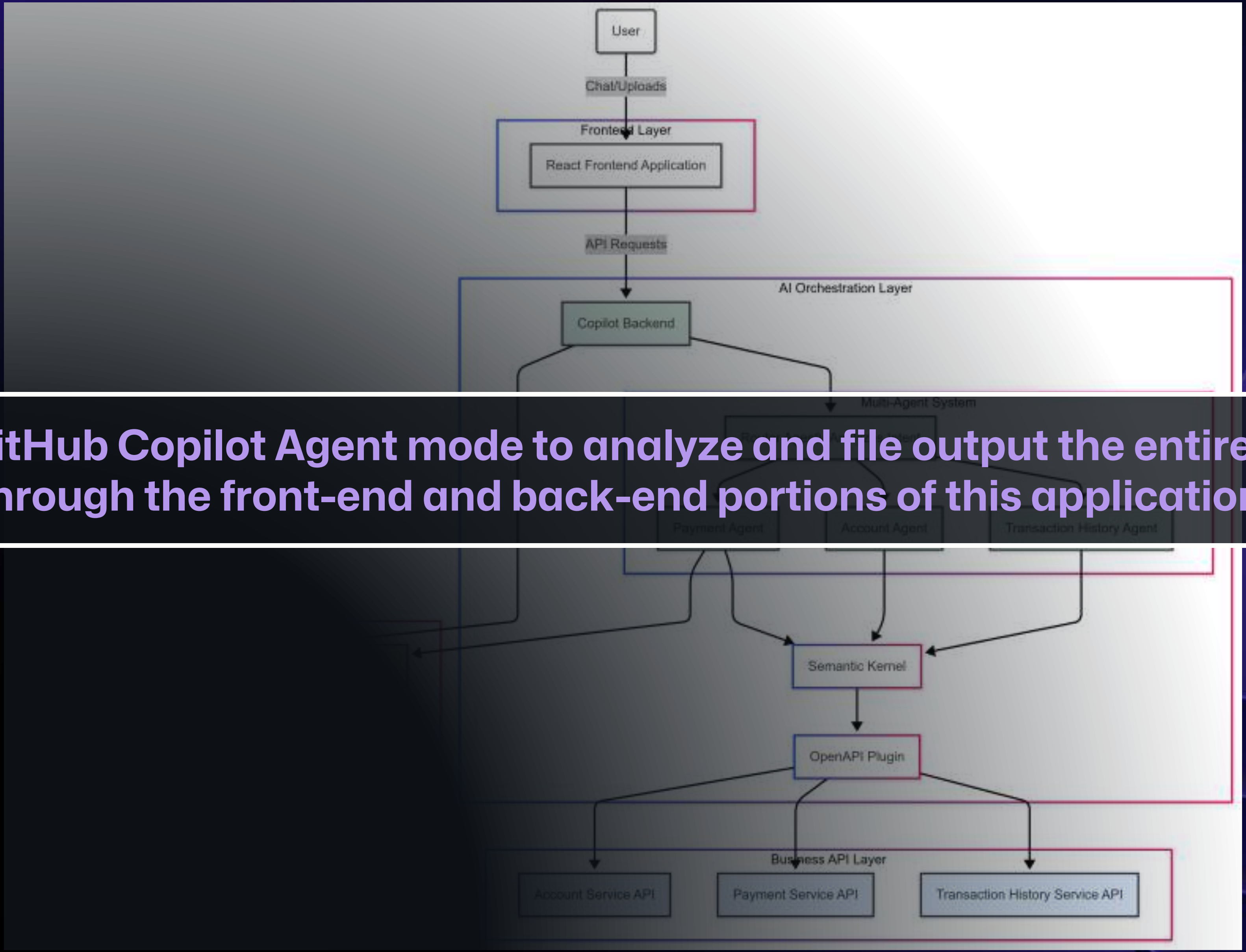
Application folder

The application follows a microservices architecture organized as follows:

```
app/
├── compose.yaml              # Docker Compose configuration for local deployment
├── start-compose.ps1         # PowerShell script to start services using Docker Compose
├── start-compose.sh          # Bash script to start services using Docker Compose
├── business-api/             # Backend business microservices
│   ├── account/              # Account management service
│   │   ├── Dockerfile        # Container definition for account service
│   │   ├── pom.xml           # Maven build configuration
│   │   └── src/              # Source code for account service
│   ├── payment/              # Payment processing service
│   │   ├── Dockerfile        # Container definition for payment service
                              # Maven build configuration
                              # Source code for transaction history service
                              # Copilot service - AI orchestration layer
