Den norske dataforening

Testkonferansen

ODIN 2024

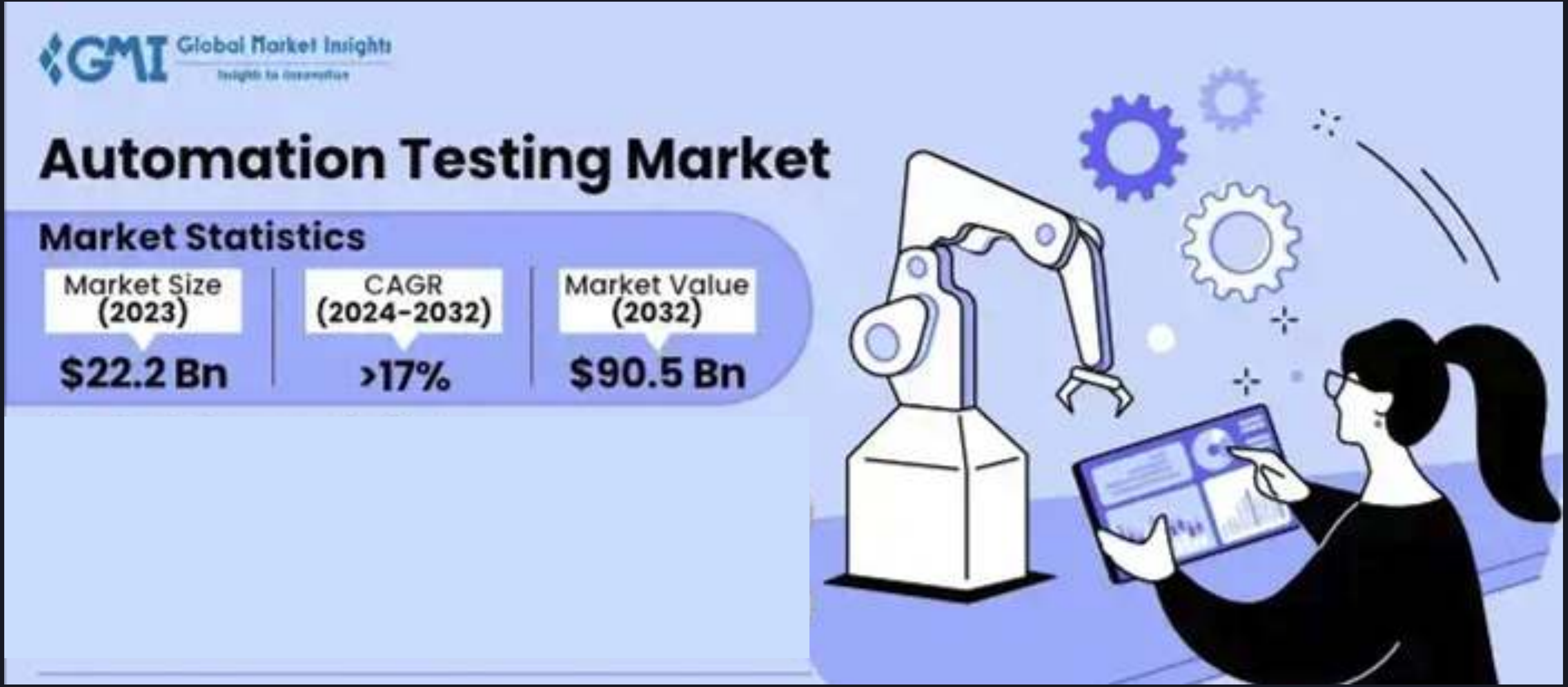
Chao Tan

Senior test engineer at Testify AS

How AI Might Relieve Your Test Automation Pains

How AI Could Help to Relieve Your Test Automation Pain

Steady growth forecast of the Test Automation Market



Automation Testing Market

Market Statistics

Market Size (2023)	CAGR (2024-2032)	Market Value (2032)
$22.2 Bn	>17%	$90.5 Bn

GMI Global Market Insights
Insights to Innovation

Test Automation Pain Points

Independent studies have consistently shown that **65% to 70%** of testing time is spent on maintaining existing tests that have failed in subsequent software releases.

Test Authoring Complexity

Creating automated tests can be challenging and time-consuming.

Test Maintenance Overhead

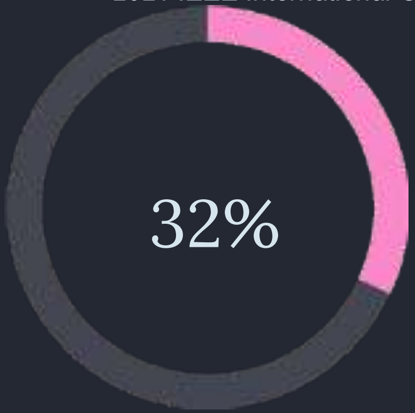
Keeping tests up-to-date with changing software requires significant effort.

Unstable Test Execution

Inconsistent test results lead to frustration and wasted time.

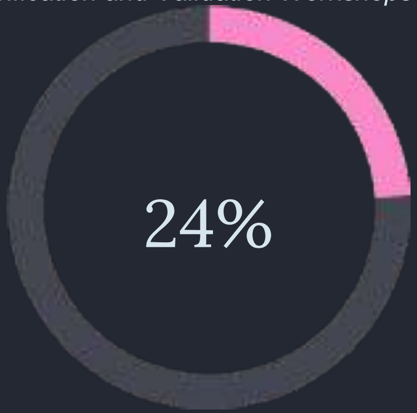
Literature reviews: Test Automation Problems

Ricca, Filippo, Alessandro Marchetto, and Andrea Stocco. "**Ai-based test automation: A grey literature analysis.**" *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021.



