

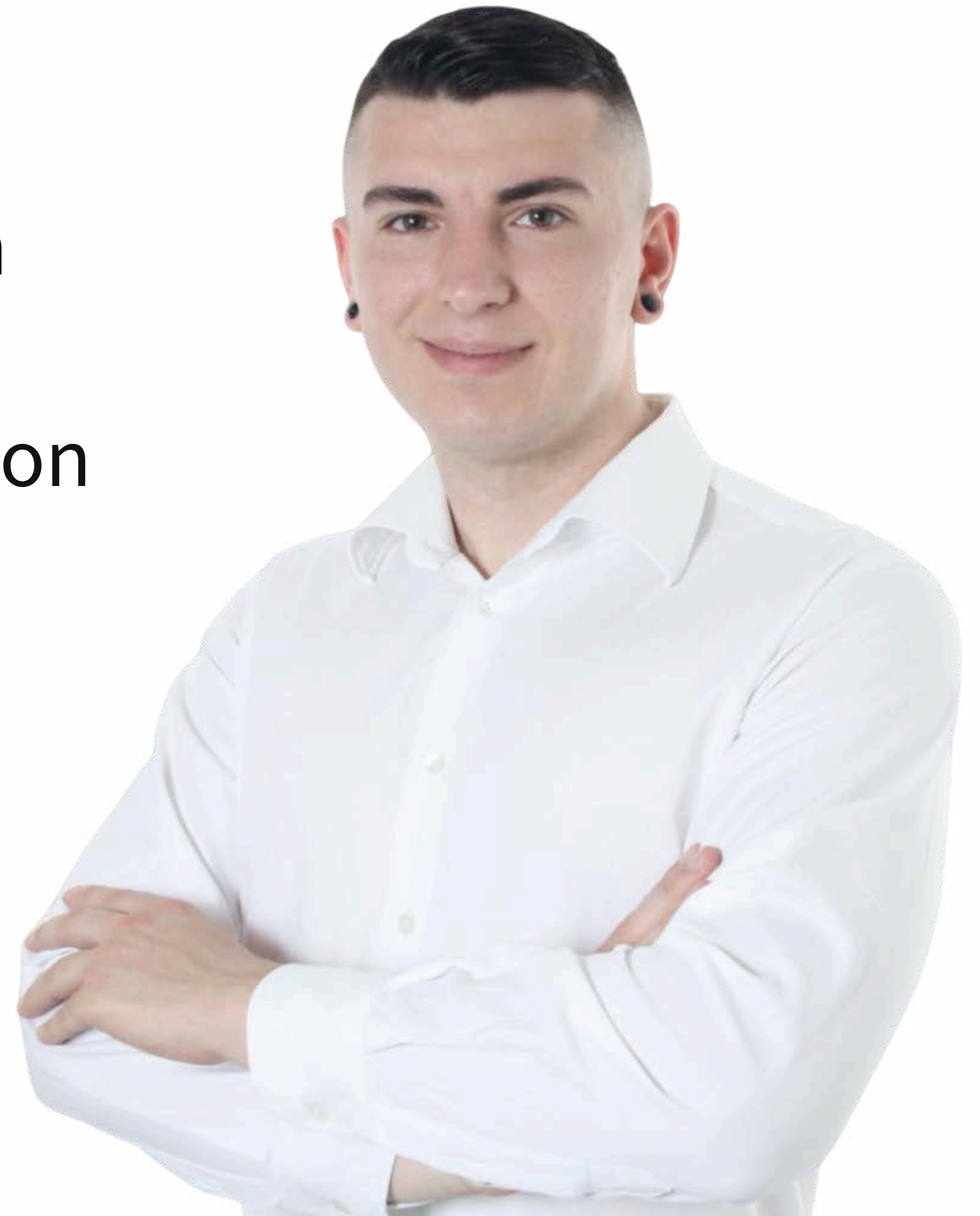
# **Empowering Testing with AI:** Best Practice and Strategies That Truly Work



@ludovicobesana

# About me

- Senior Test Engineer @Nearform
- SheTech Ambassador
- Grafana & BrowserStack Champion
- Startups Co-Founder
- Open Source Contributor
- Some passions: 🚗, 🐕, 🎮





# Event Organizer:



## Coding Bunker

<https://www.meetup.com/it-IT/coding-bunker/>



## Ministry of Testing Milan

<https://www.meetup.com/it-it/mot-milan/>

@ludovicobesana

# Nearform.



## OSS at Nearform\_

“At the heart of Nearform’s mission is a profound commitment to the Node.js ecosystem and the OpenJS Foundation. Their support and contributions are instrumental in driving the growth and success of the open-source community.” - Robin Bender Ginn, Executive Director, OpenJS Foundation

15m+  
Downloads

Over 15 million  
downloads every week



Node JS

1192k+  
Contributions

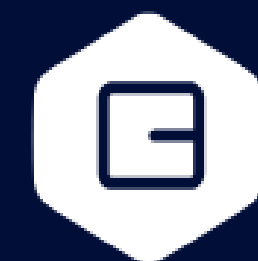
Thousands of contributions  
every year



Fastify

166+  
Packages

We maintain many  
popular packages for the  
community



Clinic JS

10+ years  
experience

We are a trusted partner for  
our international clients,  
passionately delivering  
digital excellence across a  
range of industries.



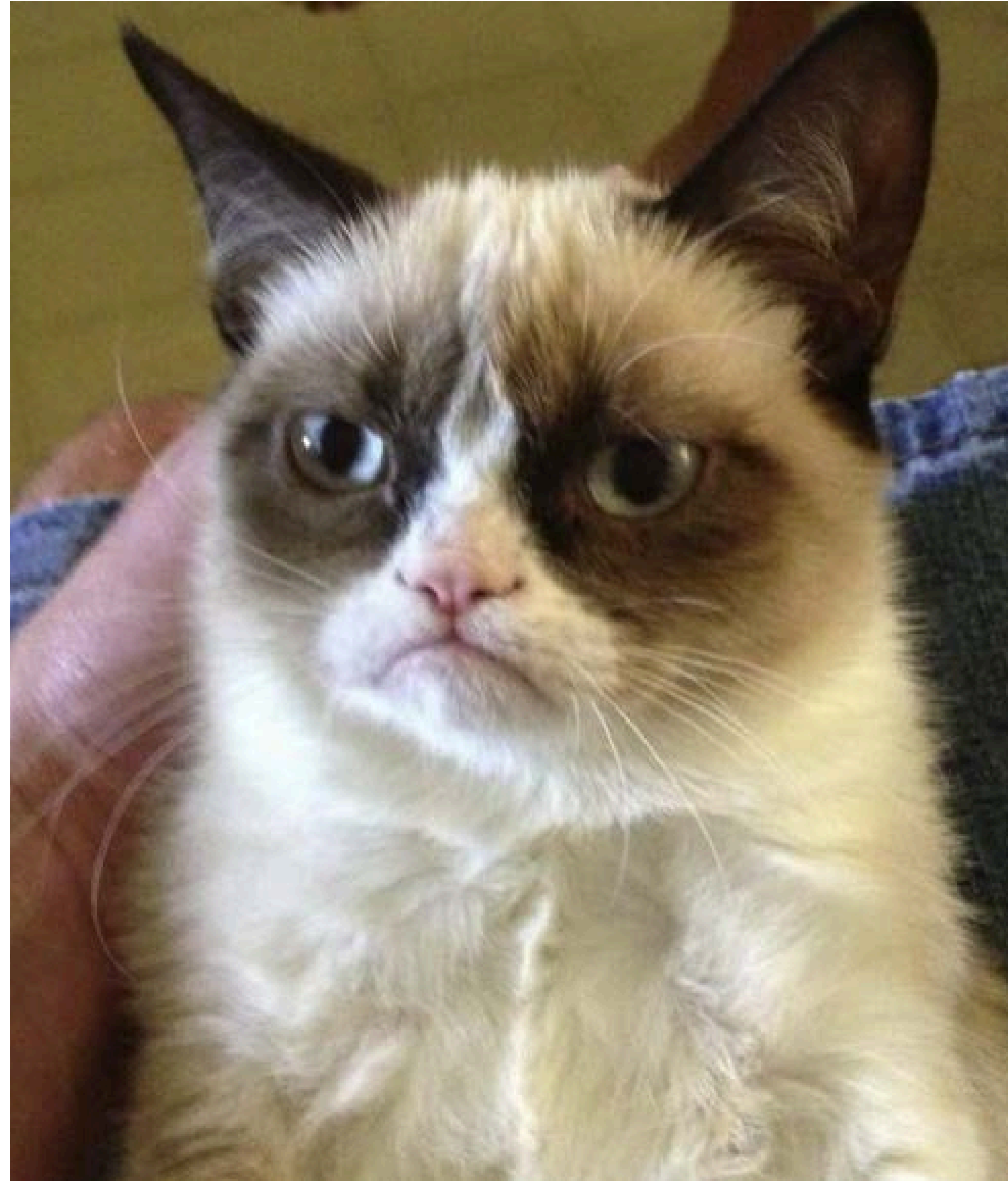
Pino



React Native Auth

We're hiring! [nearform.com/careers](https://nearform.com/careers)