                              # Container definition for copilot service
                              # Maven parent build configuration
                              # Main copilot service implementation
                              # Maven build configuration
                              # Kubernetes manifests for deployment
                               ource code for copilot backend
                                   # Semantic Kernel OpenAPI plugin
                                build configuration
                                 source code
                                 nterface application
                                   definition for production
                                   definition for AKS deployment
                                  ndencies and scripts
                                    configuration
                                    configuration
                                    manifests for deployment
```

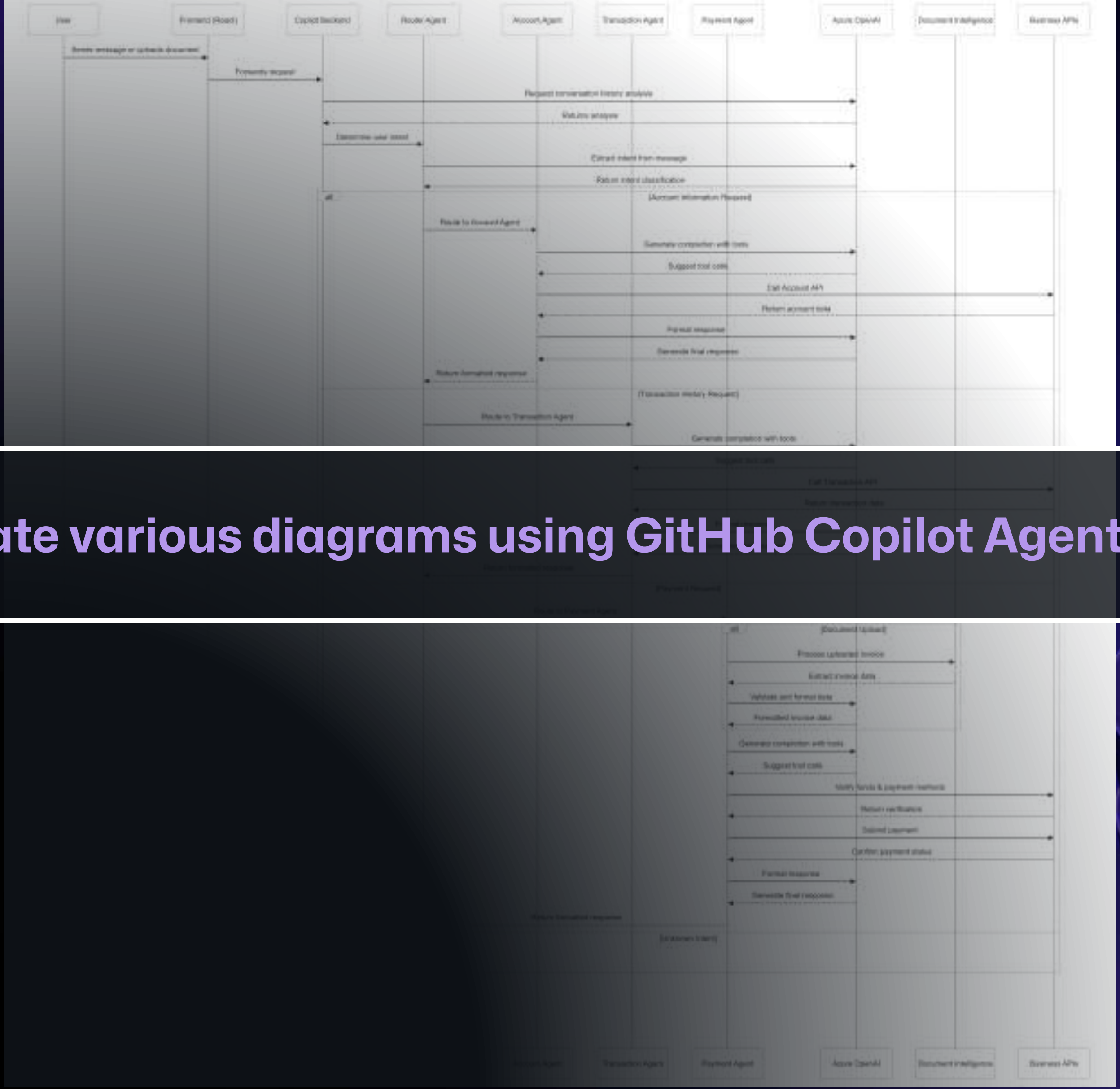
Try using Copilot Chat or Edits by drag & drop such specific items into a Working Set.