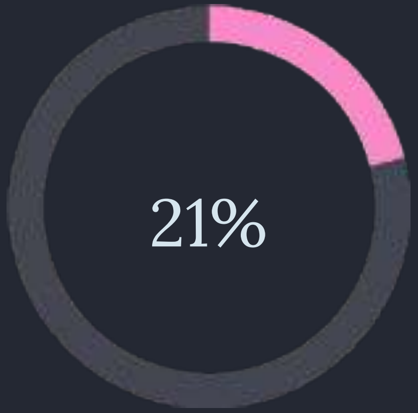
Test Authoring

Manual code development, manual data creation, test object identification, cross-platform testing



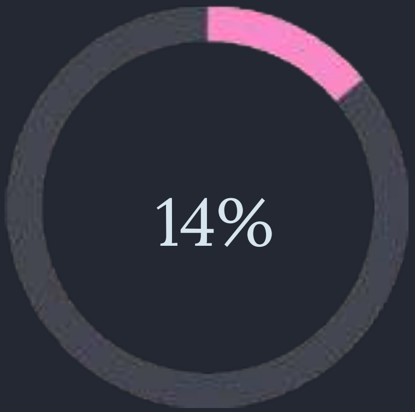
Test Maintenance

Manual test code update, manual test migration, fragile test script, costly GUI visual regression,



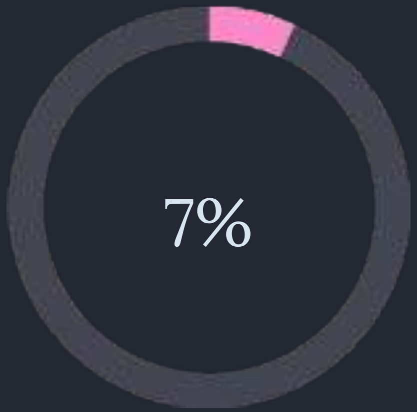
Test Execution

Insufficient coverage, flakiness, slow execution, useless re-test



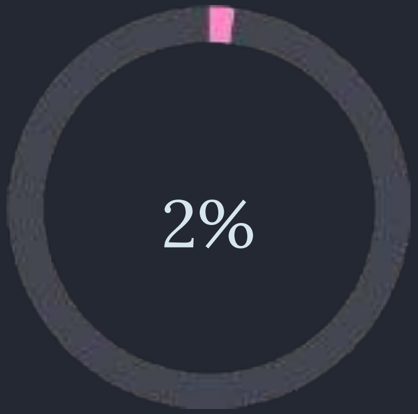
Test Closure

Manual debugging overhead, costly result inspection, visual analysis



Test Planning

Critical paths identification, test selection and prioritization, planning long release cycles



Test Design

Programming skills required, Domain knowledge required

Impact of Pain Points on Teams

Escalating time and cost

Eroded confidence in Quality

Diminished morale and productivity

AI and testing

AI Advancement

Significant advancement in NLP, computer vision and ML leads to sophisticated application across various industries.

Relevance to Testing

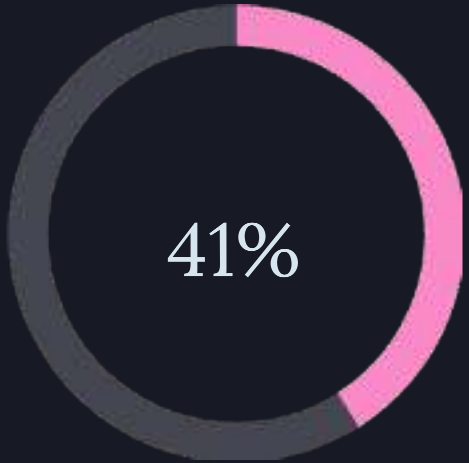
AI can automate complex tasks, analyze patterns, and make intelligent decisions in testing processes.

Integration

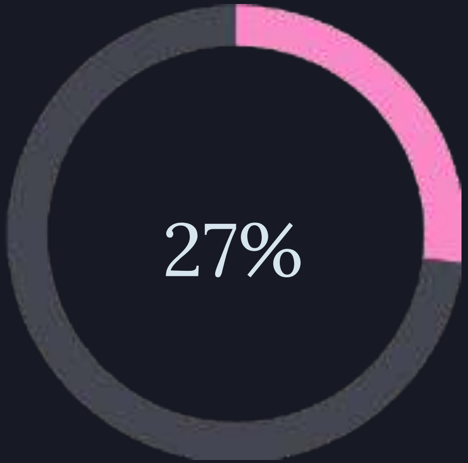
AI is being integrated into various testing tools and platforms to enhance capabilities.

Literature review: AI-based Solutions

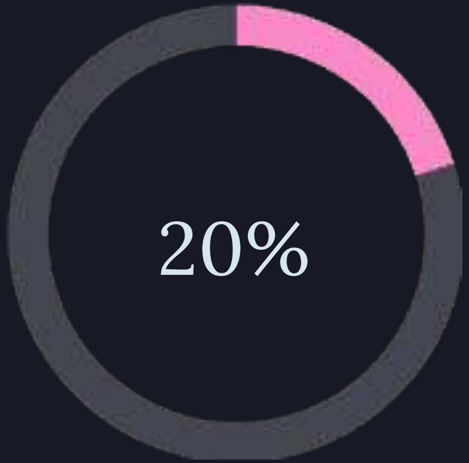
Ricca, Filippo, Alessandro Marchetto, and Andrea Stocco. "**Ai-based test automation: A grey literature analysis.**" *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021.