# AI

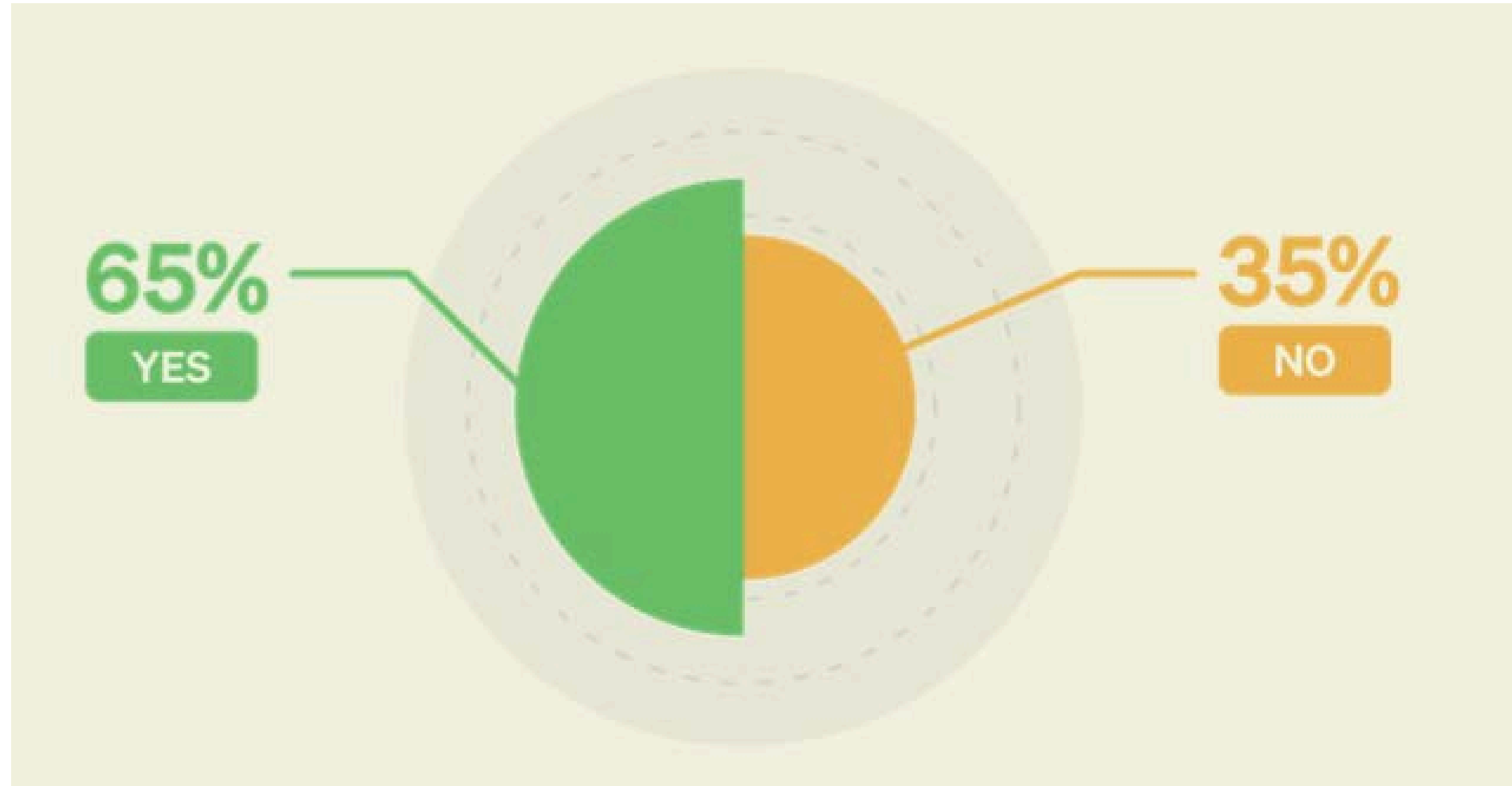






**Do you use ChatGPT/Copilot  
when doing testing?**

**Have you already introduced AI  
into your QA processes?**



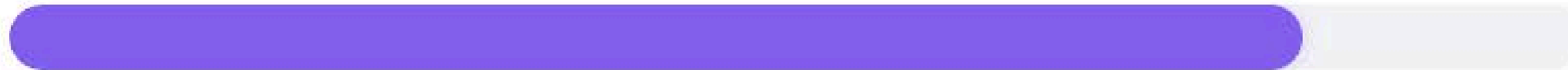
**Source: TestRail AI Report 2024**

**65% of QA teams have already  
integrated AI into their processes.**

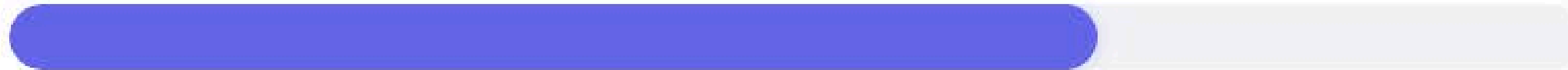
Source: TestRail AI Report 2024



**82%** use AI for code generation/review



**69%** say less than 25% of code is AI-made



**46%** use AI tools multiple times a day



**Source: State of AI 2025**

# The job market is changing too...

### What we're looking for?

- Extensive experience in a QA leadership role (5–7 years), preferably in Agile, fast-paced environments with distributed teams
- Proven ability to lead and develop diverse QA teams, inspiring both individual ownership and collective collaboration
- Ability to adapt to changing project requirements and priorities, maintaining a flexible approach to QA processes.
- Deep expertise in both manual and automated testing — from strategy to execution — with hands-on familiarity with modern testing frameworks and tools
- A strategic mindset paired with technical fluency — someone who can think big picture but isn't afraid to roll up their sleeves when needed
- Experience or strong interest in AI-assisted testing, intelligent automation, or leveraging machine learning to improve QA practices is a big plus
- Exceptional communication and interpersonal skills — empathetic, clear, and capable of building strong relationships across technical and non-technical teams
- Familiarity with the online gambling, sports betting, or iGaming industry is a strong advantage — understanding regulatory requirements, risk profiles, and user expectations in this space is highly valuable



altermAlnd



## AI Quality Assurance & GenAI Operations Engineer

Milano, Lombardia, Italia · 3 settimane fa · 60 candidati

Promossa da recruiter · Nessuna informazione sulla risposta



Ibrido · Contratto · Livello medio-alto



7 competenze su 10 corrispondono al tuo profilo, potresti avere un profilo adeguato



Vedi come ti posizioni rispetto a 60 candidati. [Riattiva Premium](#)

Candidatura semplice

Salvato

### Informazioni sull'offerta di lavoro

*Do you want to be part of a technology company with an AI-first approach, shaping the invisible by blending AI innovation with human experience?*

altermAlnd is a company that offers solutions enabling organizations to overcome limits, uncover hidden opportunities, and drive transformative growth.

**Location:** Flexible / Remote

#### About the Role

We are looking for a hands-on engineer (*mid-level or senior, depending on experience*) who will own both Quality Assurance for our AI initiatives and the day-to-day running of our cloud-native Generative-AI applications. You will design and automate test strategies for large-language-model (LLM) workflows, keep production GenAI services healthy, and contribute improvements using Python, Docker, and modern open-source frameworks such as LlamaIndex and AutoGen.

#### What will you do?

- **AI Quality Assurance**
- Create, automate, and execute test plans for LLM pipelines, data ingestion, and RAG (Retrieval-Augmented Generation) solutions.
- Monitor model accuracy, bias, and drift; investigate anomalies and coordinate fixes with data scientists.





⌵

jules

ludovicobesana/PlaywrightStarterKit

+

Feedback

// Add more test cases

Today 09:38

1

Add a UI test for login with an invalid username in tests/ui/desktop-tests/ds.login.spec.ts.

✓

2

Add a UI test for adding an item to the cart from the PDP in a new file tests/ui/desktop-tests/ds.pdp.spec.ts.

✓

3

Create page object model (POM) for PDP if it doesn't fully support cart interactions in services/pages/pdp.page.ts.

✓

4

Create step definitions for PDP cart interactions in services/steps/pdp.steps.ts.

✓

5

Add an API test for creating a new pet using POST /pet in tests/api/api.pet.spec.ts.

✓

6

Submit the changes.

✓

Hide

I've created a plan to add more test cases across UI and API. Please review the plan and let me know if you have any feedback. Otherwise, hit approve, and I'll get started.

Plan approved

Added a UI test for login with an invalid username, including a new step definition for checking invalid credentials error message.

Updated services/steps/login.steps.ts and tests/ui/desktop-tests/ds.login.spec.ts

Working on

Add a UI test for adding an item to the cart from the PDP in a new file tests/ui/desktop-tests/ds.pdp.spec.ts.

This test will first log in (using existing login steps).

- Navigate to a PDP page (this might require selecting a product from a PLP or directly visiting a PDP URL, if the structure allows).
- Click the "Add to Cart" button for a product.
- Verify that the cart updates correctly (e.g., cart icon shows item count, or navigate to cart page and verify item is present).

Talk to Jules

Jules can make mistakes so double-check it and use code with caution.