4. Create a flowchart of the application



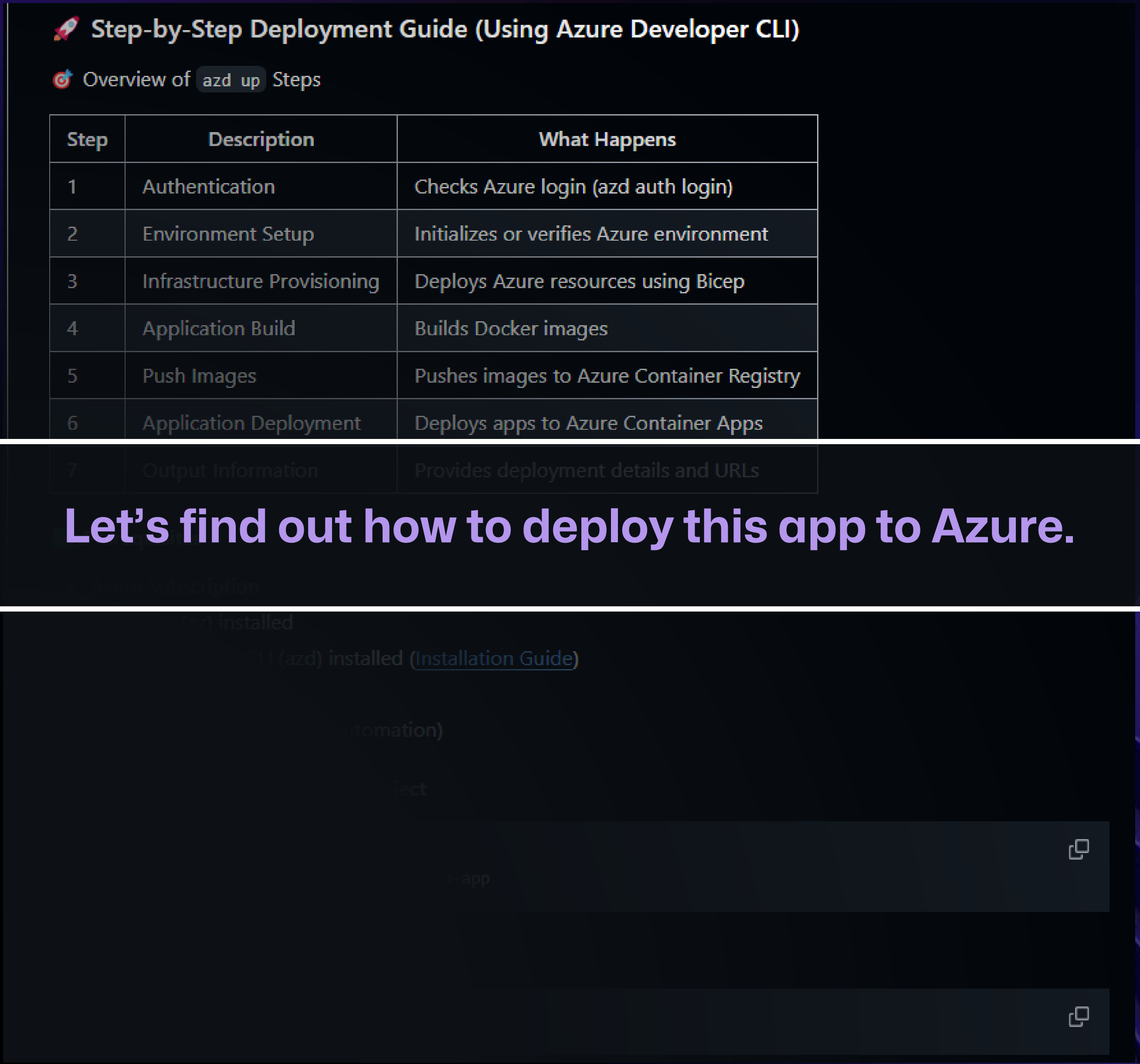
Use the GitHub Copilot Agent mode to analyze and file output the entire flowchart through the front-end and back-end portions of this application.