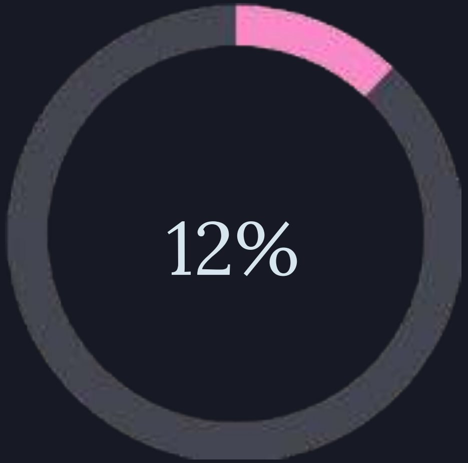
Test Generation



Test Maintenance



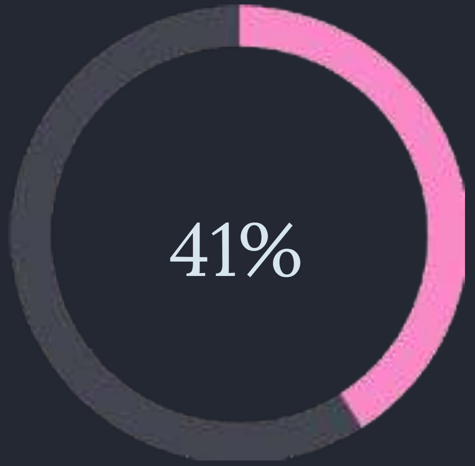
Debugging



Oracle

Testify
smartere testing

AI-powered test generation



Test Generation

1 Automated test code generation

Example: Utilize NLP to drive the creation of tests.

Tool example: Testsigma

2 Automated data generation

Example: Using AI to create synthetic data that mimics real-world data.

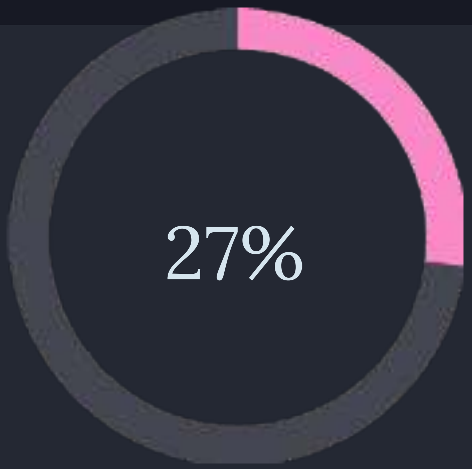
Project example: Synthetic Test Data for Norwegian Population Registry

3 Automation of UI test generation

Example technologies: Robust element localization, dynamic properties recognition, object recognition engine

Testify smartere testing

Maintenance and execution



Test Maintenance

1. Self-healing mechanism:
   self-healing scripts and smart locators

   Tool example: Testim, Testsigma, Mabl

2. Intelligent fault prediction

   Example: predictive test selection based on fault prediction at Facebook

3. Intelligent test case prioritization and adaptive tests

   Example: Reinforcement learning for test case prioritization in CI at Netflix

Debugging

1. Intelligent test analysis
2. Automated coverage report
3. Noticeable code changes identification
4. Flaky test identification
5. Tool examples

   Testim, Mabl, Rainforest QA, Codacy

Debugging

Testify
smartere testing

Oracle

12%

Oracle

1. Optical character recognition(OCR)

   Test the visual correctness of GUI using OCR(Optical character recognition) or image-recognition techniques.

2. Automatic visual discrepancy detection

   Compares the current visual state to correspondent ground truth of the page.

   Tool example: Applitools

3. Deep learning classifier

   Train deep learning classifier to detect visual imperfections, such as images partially occluded by other image or text

   Example: GUI testing at eBay

Recommendations for Getting Started

1 Start Small

Begin with a pilot project to test AI-powered tools.

2 Choose the tools that suites your need

Ensure your team is well-versed in AI-based testing techniques.

3 Integrate Gradually

Slowly incorporate AI tools into existing workflows for smooth transition.

4 Monitor and Adjust

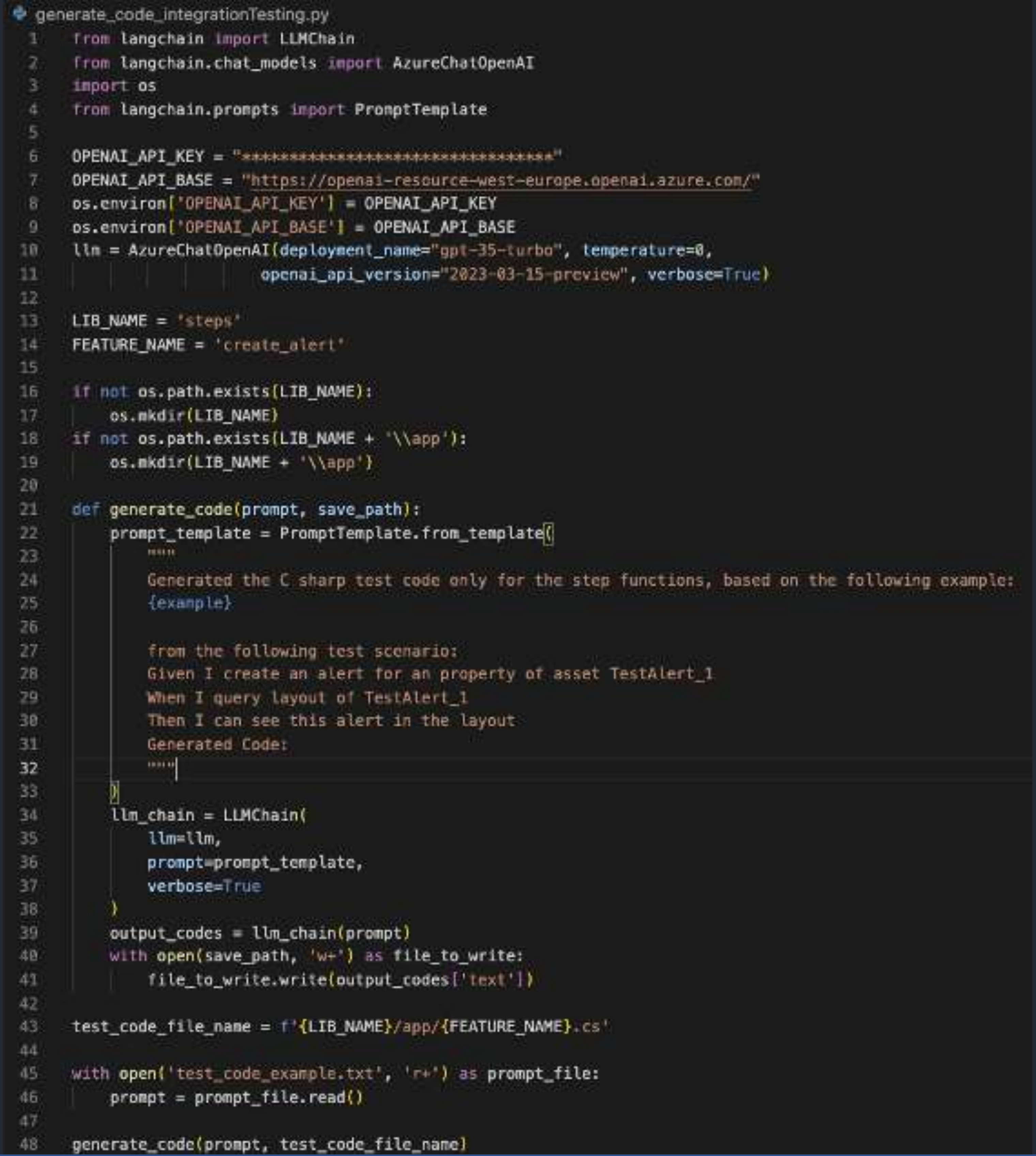
Continuously evaluate the impact of AI on your testing processes.