<> Code

login.steps.ts

ds.login.spec.ts

services/steps/login.steps.ts

```
1: import { Page, expect } from '@playwright/test';
2: import { LoginPage } from '@pages/login.page';
3:
4: export default class LoginSteps {
5:   private page: Page;
6:   private loginPage: LoginPage;
7:
8:   constructor(page: Page, loginPage: LoginPage) {
9:     this.page = page;
10:    this.loginPage = new LoginPage(page);
11:  }
12:
13:   async typeUsername(username: string) {
14:     await this.loginPage.getUsernameInput().fill(username);
15:   }
16:
17:   async typePassword(password: string) {
18:     await this.loginPage.getPasswordInput().fill(password);
19:   }
20:
21:   async clickLoginButton() {
22:     await this.loginPage.getLoginButton().click();
23:   }
24:
25:   async checkUsernameErrorMessage() {
26:     await expect.soft(this.loginPage.getError()).toBeVisible();
27:     await expect(this.loginPage.getError()).toContainText('Epic sadface:
28:   }
29: }
30:
```

```
1: import { Page, expect } from '@playwright/test';
2: import { LoginPage } from '@pages/login.page';
3:
4: export default class LoginSteps {
5:   private page: Page;
6:   private loginPage: LoginPage;
7:
8:   constructor(page: Page, loginPage: LoginPage) {
9:     this.page = page;
10:    this.loginPage = new LoginPage(page);
11:  }
12:
13:   async typeUsername(username: string) {
14:     await this.loginPage.getUsernameInput().fill(username);
15:   }
16:
17:   async typePassword(password: string) {
18:     await this.loginPage.getPasswordInput().fill(password);
19:   }
20:
21:   async clickLoginButton() {
22:     await this.loginPage.getLoginButton().click();
23:   }
24:
25:   async checkUsernameErrorMessage() {
26:     await expect.soft(this.loginPage.getError()).toBeVisible();
27:     await expect(this.loginPage.getError()).toContainText('Epic sadface:
28:   }
29:
30:   async checkInvalidCredentialsErrorMessage() {
31:     await expect.soft(this.loginPage.getError()).toBeVisible();
32:     await expect(this.loginPage.getError()).toContainText('Epic sadface:
33:   }
34: }
35:
```

bcs

The  
Chartered  
Institute  
for IT

@ludovicobesana



The screenshot displays a GitHub pull request interface. At the top, the repository is identified as 'ludovicobesano/PlaywrightStarterKit'. The pull request title is 'Add more test cases', and it is marked as 'Ready for review' with a green checkmark and a '+111 -0' status.

The 'Plan' section on the left lists the following changes:

- I've created a plan to add more test cases across UI and API. Please review the plan and let me know if you have any feedback. Otherwise, hit approve, and I'll get started.
- Added a UI test for login with an invalid username, including a new step definition for checking invalid credentials error message.
- Updated `services/steps/login.steps.ts` and `tests/ui/desktop-tests/ds.login.spec.ts`.
- Added a UI test for adding an item to the cart from PDP. This included creating the new test file `tests/ui/desktop-tests/ds.pdp.spec.ts`, updating `services/pages/pdp.page.ts` with new selectors and methods for cart interactions, and adding new step definitions in `services/steps/pdp.steps.ts`.
- Updated `selectors.json`, `services/pages/pdp.page.ts`, `services/steps/pdp.steps.ts` and 1 more.
- The Page Object Model for PDP (`services/pages/pdp.page.ts`) was updated to support cart interactions as part of the previous step.
- The step definitions for PDP cart interactions (`services/steps/pdp.steps.ts`) were created as part of implementing the PDP 'add to cart' test.
- Added an API test case for 'POST /pet' to create a new pet in `tests/api/api.pet.spec.ts`. The test uses a dynamic pet ID and verifies the response details.
- Updated `tests/api/api.pet.spec.ts`.


The 'Diff' view on the right shows the changes to the test files. The diff for `tests/ui/desktop-tests/ds.login.spec.ts` includes:

```
1 import { test } from '@playwright/test';
2 import { LoginPage } from '@pages/login.page';
3 import { PDPPage } from '@pages/pdp.page';
4 import LoginSteps from '@steps/login.steps';
5 import PDPSteps from '@steps/pdp.steps';
6
7 test.describe('Login Feature @Desktop', () => {
8   let loginPage: LoginPage;
9   let pdpPage: PDPPage;
10   let loginPageSteps: LoginSteps;
11   let pdpPageSteps: PDPSteps;
12
13   test.beforeEach(async ({ page }) => {
14     await page.goto('/');
15   });
16
17   test('Failed login shows error message @Atomic', async ({ page }) => {
18     loginPage = new LoginPage(page);
19     loginPageSteps = new LoginSteps(page, loginPage);
20     await loginPageSteps.clickLoginButton();
21     await loginPageSteps.checkUsernameErrorMessage();
22   });
23
24   test('User can complete login @Flow', async ({ page }) => {
25     loginPage = new LoginPage(page);
26     loginPageSteps = new LoginSteps(page, loginPage);
27     pdpPageSteps = new PDPSteps(page, pdpPage);
28
29     await loginPageSteps.typeUsername('standard_user');
30     await loginPageSteps.typePassword('secret_sauce');
31     await loginPageSteps.clickLoginButton();
32     await pdpPageSteps.checkPDPPageTitle();
33   });
34 });
```

The diff for `tests/ui/desktop-tests/ds.pdp.spec.ts` includes:


```
1 import { test } from '@playwright/test';
2 import { LoginPage } from '@pages/login.page';
3 import { PDPPage } from '@pages/pdp.page';
4 import LoginSteps from '@steps/login.steps';
5 import PDPSteps from '@steps/pdp.steps';
6
7 test.describe('Login Feature @Desktop', () => {
8   let loginPage: LoginPage;
9   let pdpPage: PDPPage;
10   let loginPageSteps: LoginSteps;
11   let pdpPageSteps: PDPSteps;
12
13   test.beforeEach(async ({ page }) => {
14     await page.goto('/');
15   });
16
17   test('Failed login shows error message @Atomic', async ({ page }) => {
18     loginPage = new LoginPage(page);
19     loginPageSteps = new LoginSteps(page, loginPage);
20     await loginPageSteps.clickLoginButton();
21     await loginPageSteps.checkUsernameErrorMessage();
22   });
23
24   test('Failed login with invalid username shows error message @Atomic', async ({ page }) => {
25     loginPage = new LoginPage(page);
26     loginPageSteps = new LoginSteps(page, loginPage);
27     await loginPageSteps.typeUsername('invalid_user');
28     await loginPageSteps.typePassword('secret_sauce');
29     await loginPageSteps.clickLoginButton();
30     await loginPageSteps.checkInvalidCredentialsErrorMessage();
31   });
32
33   test('User can complete login @Flow', async ({ page }) => {
34     loginPage = new LoginPage(page);
35     loginPageSteps = new LoginSteps(page, loginPage);
36     pdpPageSteps = new PDPSteps(page, pdpPage);
37
38     await loginPageSteps.typeUsername('standard_user');
39     await loginPageSteps.typePassword('secret_sauce');
40     await loginPageSteps.clickLoginButton();
41     await pdpPageSteps.checkPDPPageTitle();
42   });
43 });
```

At the bottom, there is a 'Publish branch' button and a note: 'Jules can make mistakes so double-check it and use code with caution'.



Ready for review 🚀

+111 -0

 add-more-test-cases

Add more test cases for UI and API



This commit introduces several new test cases to improve test coverage:

UI Tests (Desktop):


- Added a test for failed login attempts due to an invalid username.
- Added a test for successfully adding a product to the cart from the Product Display Page (PDP).
- Updated Page Object Models (POM) and Step Definitions for PDP interactions, including adding to cart and verifying cart badge count.

API Tests (Pet Store):

- Added a test for creating a new pet using the POST /pet endpoint.
- This test uses a dynamic pet ID for robustness and verifies the created pet's details in the response.

How'd Jules do?  

Time: 5 mins

Publish branch 





# Risks and Challenges of AI in QA

# Data from the TestRail 2024 Report

---

## Data from the TestRail 2024 Report

54% 🤔 of teams are unsure about the benefits AI can bring.

## Privacy concerns:

53% 🔒 are worried about data security and compliance.

# Lack of Context

AI generates output based on the data provided but doesn't understand context like a human tester.

**Risk:** Inaccurate or incomplete answers.

**Solution:** Always review outputs and complement them with human expertise.



# Prompt Quality

---

ChatGPT (or Gemini) relies on the quality of the input prompts.

**Risk:** Vague or incomplete prompts can lead to low-quality results.

**Solution:** Train teams to write clear, precise prompts, and repeat the same prompt multiple times to test effectiveness.

# Answer Reliability

---

AI can produce responses that sound convincing but are incorrect or incomplete.

**Risk:** False confidence in generated test results.

**Solution:** Validate every AI output with manual checks.

# Tool Opacity

---

**Problem:** Some tools, like modern AI-based IDEs, don't clearly state where processed data ends up.

**Risk:** Exposure of sensitive or proprietary data without knowing how it will be used.

**Solution:** Use transparent tools with clear security policies and, when possible, opt for on-premise or private hosting.