5. Create various diagrams



Generate various diagrams using GitHub Copilot Agent mode.

6. Discover how to deploy to Azure



🚀 Step-by-Step Deployment Guide (Using Azure Developer CLI)

🎯 Overview of `azd up` Steps

Step	Description	What Happens
1	Authentication	Checks Azure login (azd auth login)
2	Environment Setup	Initializes or verifies Azure environment
3	Infrastructure Provisioning	Deploys Azure resources using Bicep
4	Application Build	Builds Docker images
5	Push Images	Pushes images to Azure Container Registry
6	Application Deployment	Deploys apps to Azure Container Apps
7	Output Information	Provides deployment details and URLs

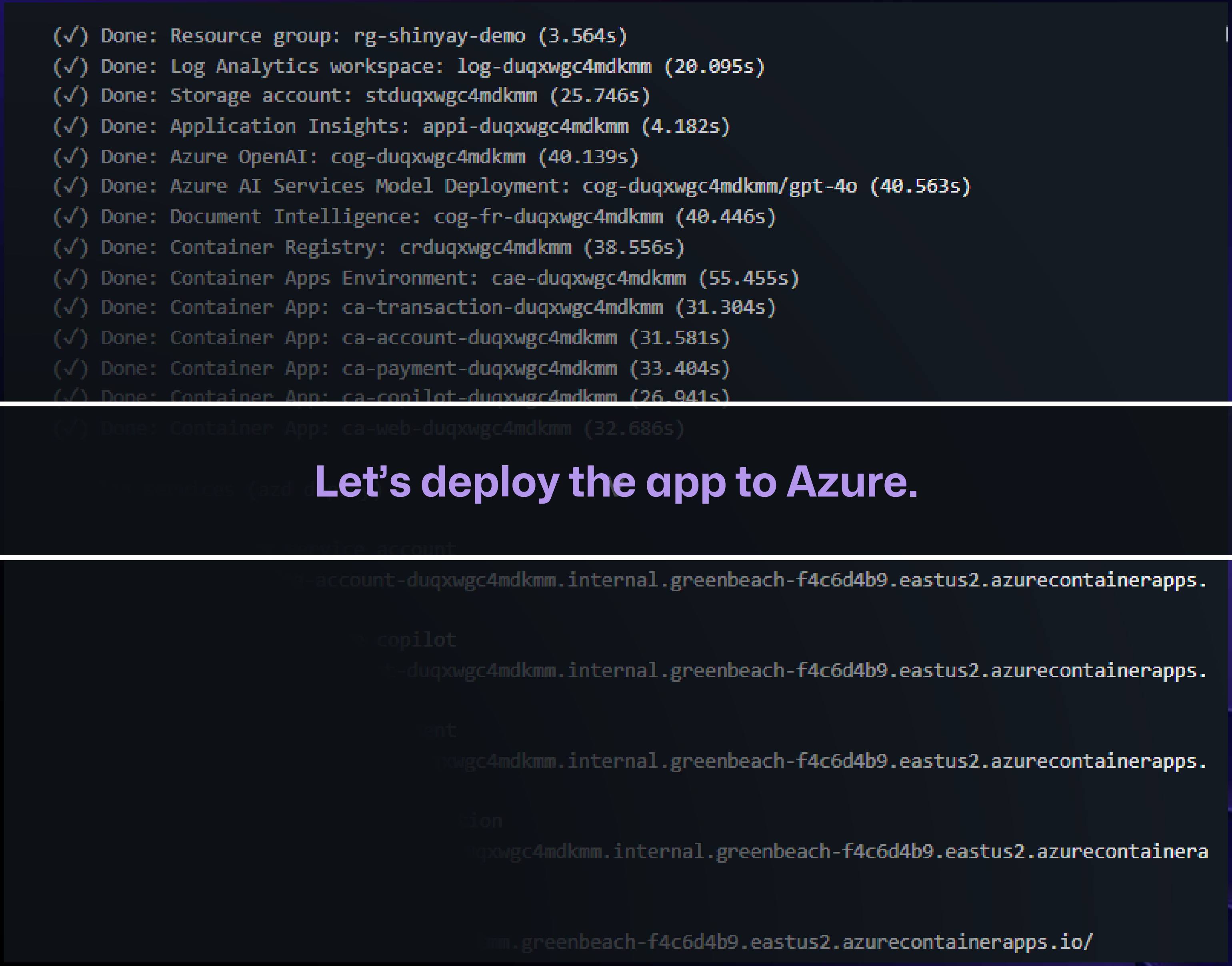
Azure Subscription

installed

(azd) installed (Installation Guide)

Let's find out how to deploy this app to Azure.