Testify smartere testing

Demonstration

Lightweight example of test code generation with OpenAI:

- Python script to interact with large language models
- Provide example code as prompt input
- Provide instruction for coding
- Generate code


generate_code_integrationTesting.py

```python
from langchain import LLMChain
from langchain.chat_models import AzureChatOpenAI
import os
from langchain.prompts import PromptTemplate

OPENAI_API_KEY = "**********************************"
OPENAI_API_BASE = "https://openai-resource-west-europe.openai.azure.com/"
os.environ['OPENAI_API_KEY'] = OPENAI_API_KEY
os.environ['OPENAI_API_BASE'] = OPENAI_API_BASE
llm = AzureChatOpenAI(deployment_name="gpt-35-turbo", temperature=0,
                      openai_api_version="2023-03-15-preview", verbose=True)

LIB_NAME = 'steps'
FEATURE_NAME = 'create_alert'

if not os.path.exists(LIB_NAME):
    os.mkdir(LIB_NAME)
if not os.path.exists(LIB_NAME + '\\app'):
    os.mkdir(LIB_NAME + '\\app')

def generate_code(prompt, save_path):
    prompt_template = PromptTemplate.from_template(
        """
        Generated the C sharp test code only for the step functions, based on the following example:
        {example}

        from the following test scenario:
        Given I create an alert for an property of asset TestAlert_1
        When I query layout of TestAlert_1
        Then I can see this alert in the layout
        Generated Code:
        """
    )
    llm_chain = LLMChain(
        llm=llm,
        prompt=prompt_template,
        verbose=True
    )
    output_codes = llm_chain(prompt)
    with open(save_path, 'w+') as file_to_write:
        file_to_write.write(output_codes['text'])

test_code_file_name = f'{LIB_NAME}/app/{FEATURE_NAME}.cs'

with open('test_code_example.txt', 'r+') as prompt_file:
    prompt = prompt_file.read()

generate_code(prompt, test_code_file_name)
```

1

Python code example

Utilize OpenAI gpt-35 model to generate specflow step implementation with Gherkin scenario as input promt

2

LangChain framework

A python framework designed to simplify the development of applications that leverage LLM models