# Why integrate AI into QA?

# The goal isn't to replace QA...

---

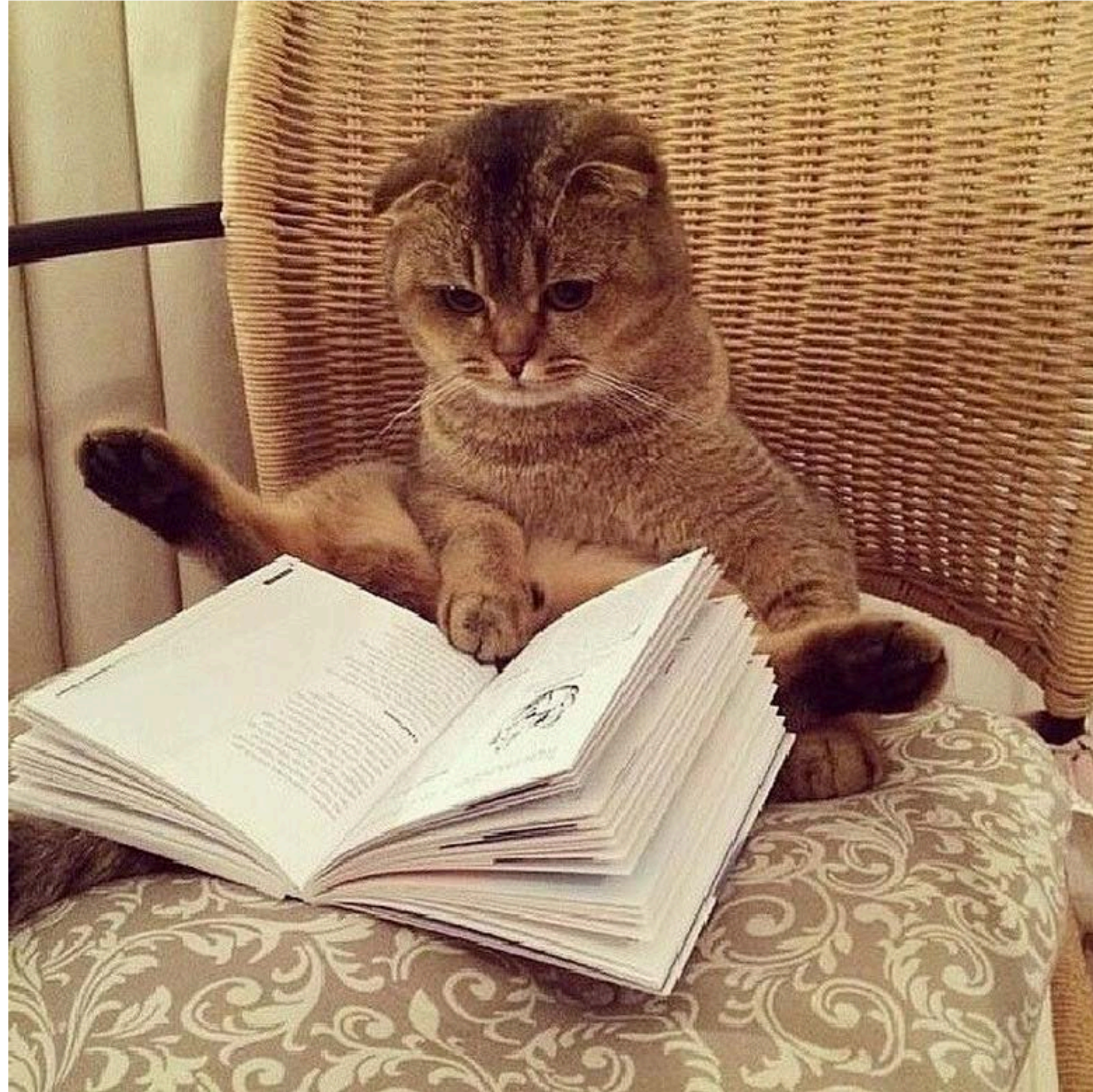
It's to **enhance it**. In fact, AI can:

- Automate repetitive and time-consuming tasks
- Improve test coverage
- Foster better collaboration between QA, devs, and stakeholders
- Reduce release times
- Assist in bug and anomaly triage



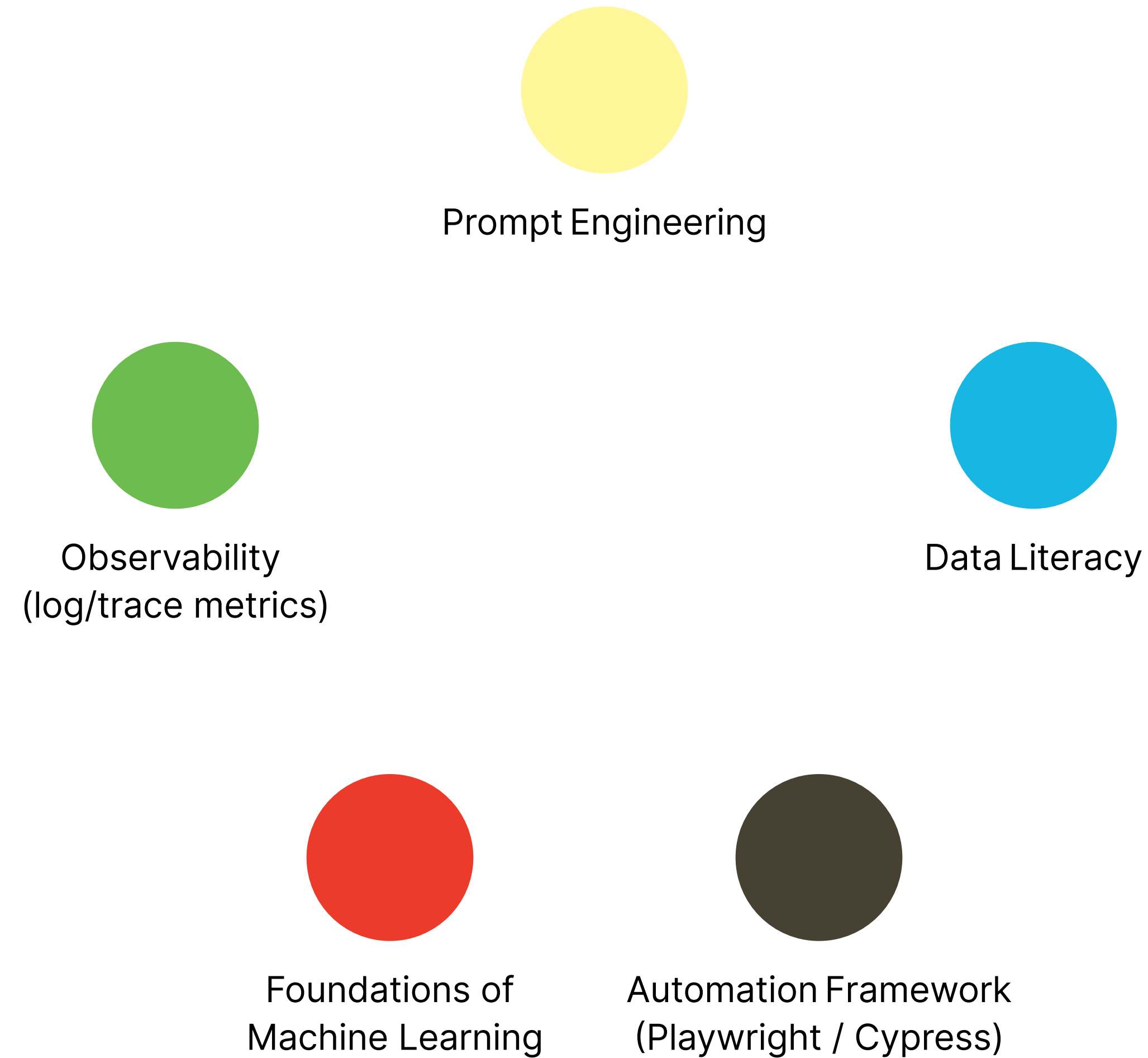
# How do I integrate it?







# The 5 Essential Skills for an AI-Ready QA



# Why?



## Prompt Engineering

It's the skill with the highest immediate ROI. It drastically improves the quality of generated output.



## Automation Framework

Many AI tools generate test code but you need to know how to read, adapt, and maintain it.



## Observability

Many AI tools rely on logs and metrics. If QA can't read or structure them, AI can't help.



## Data Literacy

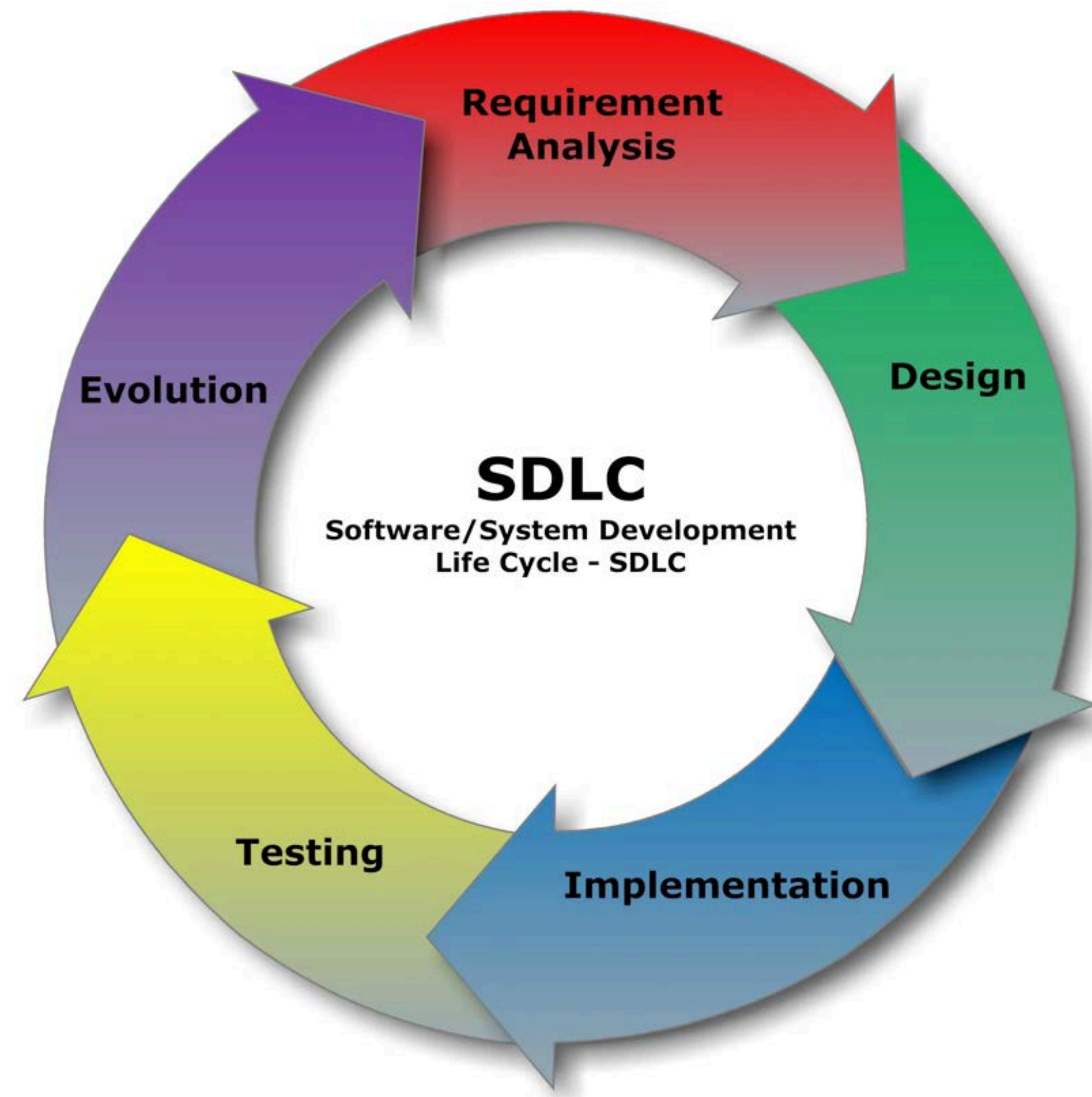
AI needs good data to work. Knowing how to prepare, normalize, and anonymize it makes all the difference.



