

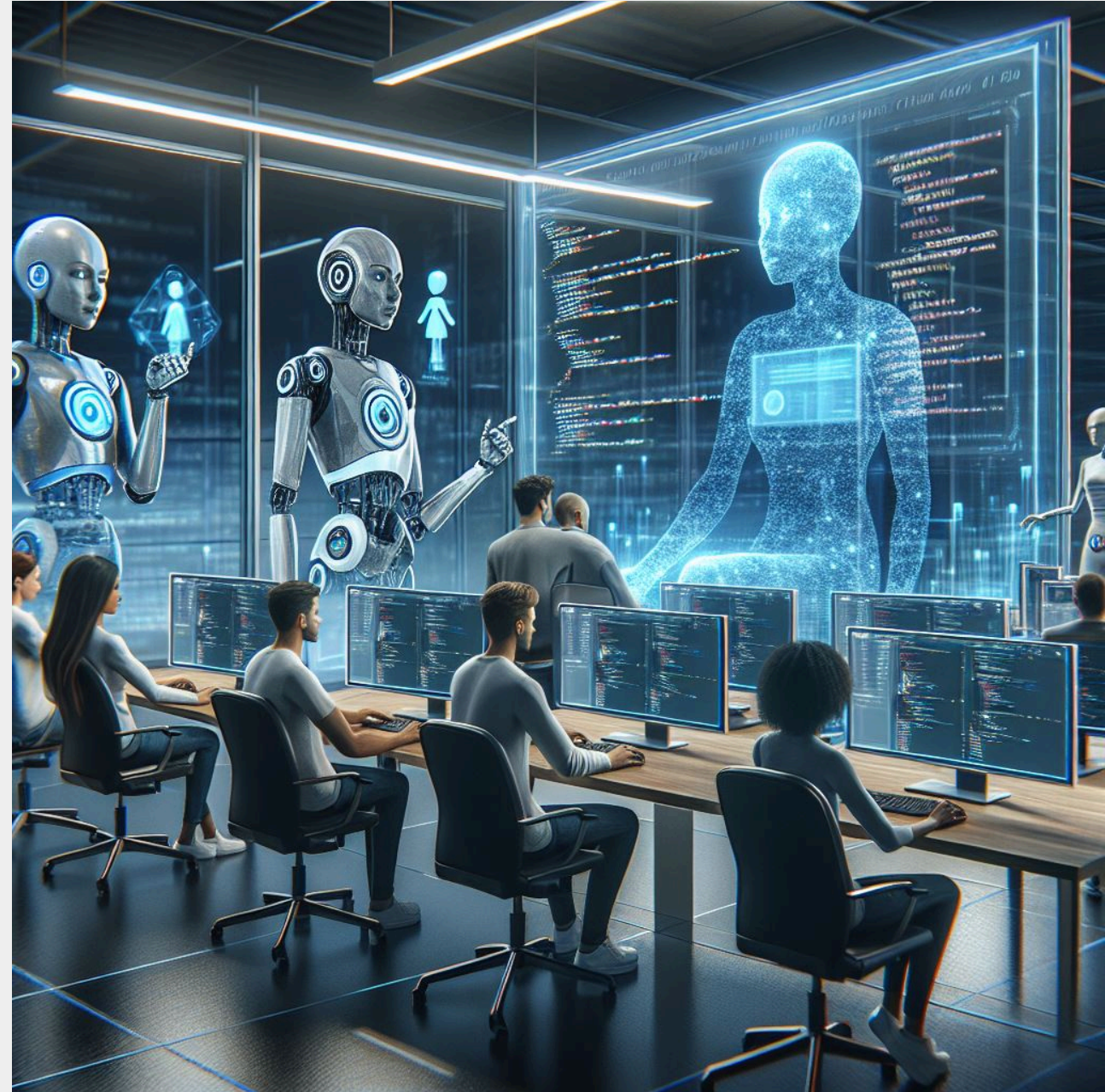
Beyond the IDE - Introducing AI Tools in our Development Flow

Beyond the IDE

Introducing AI Tools in our Development Flow

Filippos Karailanidis

@filkaris



Who are we?