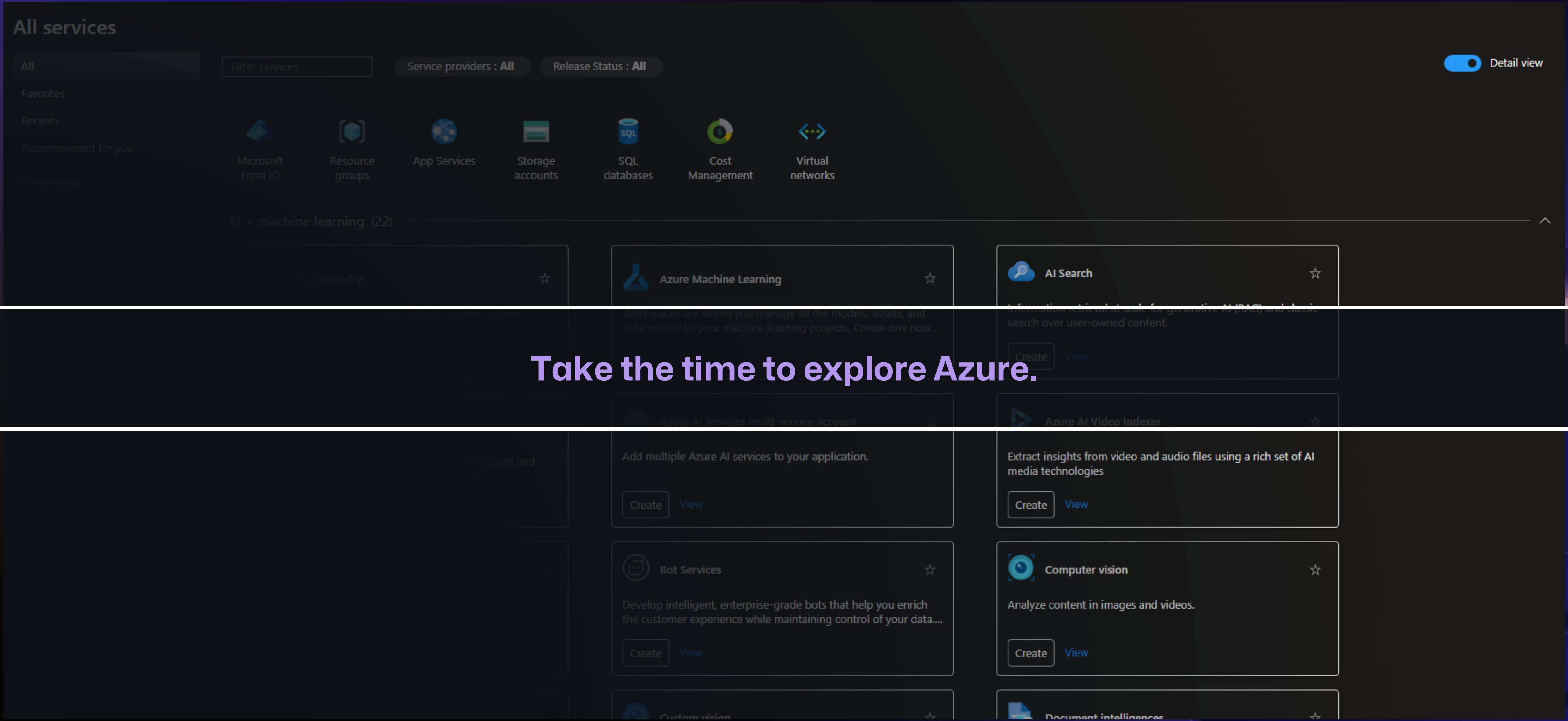
7. Deploy To Azure



```
(✓) Done: Resource group: rg-shinyay-demo (3.564s)
(✓) Done: Log Analytics workspace: log-duqxwgc4mdkmm (20.095s)
(✓) Done: Storage account: stduqxwgc4mdkmm (25.746s)
(✓) Done: Application Insights: appi-duqxwgc4mdkmm (4.182s)
(✓) Done: Azure OpenAI: cog-duqxwgc4mdkmm (40.139s)
(✓) Done: Azure AI Services Model Deployment: cog-duqxwgc4mdkmm/gpt-4o (40.563s)
(✓) Done: Document Intelligence: cog-fr-duqxwgc4mdkmm (40.446s)
(✓) Done: Container Registry: crduqxwgc4mdkmm (38.556s)
(✓) Done: Container Apps Environment: cae-duqxwgc4mdkmm (55.455s)
(✓) Done: Container App: ca-transaction-duqxwgc4mdkmm (31.304s)
(✓) Done: Container App: ca-account-duqxwgc4mdkmm (31.581s)
(✓) Done: Container App: ca-payment-duqxwgc4mdkmm (33.404s)
(✓) Done: Container App: ca-copilot-duqxwgc4mdkmm (26.941s)
(✓) Done: Container App: ca-web-duqxwgc4mdkmm (32.686s)

                                  ervice account
            -account-duqxwgc4mdkmm.internal.greenbeach-f4c6d4b9.eastus2.azurecontainerapps.

                                  copilot
                 -duqxwgc4mdkmm.internal.greenbeach-f4c6d4b9.eastus2.azurecontainerapps.

                                     nt
                   wgc4mdkmm.internal.greenbeach-f4c6d4b9.eastus2.azurecontainerapps.

                                        ion
                      qxwgc4mdkmm.internal.greenbeach-f4c6d4b9.eastus2.azurecontainera


                           m.greenbeach-f4c6d4b9.eastus2.azurecontainerapps.io/
```

Let’s deploy the app to Azure.

8. Experience Azure



Workspaces are where you manage all the models, assets, and data related to your machine learning projects. Create one now...

search over user-owned content.

Add multiple Azure AI services to your application.