## Foundations of Machine Learning

It enables analysis of large volumes of logs, grouping similar errors, and spotting outliers.

# Now let's reflect...

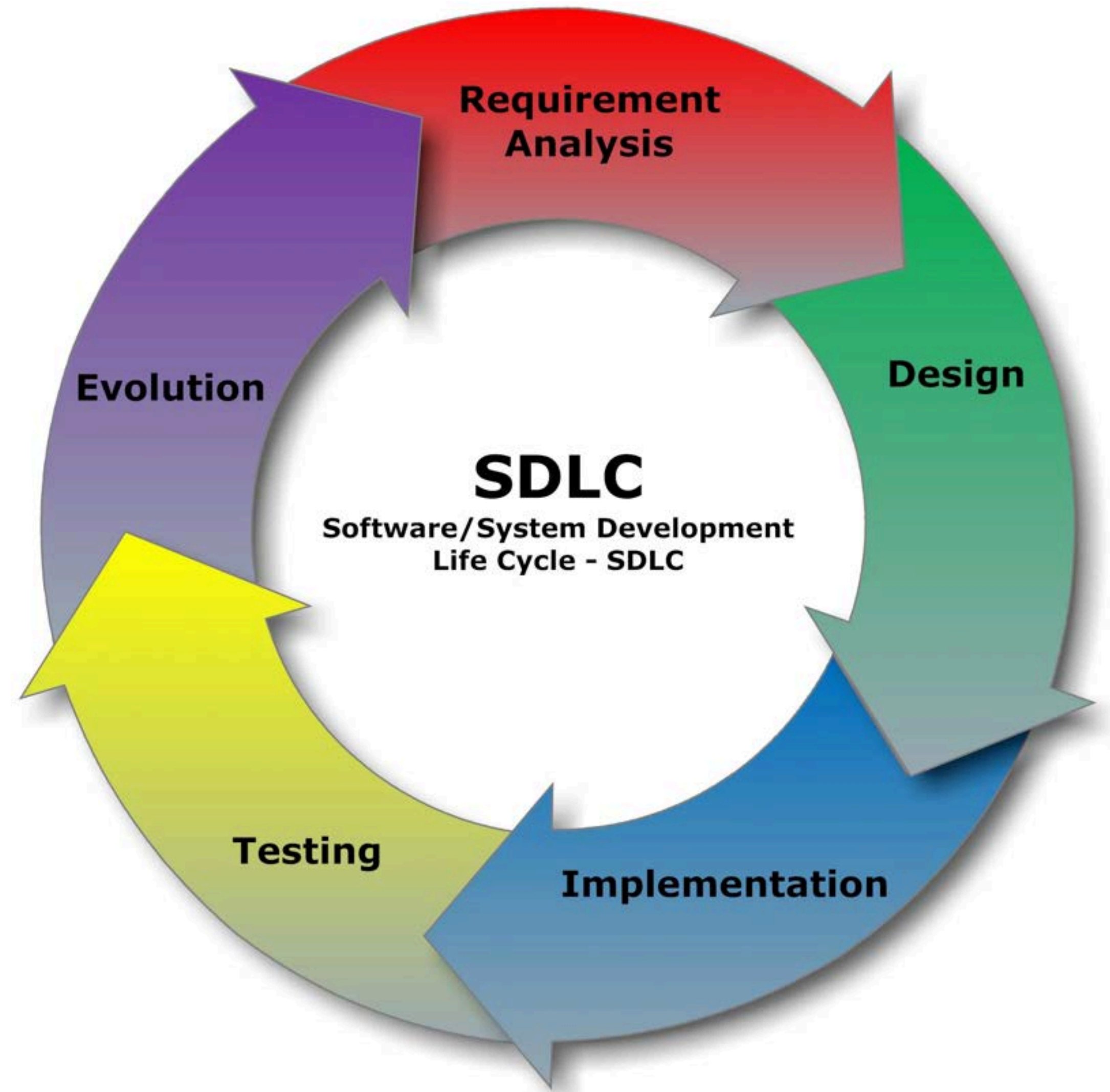


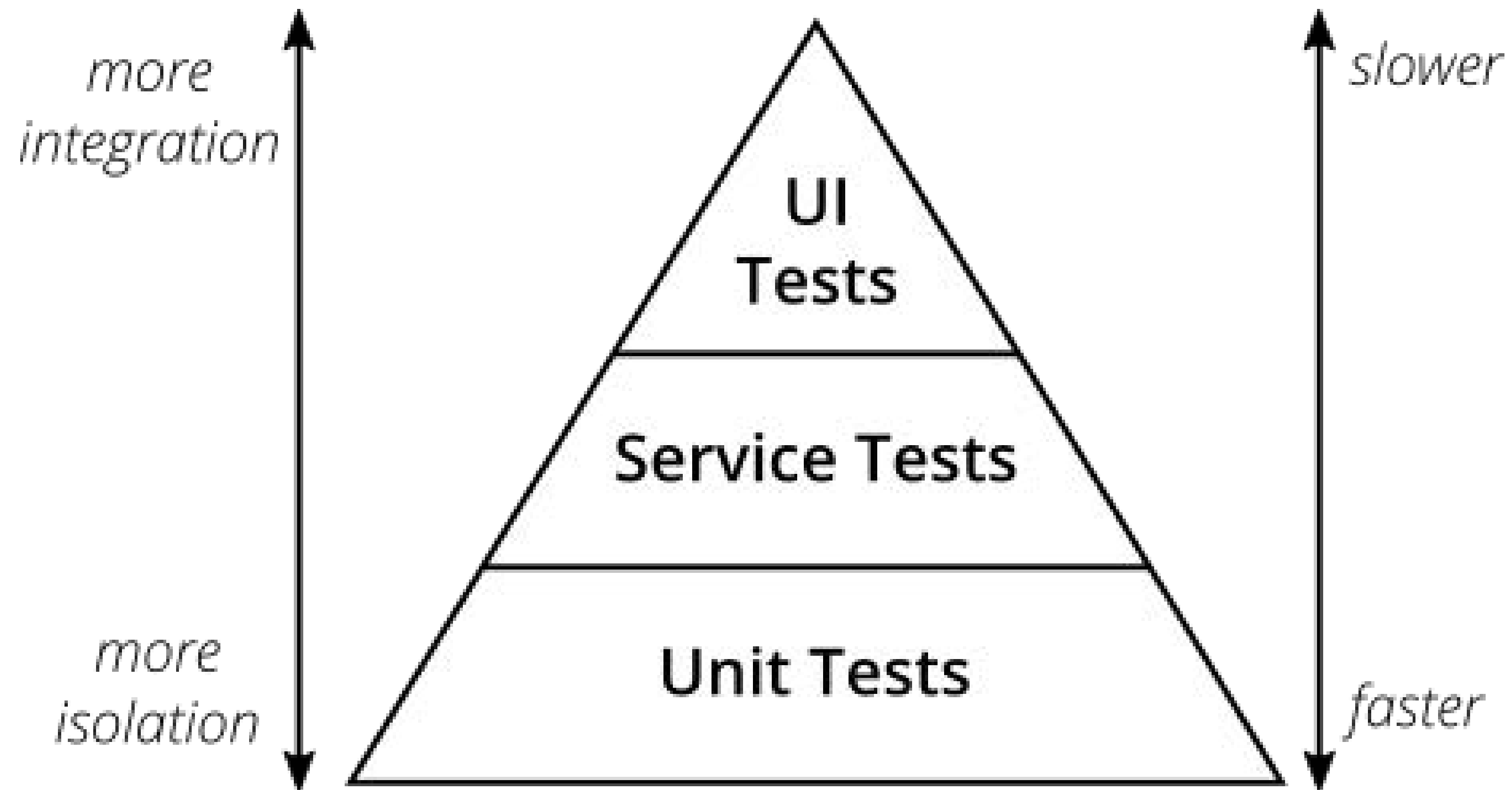


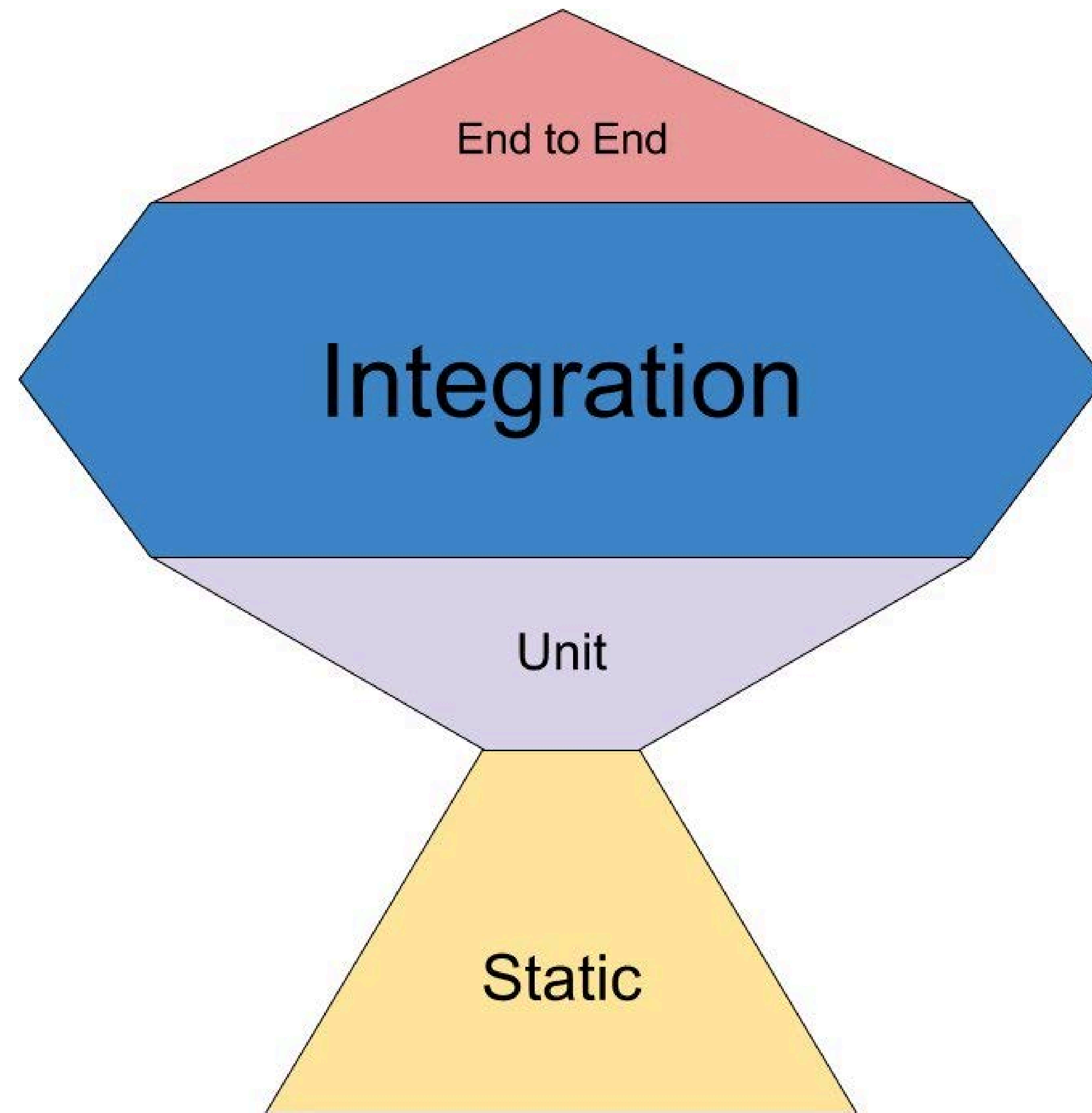
What activities are carried out in each phase?