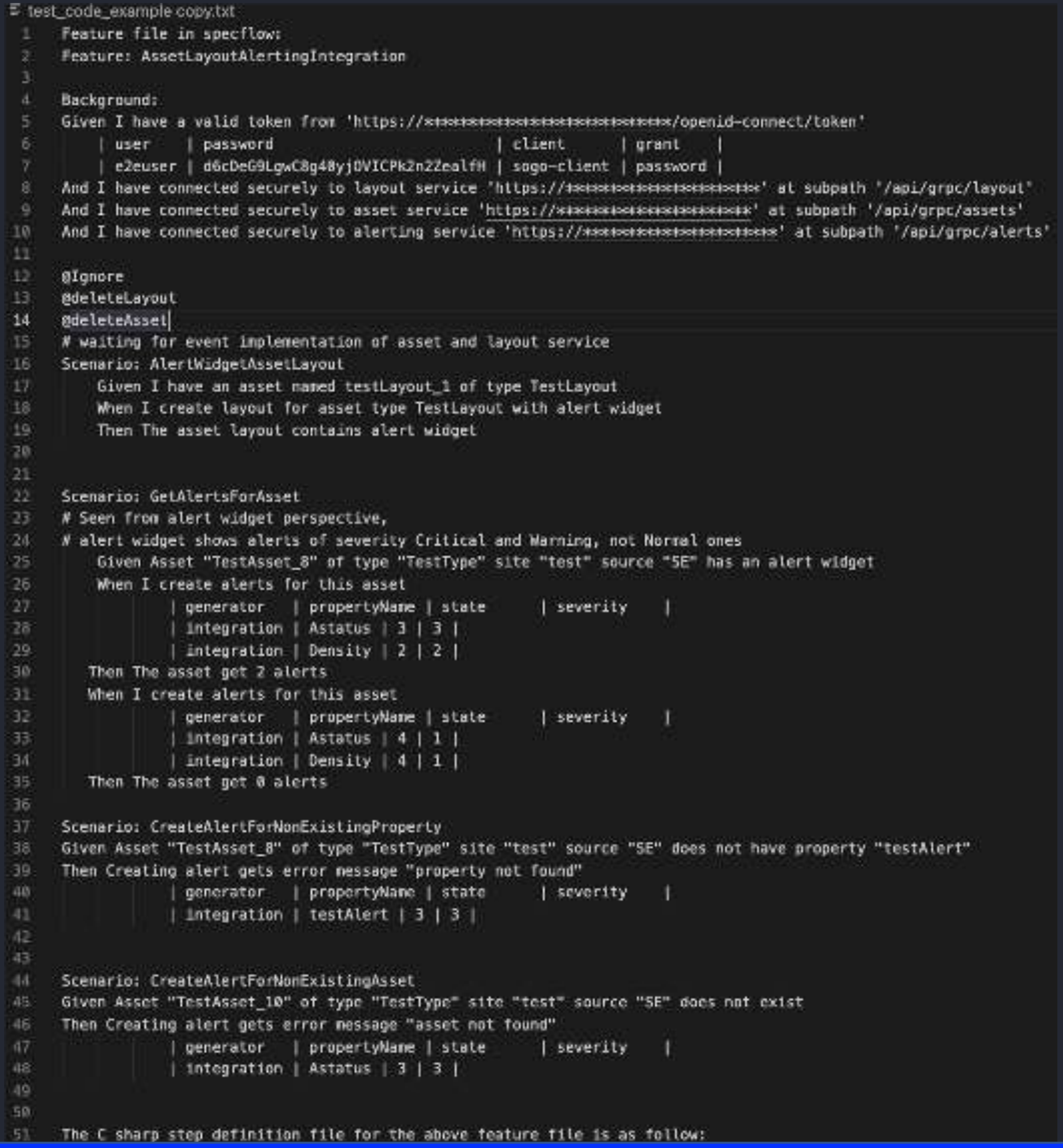
3

Logic steps

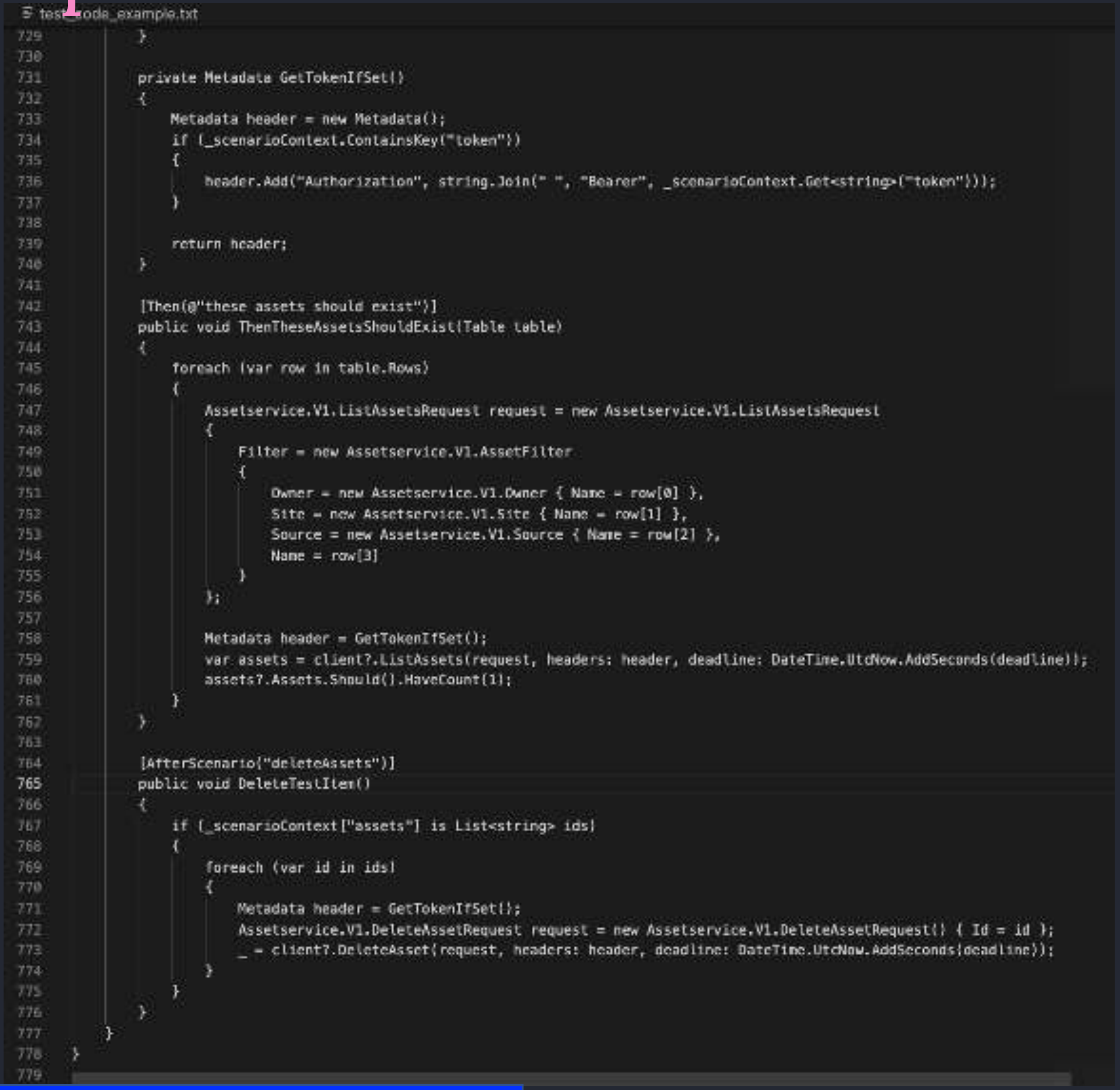
Provide examples to the model, input prompt and generate code.