- Leading fintech company
- Specializes in Currency trading
- Offices in Cyprus, Greece, London, Dubai, Japan, USA
- Supports 30 languages
- 10+ Million clients worldwide
- 1400+ employees / 600+ IT



What tools are we using?

What tools are we using?



Github Copilot

What tools are we using?



Github Copilot



JetBrains AI Assistant

What tools are we using?



GitHub Copilot

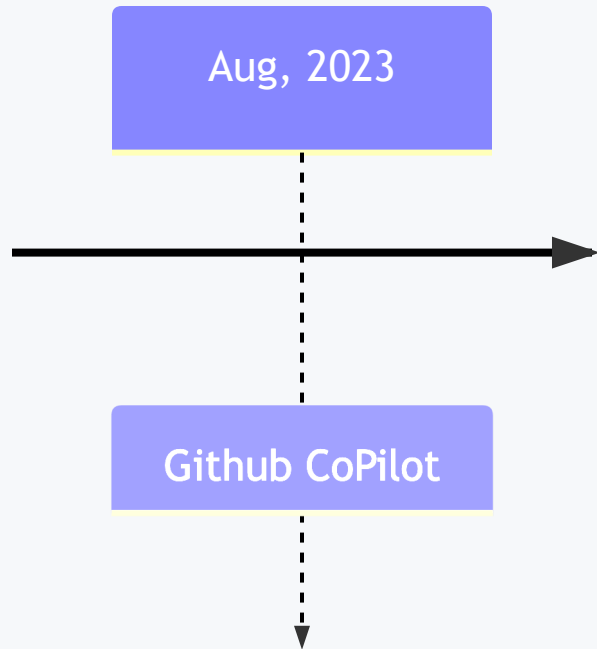


JetBrains AI Assistant

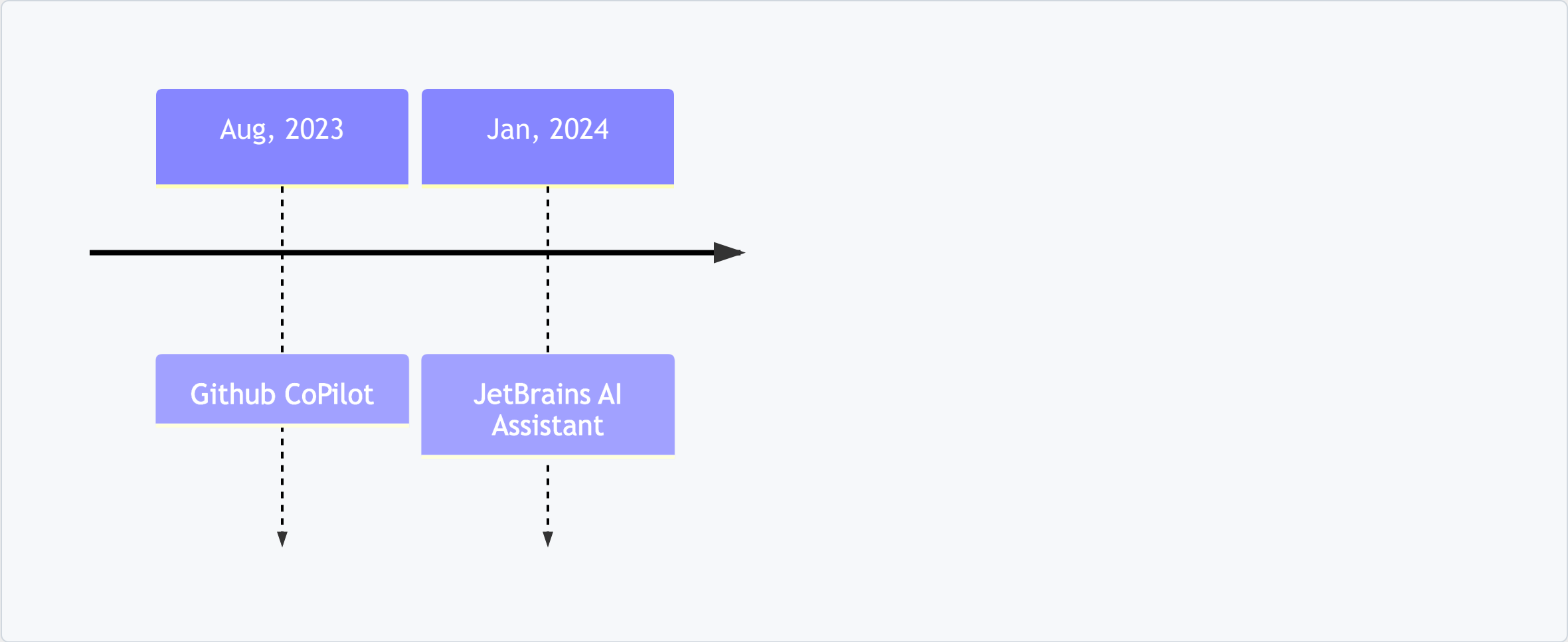


Amazon Q

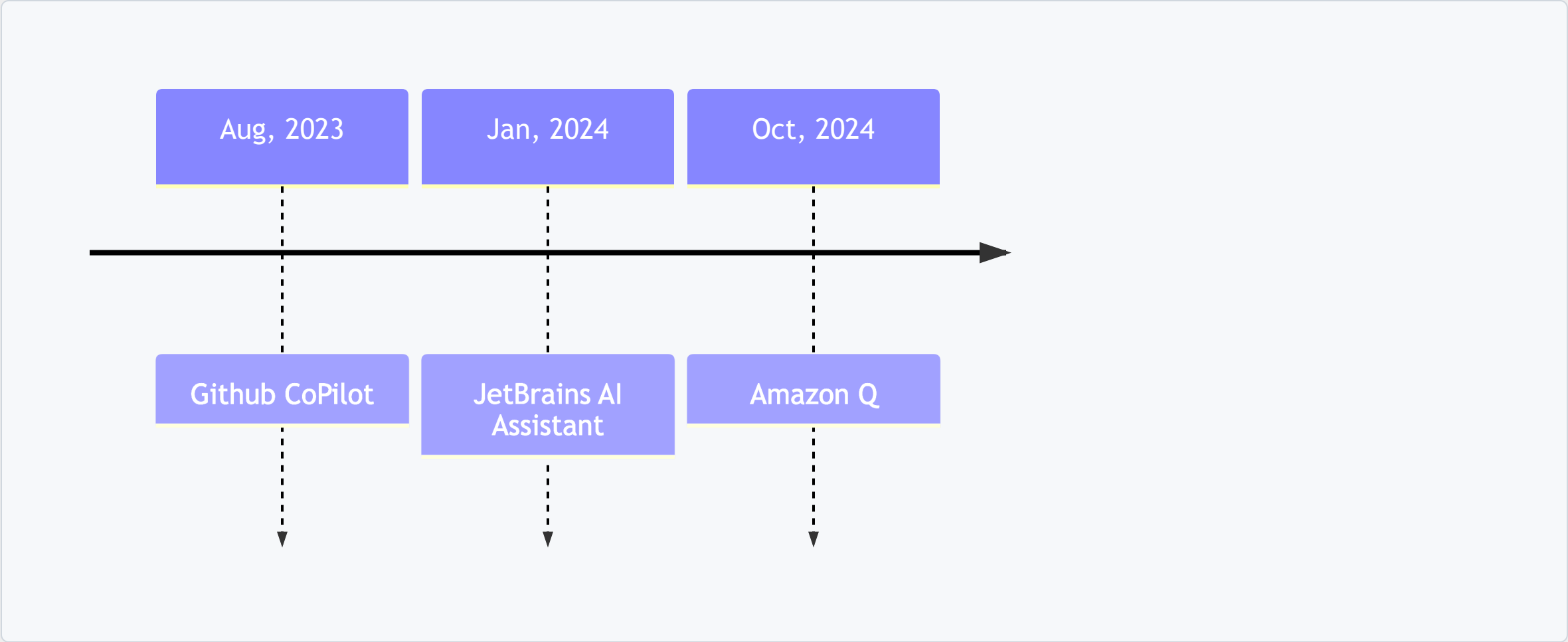
AI Tools Adoption



AI Tools Adoption

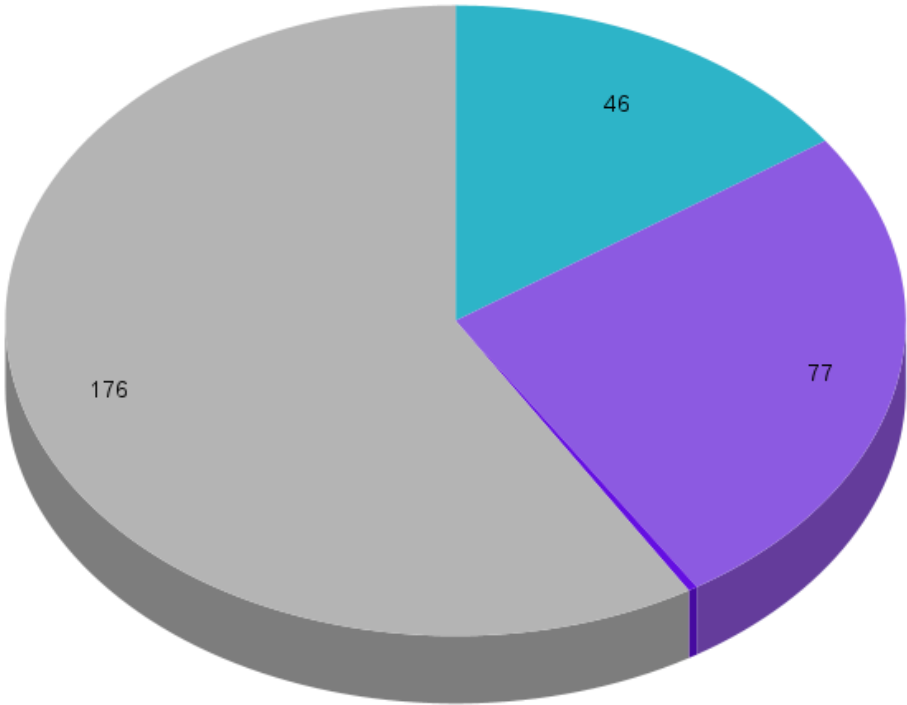


AI Tools Adoption



AI Tools Adoption

- Github Copilot
- JetBrains AI Assistant
- Amazon Q
- No Assistant



What these tools do

What these tools do - Autocomplete

What these tools do - Autocomplete

```
131
132  const getGithubUser = async () => {