For example:

- Writing user stories
- Writing unit tests
- etc.



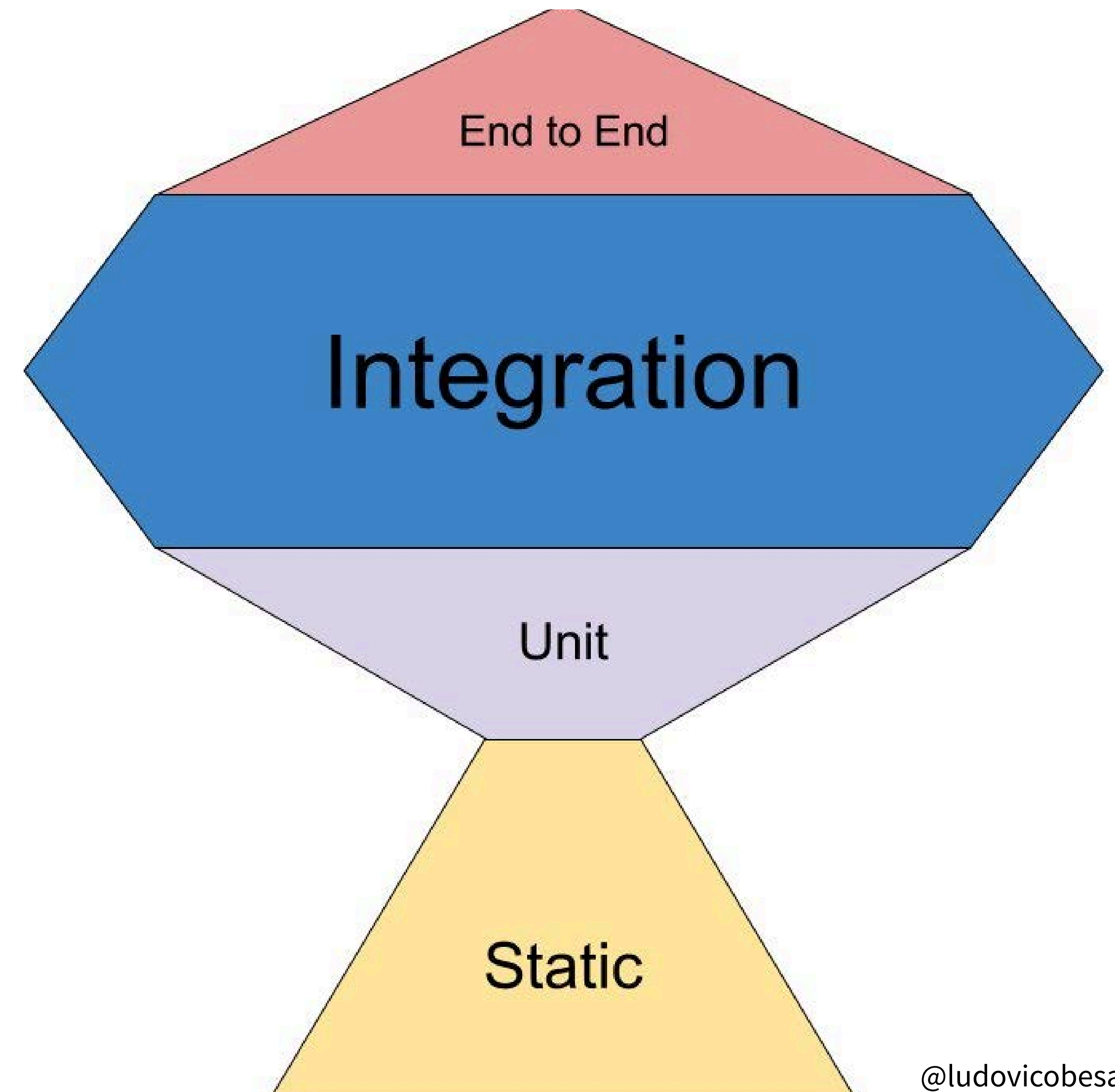




What types of tests are included in my pyramid?

For example:

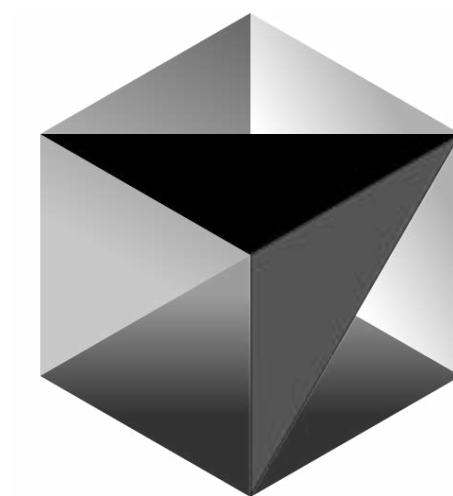
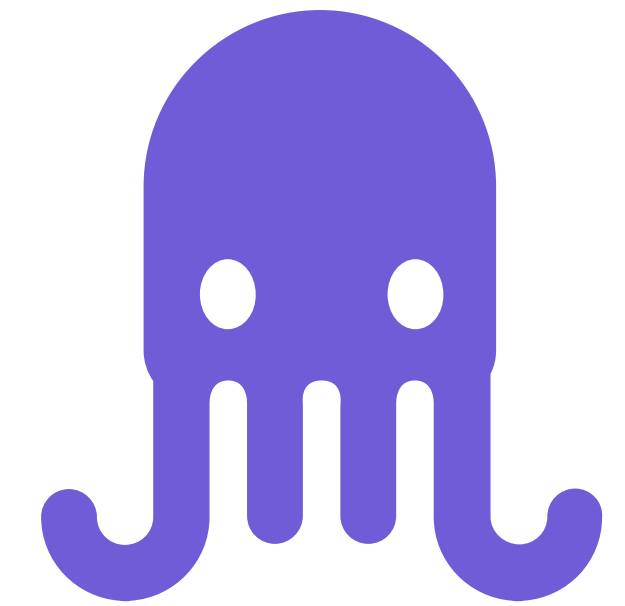
- UI
- Accessibility
- etc.