https://gist.github.com/ChaoTanTestify/1140ad43af6952952660c19405cada98

Example file


test_code_example copy.txt

```
Feature file in specflow:
Feature: AssetLayoutAlertingIntegration

Background:
Given I have a valid token from 'https://****************************/openid-connect/token'
    | user    | password                         | client      | grant    |
    | e2euser | d6cDeG9LgwC8g48yjOVICPk2n2ZealfH | sogo-client | password |
And I have connected securely to layout service 'https://**********************' at subpath '/api/grpc/layout'
And I have connected securely to asset service 'https://**********************' at subpath '/api/grpc/assets'
And I have connected securely to alerting service 'https://**********************' at subpath '/api/grpc/alerts'

@Ignore
@deleteLayout
@deleteAsset
# waiting for event implementation of asset and layout service
Scenario: AlertWidgetAssetLayout
    Given I have an asset named testLayout_1 of type TestLayout
    When I create layout for asset type TestLayout with alert widget
    Then The asset layout contains alert widget


Scenario: GetAlertsForAsset
# Seen from alert widget perspective,
# alert widget shows alerts of severity Critical and Warning, not Normal ones
    Given Asset "TestAsset_8" of type "TestType" site "test" source "SE" has an alert widget
    When I create alerts for this asset
            | generator   | propertyName | state      | severity    |
            | integration | Astatus | 3 | 3 |
            | integration | Density | 2 | 2 |
   Then The asset get 2 alerts
   When I create alerts for this asset
            | generator   | propertyName | state      | severity    |
            | integration | Astatus | 4 | 1 |
            | integration | Density | 4 | 1 |
   Then The asset get 0 alerts

Scenario: CreateAlertForNonExistingProperty
Given Asset "TestAsset_8" of type "TestType" site "test" source "SE" does not have property "testAlert"
Then Creating alert gets error message "property not found"
            | generator   | propertyName | state      | severity    |
            | integration | testAlert | 3 | 3 |


Scenario: CreateAlertForNonExistingAsset
Given Asset "TestAsset_10" of type "TestType" site "test" source "SE" does not exist
Then Creating alert gets error message "asset not found"
            | generator   | propertyName | state      | severity    |
            | integration | Astatus | 3 | 3 |


The C sharp step definition file for the above feature file is as follow:
```