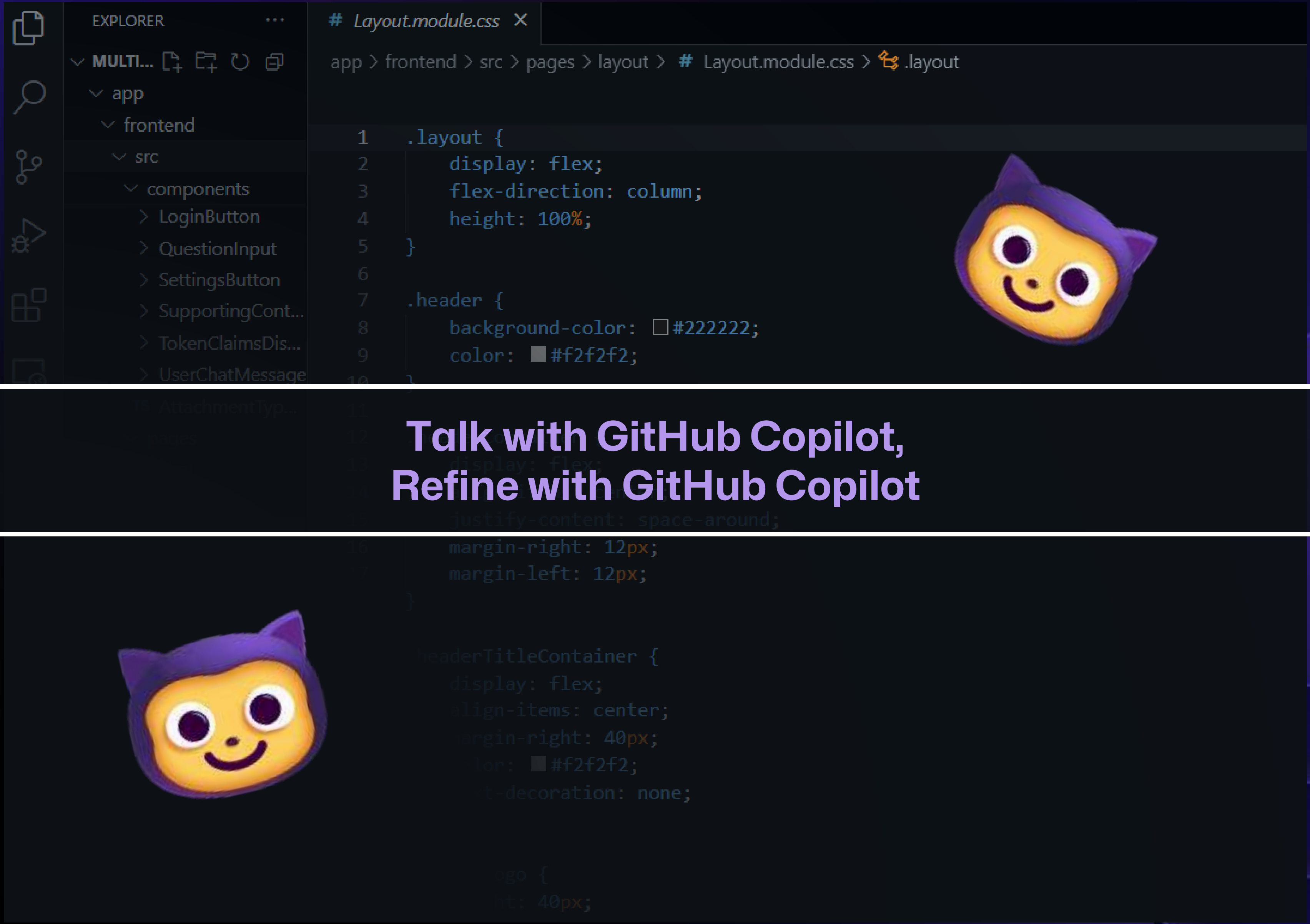
Extract insights from video and audio files using a rich set of AI media technologies

Develop intelligent, enterprise-grade bots that help you enrich the customer experience while maintaining control of your data....

Analyze content in images and videos.

Take the time to explore Azure.

9. Customize your Application with GitHub



```css
.layout {
    display: flex;
    flex-direction: column;
    height: 100%;
}

.header {
    background-color: #222222;
    color: #f2f2f2;
}
```

Talk with GitHub Copilot,
Refine with GitHub Copilot