**Now I can look for  
the tools**









BrowserStack

Products

Developers

Live for Teams

Pricing

Sign in

Free Trial

Documentation

Test Management

Search across documentation

OVERVIEW

IMPORT PROJECTS AND TEST CASES

WORKING WITH TEST MANAGEMENT

TEST RUNS

GENERATIVE AI

Overview

Generate Test Cases using AI

Generate Test Runs using AI

Low Code Automation

TEST PLANS

UPLOAD TEST REPORTS USING CLI

REPORTS AND ANALYTICS

CI/CD INTEGRATION

INTEGRATE ISSUE TRACKERS

JIRA BINDING AND INTEGRATION

AZURE DEVOPS INTEGRATION

ADVANCED FEATURES

USER ACCESS CONTROL

TROUBLESHOOTING

REFERENCES

Generate Test Cases using AI

Learn how AI can assist with creating test cases in BrowserStack Test Management.

Test case authoring can be complex and time-consuming. To streamline this process, BrowserStack Test Management harnesses the power of generative AI. Our AI assistant analyzes your existing test cases, providing intelligent test case suggestions to simplify the creation of new test cases and autofill essential details. This approach leverages your existing test data to improve the accuracy of suggestions.

Key Features

Test Case Generation:

Generate a comprehensive set of test case suggestions based on your existing test cases, ensuring optimal test coverage.

Autofill Test Case details:

AI intelligently auto-populates test case fields (Description, Test Steps, Expected Results, Preconditions, Priority, and Test Case Type), saving you time and ensuring consistency.

Generate Test Cases

Create your first test case for AI to offer suggestions.

Follow the below steps to generate test case suggestions.

1 Click the **Suggest Test Cases** icon in the Test Case view.

Test Cases

unit-tests

functional-test

api

flaky-tests

visual

flaky-tests (2)

ID

TITLE

TC-177645

test\_flaky\_data

TC-177644

Setup\_hook\_for\_test\_flaky\_data

Priority

Medium

Medium

Tags

Suggest Test Cases

Save time with AI-generated Test Cases! Let AI figure out missing cases and fill in all the details.

Learn more

bcs

The  
Chartered  
Institute  
for IT

@ludovicobesana



CHAT: GITHUB COPILOT

GitHub Copilot

Sure, here's a set of detailed unit test functions for the `is_prime` function:

```
def test_is_prime_with_prime_numbers():
    assert is_prime(2) == True
    assert is_prime(3) == True
    assert is_prime(5) == True
    assert is_prime(7) == True
    assert is_prime(11) == True
    assert is_prime(13) == True
    assert is_prime(17) == True
    assert is_prime(19) == True
    assert is_prime(23) == True
    assert is_prime(29) == True

def test_is_prime_with_non_prime_number
```

JS index.js 1

prime.py

```
1 def is_prime(n):
2     """
3     Returns True if the given number
4     Args:
5     n (int): The number to check for
6     Returns:
7     bool: True if the given number is
8     """
9     if n < 2:
10        return False
11    for i in range(2, int(n**0.5)+1):
12        if n % i == 0:
13            return False
14    return True
```





Prompt:

You are a Senior Test Engineer with 10 years of experience. You will be provided with UI screenshots of a website. Your task is the following:  
Generate all test cases using the GIVEN-WHEN-THEN format, analyzing visible user interactions and underlying functionality.  
Organize the results in a table with four columns: "Test Title", "GIVEN", "WHEN", "THEN".  
Each row in the table must represent a distinct test case and clearly describe the initial conditions, the action, and the expected result.  
Export the table to an Excel file and provide me with a download link.  
The output must also be displayed in the chat as a table.  
Make sure the tests are thorough and cover all possible flows identified in the screenshots.



## Asgaard sofa

Rs. 250,000.00



5 Customer Review

Setting the bar as one of the loudest speakers in its class, the Kilburn is a compact, stout-hearted hero with a well-balanced audio which boasts a clear midrange and extended highs for a sound.

Size

☒ L ☐ XL ☐ XS

Color



- 1 +

Add To Cart

SKU : SS001

Category : Sofas

Tags : Sofa, Chair, Home, Shop

Share : [Facebook](#) [LinkedIn](#) [Twitter](#)



### Description

### Additional Information

### Reviews [5]

Embodying the raw, wayward spirit of rock 'n' roll, the Kilburn portable active stereo speaker takes the unmistakable look and sound of Marshall, unplugs the chords, and takes the show on the road.

Weighing in under 7 pounds, the Kilburn is a lightweight piece of vintage styled engineering. Setting the bar as one of the loudest speakers in its class, the Kilburn is a compact, stout-hearted hero with a well-balanced audio which boasts a clear midrange and extended highs for a sound that is both articulate and pronounced. The analogue knobs allow you to fine tune the controls



Test Cases For Single Product Page

	Title	GIVEN	WHEN	THEN	
1	Test product page loads correctly	User is on the product page	The product page loads	The product details are displayed with options to select size and color	
2	Test selecting size L	User selects 'Size L'	The user clicks on the 'L' button in Size options	The size 'L' is selected, and the button appears highlighted	
3	Test selecting color Black	User selects 'Color Black'	The user clicks on the 'Black' color option	The color 'Black' is selected, and the button appears highlighted	
4	Test increasing product quantity	User increases the quantity to 2	The user clicks the '+' icon to increase the quantity	The quantity is updated to 2 in the input box	
5	Test adding product to cart	User clicks 'Add to Cart'	The user clicks the 'Add to Cart' button	The product is added to the cart and the cart icon is updated with the new total	
6	Test viewing product description	User clicks on the 'Description' tab	The user clicks the 'Description' tab	The 'Description' section content is displayed	
7	Test viewing additional information	User clicks on the 'Additional Information' tab	The user clicks the 'Additional Information' tab	The 'Additional Information' section content is displayed	
8	Test viewing customer reviews	User clicks on the 'Reviews' tab	The user clicks the 'Reviews' tab	The 'Reviews' section content is displayed	
9	Test navigating to related product	User clicks a related product	The user clicks on a related product thumbnail	The user is navigated to the selected product's page	

tabella con quattro colonne: "Titolo del Test", "GIVEN", "WHEN", "THEN".  
Ogni riga della tabella deve rappresentare un caso di test distinto e descrivere in modo chiaro le condizioni iniziali, l'azione e il risultato atteso.  
Esporta la tabella in un file Excel e forniscimi un link per scaricarlo.  
L'output deve essere mostrato anche in chat sotto forma di tabella.  
Assicurati che i test siano completi e coprano tutti i possibili flussi rilevati dalle schermate.

Test Cases For Singl...

Ho generato i casi di test per la nuova schermata UI, inclusa la colonna con i titoli per ciascun caso. Puoi scaricare il file Excel utilizzando il link qui sotto:

[Scarica il file dei test case](#)

I test sono anche visualizzati qui sotto in forma di tabella. [-]

🔊 📄 👍 🗨️ ↺

🔗

Scrivi un messaggio a ChatGPT

⬆

bcs

The  
Chartered  
Institute  
for IT

@ludovicobesana



```
import { test, expect } from "@playwright/test";
import { auto } from "auto-playwright";

test("Shopping cart functionality", async ({ page }) => {
  await auto("aggiungi l'articolo 'X' al carrello", { page, test });
  await auto("vai alla pagina del carrello", { page, test });
  const itemsList = await auto("elenca tutti gli articoli nel carrello", { page, test });
  const totalPrice = await auto("qual è il prezzo totale?", { page, test });
  expect(itemsList).toContain('Articolo X');
  expect(totalPrice).toMatch(/\$\d+/);
});
```

# Best Practices and Strategy in Action



# Phase 1

---

## Requirements Analysis

- AI prompts for typos, ambiguities, and clarification questions.
- Output: semantic checks + auto-generated questions.

## User Personas Generation

- LLM with instructional prompt to create realistic user archetypes.
- Helps guide exploratory testing and identify risks.

# Phase 2

---

## LLM Notebook for Context

- All documents uploaded (requirements, user stories, business rules).
- Targeted prompts access ONLY this context (RAG → no hallucinations).

## BrowserStack Test Management + AI

- Auto-analysis of test coverage.
- Suggestions for improvements and duplicate detection.
- Increased coverage through targeted recommendations.

# Phase 3

---

## UI Comparison

- DOM-aware image comparison between builds → detects layout shifts, incorrect fonts, missing elements.
- Screenshot diff (build A vs build B) with optional auto-alerts for changes in text, contrast, accessibility colors, tab order.

## Code & Automation

- Custom GPTs trained on official docs → NO nonexistent methods.
- Page Objects, test scaffolding, and precise refactor assistance.

# Phase 4

---

## CI & Static Analysis

- AI in SonarQube: bug detection, smell analysis, and fix suggestions.
- Instead of simple alerts, it provides detailed remediation actions.

## Bot AI + MCP

- PR review bot: scans, clusters, and suggests fix locators.
- Uses MCP to orchestrate browser + network + DOM tests with a single prompt.

## Bug Filing with AI

- API-driven form → data collection, auto-filled issue template, automatic Jira submission.