133    const githubAuth = app.getPlugin(githubAuthApiRef);
      const githubIdentity = await githubAuth.getIdentity();
      return githubIdentity;
    }
134  ✨ ✨
135
136
```

What these tools do - Chat

What these tools do - Chat



fkarailanidis-xm

Describe the SOLID principles in one line each







GitHub Copilot

> Used 1 reference


1. **Single Responsibility Principle (SRP):** A class should have only one reason to change, meaning it should have only one job or responsibility.
2. **Open/Closed Principle (OCP):** Software entities should be open for extension but




What are some common design patterns that adhere to the SOLID principles?

Ask Copilot or type / for command    

What these tools do - Chat with Context

What these tools do - Chat with Context

 **Workspace**




  

> Used 1 reference

In the provided TypeScript code snippet, a constant named `guestModeProviderIdentity` is defined. This constant is an array of type `IdentityProviders`, which likely represents a collection of identity provider configurations used for authentication purposes within an application.

The first element of the array is an object that

What is the purpose of the `IdentityProviders` type in the code snippet?

Ask Copilot or type / for commands   

```
7      id: 'microsoft-auth-provider',
6      title: 'Azure AD',
5      message: 'Sign in using Microsoft AD',
4      apiRef: microsoftAuthApiRef,
3    },
2  ];
1  //
83 const guestModeProviderIdentity: IdentityProviders = [
1    {
2      id: 'github',
3      title: 'github-provider',
4      message: 'Sign in using Github',
5      apiRef: githubAuthApiRef,
6    },
7    'guest',
8  ];
9
10 const app = createApp({
11   apis,
12   themes: themes,
13   components: {
```