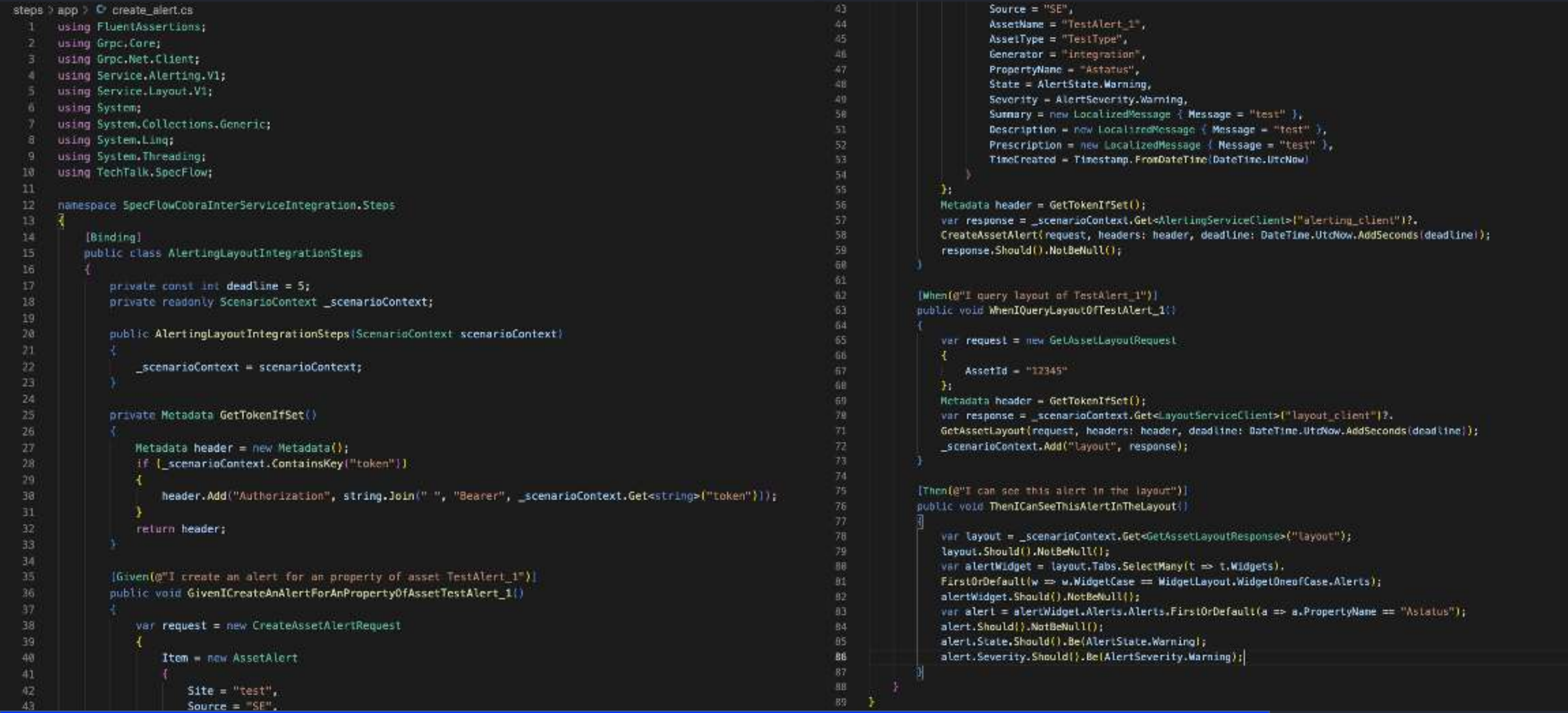

test_code_example.txt

```
            }

            private Metadata GetTokenIfSet()
            {
                Metadata header = new Metadata();
                if (_scenarioContext.ContainsKey("token"))
                {
                    header.Add("Authorization", string.Join(" ", "Bearer", _scenarioContext.Get<string>("token")));
                }

                return header;
            }

            [Then(@"these assets should exist")]
            public void ThenTheseAssetsShouldExist(Table table)
            {
                foreach (var row in table.Rows)
                {
                    Assetservice.V1.ListAssetsRequest request = new Assetservice.V1.ListAssetsRequest
                    {
                        Filter = new Assetservice.V1.AssetFilter
                        {
                            Owner = new Assetservice.V1.Owner { Name = row[0] },
                            Site = new Assetservice.V1.Site { Name = row[1] },
                            Source = new Assetservice.V1.Source { Name = row[2] },
                            Name = row[3]
                        }
                    };

                    Metadata header = GetTokenIfSet();
                    var assets = client?.ListAssets(request, headers: header, deadline: DateTime.UtcNow.AddSeconds(deadline));
                    assets?.Assets.Should().HaveCount(1);
                }
            }

            [AfterScenario("deleteAssets")]
            public void DeleteTestItem()
            {
                if (_scenarioContext["assets"] is List<string> ids)
                {
                    foreach (var id in ids)
                    {
                        Metadata header = GetTokenIfSet();
                        Assetservice.V1.DeleteAssetRequest request = new Assetservice.V1.DeleteAssetRequest() { Id = id };
                        _ = client?.DeleteAsset(request, headers: header, deadline: DateTime.UtcNow.AddSeconds(deadline));
                    }
                }
            }
        }
    }
```

Prompt input

```
"""
Generated the C sharp test code only for the step functions, based on the following example:
{example}
from the following test scenario:
Given I create an alert for a property of asset TestAlert_1
When I query the layout of TestAlert_1
Then I can see this alert in the layout
Generated Code:
"""
```

Generated code



steps > app > create_alert.cs

```csharp
using FluentAssertions;
using Grpc.Core;
using Grpc.Net.Client;
using Service.Alerting.V1;
using Service.Layout.V1;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using TechTalk.SpecFlow;

namespace SpecFlowCobraInterServiceIntegration.Steps
{
    [Binding]
    public class AlertingLayoutIntegrationSteps
    {
        private const int deadline = 5;
        private readonly ScenarioContext _scenarioContext;

        public AlertingLayoutIntegrationSteps(ScenarioContext scenarioContext)
        {
            _scenarioContext = scenarioContext;
        }

        private Metadata GetTokenIfSet()
        {
            Metadata header = new Metadata();
            if (_scenarioContext.ContainsKey("token"))
            {
                header.Add("Authorization", string.Join(" ", "Bearer", _scenarioContext.Get<string>("token")));
            }
            return header;
        }

        [Given(@"I create an alert for an property of asset TestAlert_1")]
        public void GivenICreateAnAlertForAnPropertyOfAssetTestAlert_1()
        {
            var request = new CreateAssetAlertRequest
            {
                Item = new AssetAlert
                {
                    Site = "test",
                    Source = "SE",
                    AssetName = "TestAlert_1",
                    AssetType = "TestType",
                    Generator = "integration",
                    PropertyName = "Astatus",
                    State = AlertState.Warning,
                    Severity = AlertSeverity.Warning,
                    Summary = new LocalizedMessage { Message = "test" },
                    Description = new LocalizedMessage { Message = "test" },
                    Prescription = new LocalizedMessage { Message = "test" },
                    TimeCreated = Timestamp.FromDateTime(DateTime.UtcNow)
                }
            };
            Metadata header = GetTokenIfSet();
            var response = _scenarioContext.Get<AlertingServiceClient>("alerting_client")?.
            CreateAssetAlert(request, headers: header, deadline: DateTime.UtcNow.AddSeconds(deadline));
            response.Should().NotBeNull();
        }

        [When(@"I query layout of TestAlert_1")]
        public void WhenIQueryLayoutOfTestAlert_1()
        {
            var request = new GetAssetLayoutRequest
            {
                AssetId = "12345"
            };
            Metadata header = GetTokenIfSet();
            var response = _scenarioContext.Get<LayoutServiceClient>("layout_client")?.
            GetAssetLayout(request, headers: header, deadline: DateTime.UtcNow.AddSeconds(deadline));
            _scenarioContext.Add("layout", response);
        }

        [Then(@"I can see this alert in the layout")]
        public void ThenICanSeeThisAlertInTheLayout()
        {
            var layout = _scenarioContext.Get<GetAssetLayoutResponse>("layout");
            layout.Should().NotBeNull();
            var alertWidget = layout.Tabs.SelectMany(t => t.Widgets).
            FirstOrDefault(w => w.WidgetCase == WidgetLayout.WidgetOneofCase.Alerts);
            alertWidget.Should().NotBeNull();
            var alert = alertWidget.Alerts.Alerts.FirstOrDefault(a => a.PropertyName == "Astatus");
            alert.Should().NotBeNull();
            alert.State.Should().Be(AlertState.Warning);
            alert.Severity.Should().Be(AlertSeverity.Warning);
        }
    }
}
```

Future directions of AI in Test Automation

AI will continue to shape the future of testing. It promises to further reduce manual effort and improve test reliability.