Use Cases

- Code Generation
- Code Generation with Context
- Code Review
- Bigger Picture

Code Generation - Generate Code

Code Generation - Generate Code

```
131
132  const getGithubUser = async () => {
133    const githubAuth = app.getPlugin(githubAuthApiRef);
      const githubIdentity = await githubAuth.getIdentity();
      return githubIdentity;
    }
134  ✨
135
136
```

Code Generation - Generate sample datasets

Code Generation - Generate sample datasets

Generate a js array of 10 sample users with fields name, email, age, phone, address. Use realistic names

Here's a JavaScript array with 10 sample users using realistic data: [1]

```
const users = [  
  {  
    name: "Sarah Johnson",  
    email: "sarah.johnson@email.com",  
    age: 34,  
    phone: "(555) 123-4567",  
    address: "742 Maple Street, Boston, MA 02108"  
  },  
  {  
    name: "Michael Chen",  
    email: "mchen@email.com",  
    age: 29,  
    phone: "(555) 234-5678",  
    address: "1234 Oak Avenue, San Francisco, CA 94110"  
  },  
  {
```

Code Generation - Generate pipelines / infrastructure

Code Generation - Generate pipelines / infrastructure

```
🦊 .gitlab-ci.yml
315   .restart:
321
322
323   deploy:
324     stage: deploy
        image: $IMAGE_URL
        script:
          - *common
          - *environment
          - *helm_deploy
325   ✨
```


Generation with Context - Unit Tests

Generation with Context - Unit Tests

write a unit test for this piece of code

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Generation with Context - Unit Tests

write a unit test for this piece of code

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's a unit test suite for the azureUserTranformer function using Jest: [1]

```
import { UserEntity } from '@backstage/catalog-model';  
  
describe('azureUserTranformer', () => {  
  // Mock the defaultUserTransformer function  
  const defaultUserTransformer = jest.fn();
```

Generation with Context - Translate to another Language

Generation with Context - Translate to another Language

Convert this code to Python

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Generation with Context - Translate to another Language

Convert this code to Python

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's the Python equivalent of the TypeScript code, using Python's type hints and async/await syntax:

```
from typing import Optional, Union  
from dataclasses import dataclass  
  
# Assuming these types exist in your Python codebase  
@dataclass  
class UserMetadata:  
    name: str  
    description: str
```

Generation with Context - Write Documentation

Generation with Context - Write Documentation

Write some documentation using markdown

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```


Generation with Context - Write Documentation

Write some documentation using markdown

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Azure User Transformer

A utility function that transforms an Azure Active Directory user into a Backstage user entity.

Overview

The `azureUserTranformer` function takes a user object from Azure Graph API and optionally a user phot

Beyond the IDE - Introducing AI Tools in our Development Flow

Generation with Context - Debug

Generation with Context - Debug

Help me debug this function

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Generation with Context - Debug

Help me debug this function

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's a comprehensive debugging guide for the `azureUserTranformer` function:

1. First, let's add logging to track the function's execution:

Generation with Context - Debug

Help me debug this function

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's a comprehensive debugging guide for the `azureUserTranformer` function:

1. First, let's add logging to track the function's execution:

4. Debugging checklist:

- Verify that `graphUser` contains expected data
- Confirm `defaultUserTransformer` is working correctly
- Check email extraction logic

Code Review - Quality

Code Review - Quality

Review this piece of code and provide suggestions if needed to increase code quality

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Code Review - Quality

Review this piece of code and provide suggestions if needed to increase code quality

```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's a review of the code with suggestions for improvement:

1. Input Validation:

```
async function azureUserTranformer(  
  graphUser: User,
```


Code Review - Quality

Review this piece of code and provide suggestions if needed to increase code quality