1

Enhanced Test Autonomy and Reduced Maintenance

Automated test generation and self-healing mechanisms.

Integration of NLP in test creation and maintenance.

2

Intelligent Test Prioritization and Selection

Contextual prioritization and risk Assessment.

Failure prediction and anomaly detection.

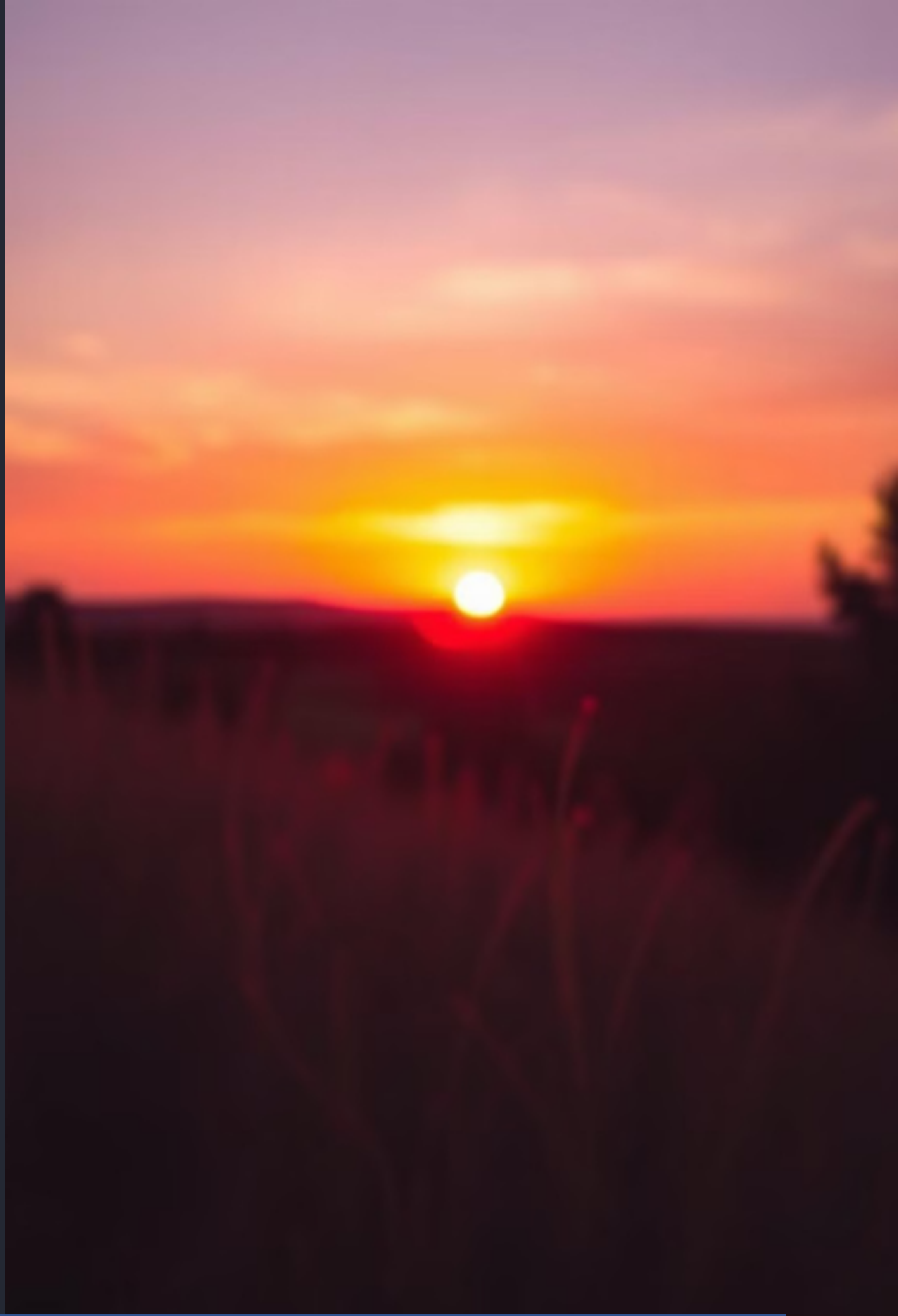
3

Improved Reporting and Insight Generation

Dynamic reporting and interactive dashboards

References and Further Reading

1. Battina, Dhaya Sindhu. **"Artificial intelligence in software test automation: A systematic literature review."** *International Journal of Emerging Technologies and Innovative Research*
2. Ricca, Filippo, Alessandro Marchetto, and Andrea Stocco. "**Ai-based test automation: A grey literature analysis.**" *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021.
3. T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin and R. Shanmugam, "**AI for Testing Today and Tomorrow: Industry Perspectives**," *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, Newark, CA, US..
4. Gao, Jerry, et al. "**What is AI software testing? and why.**" *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019.
5. Wang, Junjie, et al. "**Software testing with large language models: Survey, landscape, and vision.**" *IEEE Transactions on Software Engineering* (2024).
6. Feldt, Robert, et al. "**Towards autonomous testing agents via conversational large language models.**" *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023.
7. Pham, Phuoc, Vu Nguyen, and Tien Nguyen. **"A Review of AI-augmented End-to-End Test Automation Tools."** *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022.
8. Tufano, Michele, et al. "**Generating accurate assert statements for unit test cases using pretrained transformers.**" *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test*. 2022.



Thank you!

AI Potential

AI has the power to revolutionize test automation, addressing key pain points.

Efficiency Gains

Either in-house implementation or testing tools, engaging AI can potentially lead to significant improvements in testing efficiency.

Future-Ready

Embracing AI in testing prepares teams for the evolving landscape of software development.