```
async function azureUserTranformer(  
  graphUser: User,  
  userPhoto?: string,  
): Promise<UserEntity | undefined> {
```

Here's a review of the code with suggestions for improvement:

1. Input Validation:

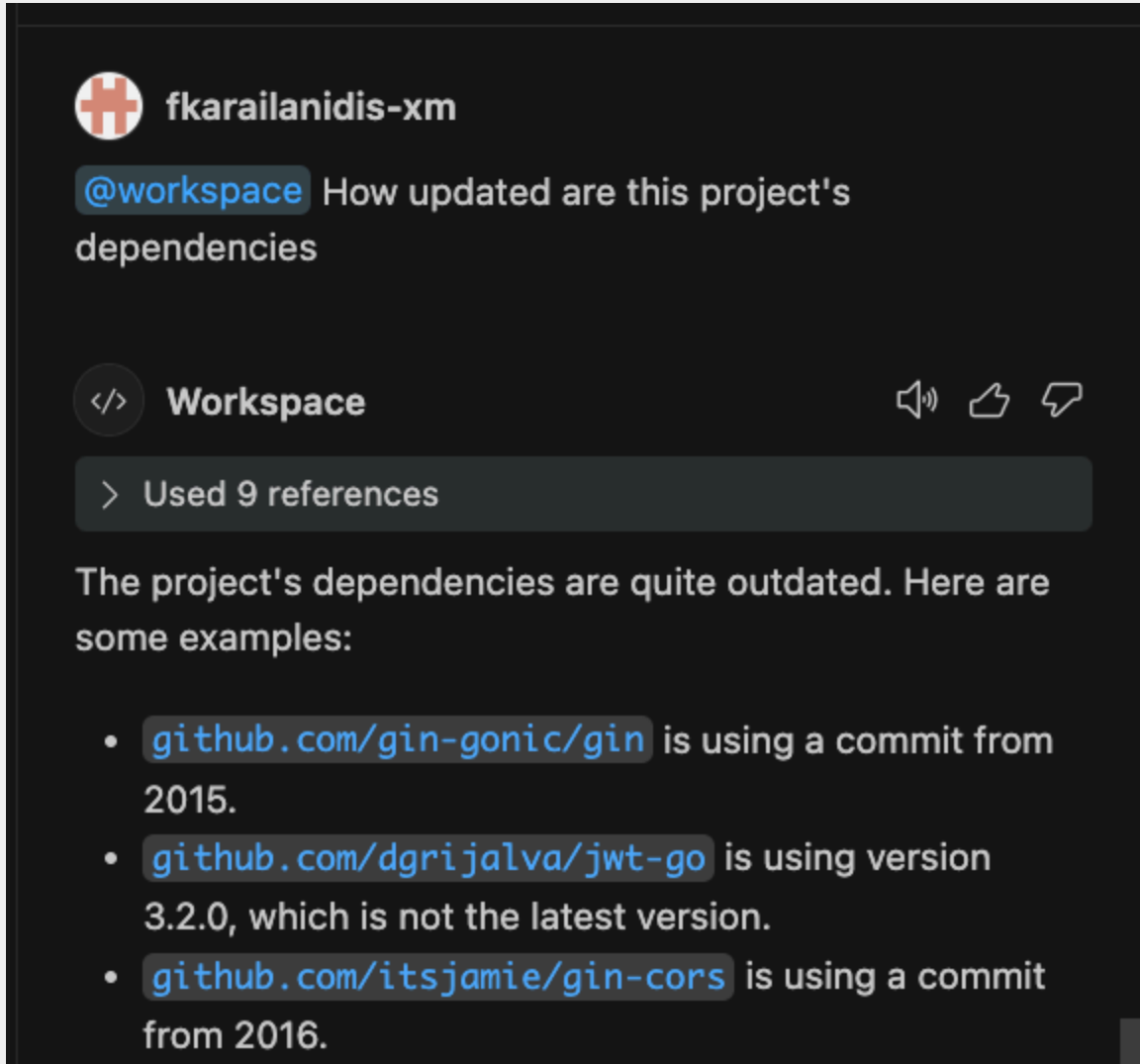
```
async function azureUserTranformer(  
  graphUser: User,
```

typescript

 View Diff  Apply Diff  Copy

Bigger Picture - Dependencies




Bigger Picture - Dependencies



A screenshot of a GitHub comment interface. At the top, the user profile for 'fkarailanidis-xm' is shown with a red cross icon. The comment text asks how updated the project's dependencies are. Below the text is a 'Workspace' label with a code icon, and icons for speaking, liking, and commenting. A dropdown menu is open, showing 'Used 9 references'. The main text of the comment states that the dependencies are outdated and provides three examples of outdated dependencies.

fkarailanidis-xm

@workspace How updated are this project's dependencies

Workspace   

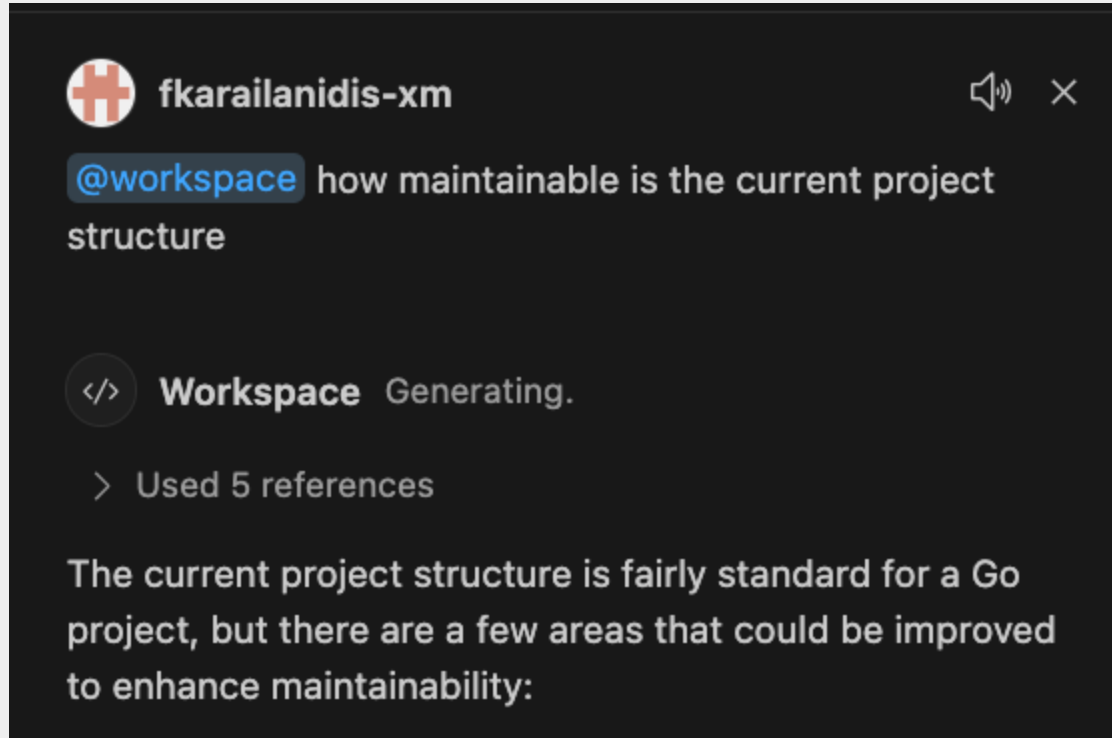
> Used 9 references

The project's dependencies are quite outdated. Here are some examples:

- github.com/gin-gonic/gin is using a commit from 2015.
- github.com/dgrijalva/jwt-go is using version 3.2.0, which is not the latest version.
- github.com/itsjamie/gin-cors is using a commit from 2016.

Bigger Picture - Rate Project Structure

Bigger Picture - Rate Project Structure



Beyond the IDE - Introducing AI Tools in our Development Flow

Bigger Picture - Upgrade Java Version

Bigger Picture - Upgrade Java Version

Welcome to code transformation!

I can help you upgrade your Java 8 and 11 codebases to Java 17.

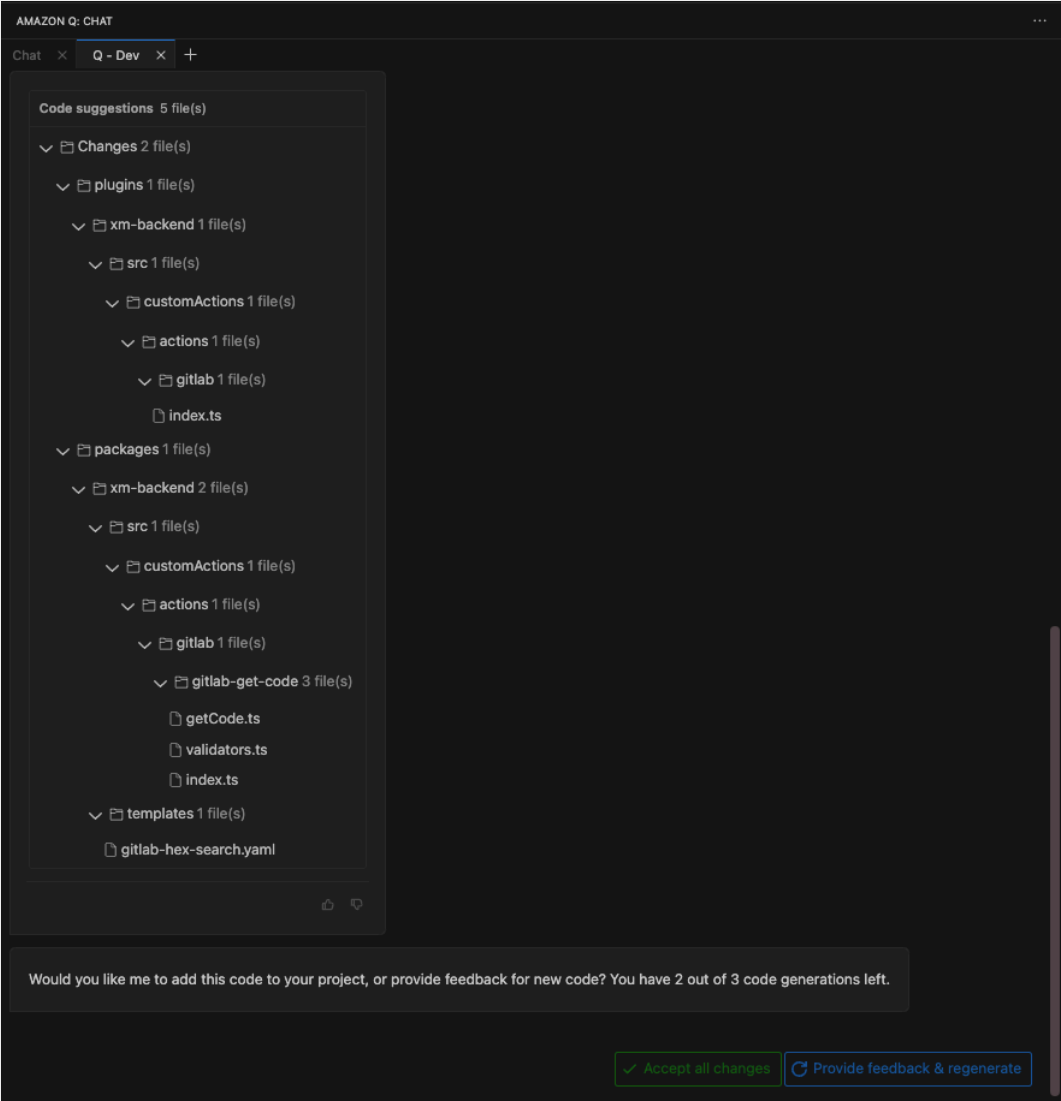
I am checking for open projects that are eligible for Code Transformation.

Sorry, I couldn't find a project that I can upgrade. Currently, I can only upgrade Java 8 or Java 11 projects built on Maven. For more information, see the [Amazon Q documentation](#).

Start a new transformation

Bigger Picture - Add a Feature

Bigger Picture - Add a Feature



Beyond the IDE - Introducing AI Tools in our Development Flow

AI Tool Awards

AI Tool Awards



Github Copilot



JetBrains AI Assistant



Amazon Q

AI Tool Awards



GitHub Copilot



JetBrains AI Assistant



Amazon Q



Best Autocomplete

AI Tool Awards



GitHub Copilot



JetBrains AI Assistant



Amazon Q



Best Autocomplete



Best IDE Integration

AI Tool Awards



GitHub Copilot



Best Autocomplete



JetBrains AI Assistant



Best IDE Integration



Amazon Q



Best Context Awareness

Our Experience

- Positive in general
- Careful with hallucinations
- Not everyone embraces it
- Used mostly by Senior engineers
- Very fast iterations by the vendors

Thank you!