# Phase 5

---

## Observability + Log Intelligence

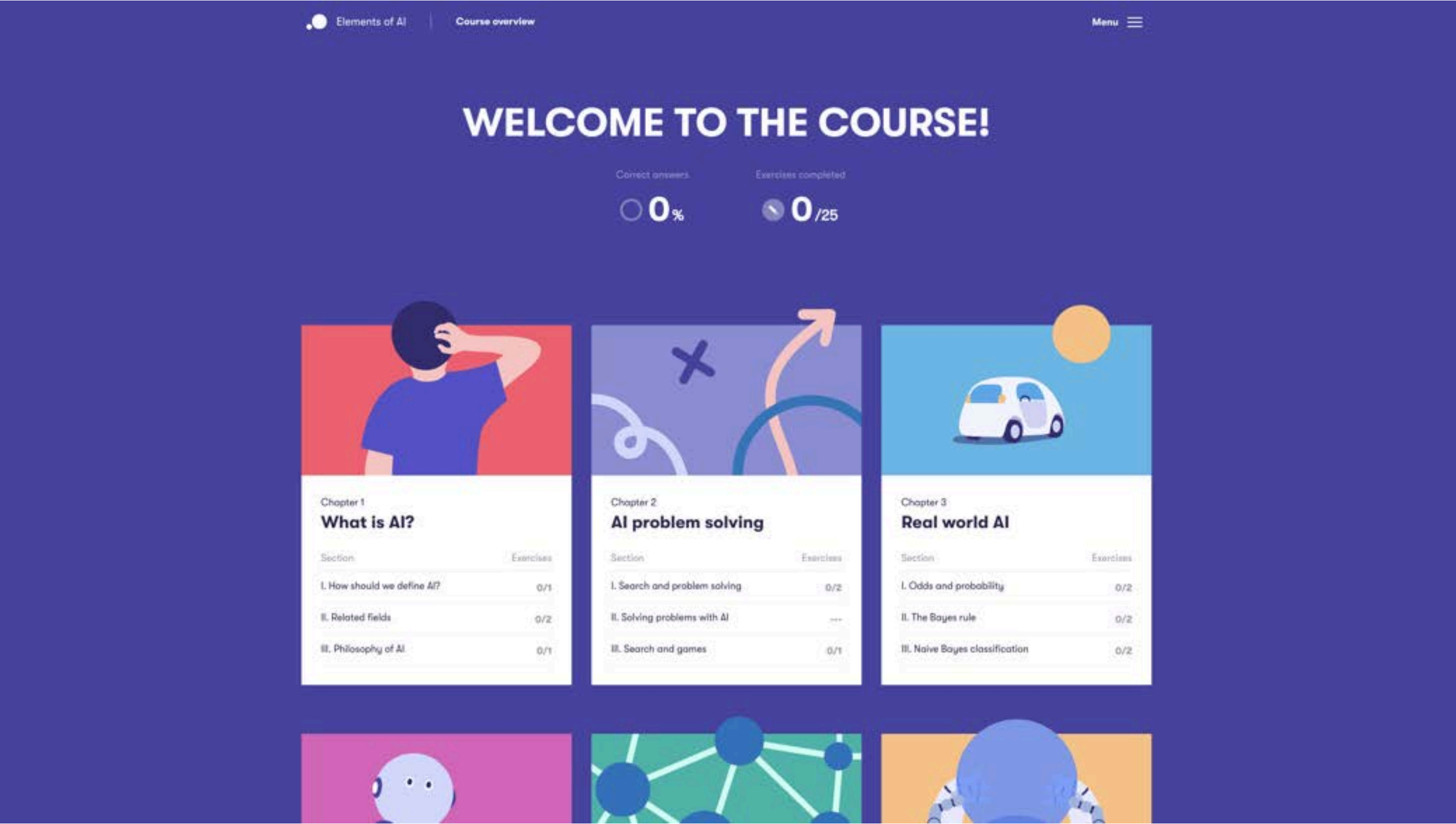
- Parse CI logs and test output using NLP (e.g. sentence-transformers, clustering, embeddings).
- Auto-generated dashboards with anomaly detection, root cause suggestions, and recent regressions.
- AI-powered correlation across traces/API/UI → detects flaky tests.
- Structured logs = fuel for QA agents & regression-focused LLMs.

## QA Chatbot

- Answers team questions using internal QA documentation as a knowledge base (via RAG).



# How can I learn AI?



<https://www.elementsofai.com/>


o.dev

Search

English

# AI and the Web


The future of the web is now. Understand how AI and machine learning can be used by developers to build a more powerful web experience.



## What is AI?

Understand the basics and definitions of the various emerging technologies, often referred to as AI.


[Read the doc](#)



## Ethics and AI

As web practitioners, it's critical that we build new technology thoughtfully and responsibly.

[Read the doc](#)



## Upgrade your site search

Discover techniques to get contextual results from generative AI, at-scale.

[Read the doc](#)

### Practical tips

#### Understand LLM sizes

Take a look at a few real-world LLMs and the practical implications of different model sizes.

#### Compare model capabilities

Evaluate models and prompts with the LLM as a judge technique. Delegate validation to another LLM, instead of humans.

<https://web.dev/explore/ai?hl=en>



# Introduction to Generative AI - Art of the Possible

Digital training ENGLISH

## Course description

Languages Available: [Español \(Latinoamérica\)](#) | [Español \(España\)](#) | [Français](#) | [Bahasa Indonesia](#) | [Italiano](#) | [日本語](#) | [한국어](#) | [Português \(Brasil\)](#) | [ไทย](#) | [中文 \(簡體\)](#) | [Tiếng Việt](#)

The Introduction to Generative AI - Art of the Possible course provides an introduction to generative AI, use cases, risks and benefits. With the help of a content generation example, we illustrate the art of the possible. By the end of the course, learners should be able to describe the basics of generative AI, its risks and benefits. They should also be able to articulate how content generation can be used in their business.

- Course level: Beginner
- Duration: 1 hour

### Keywords

- Gen AI
- Generative AI


## Enrollment options

Free

[ENROLL](#)

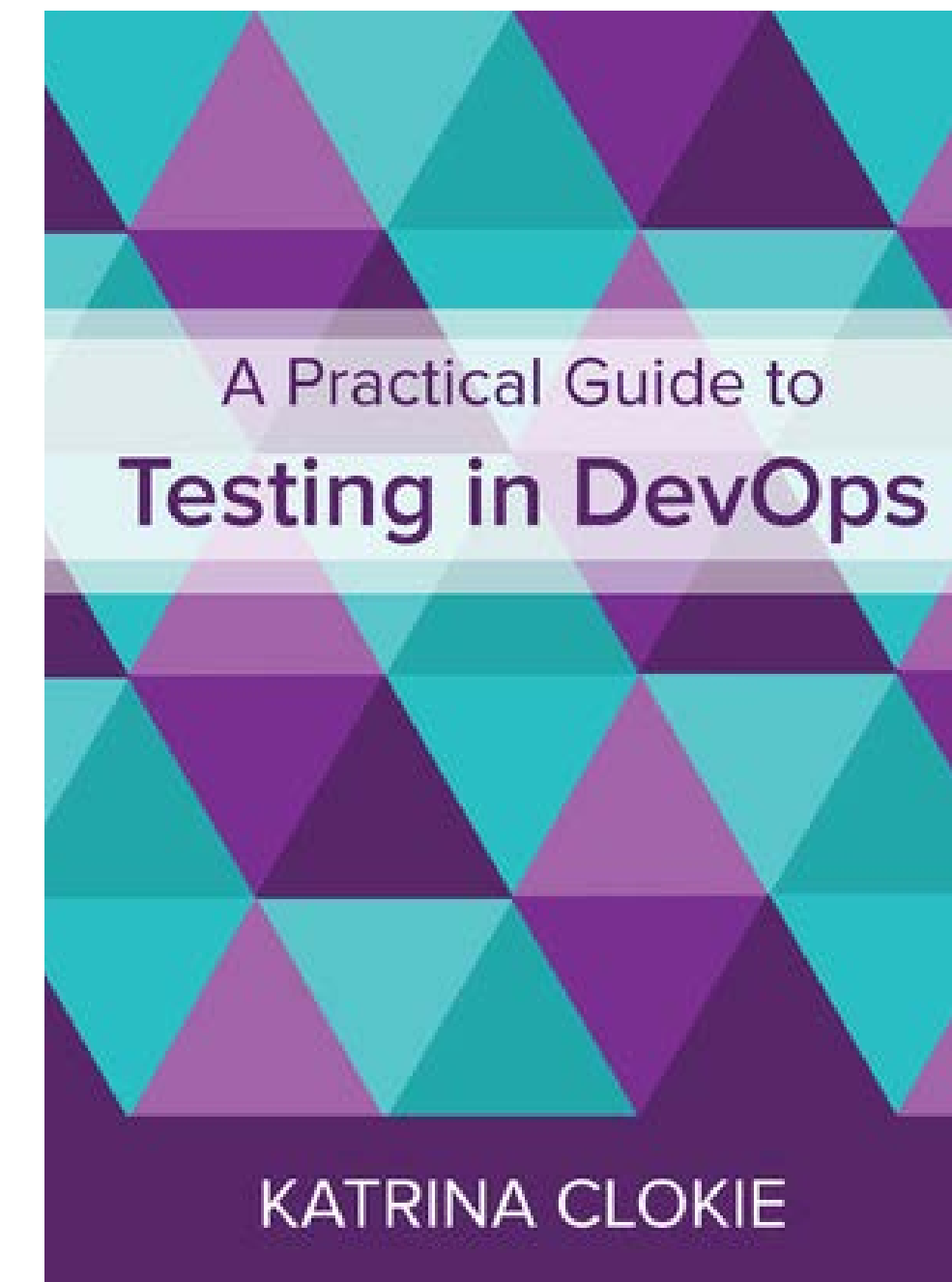
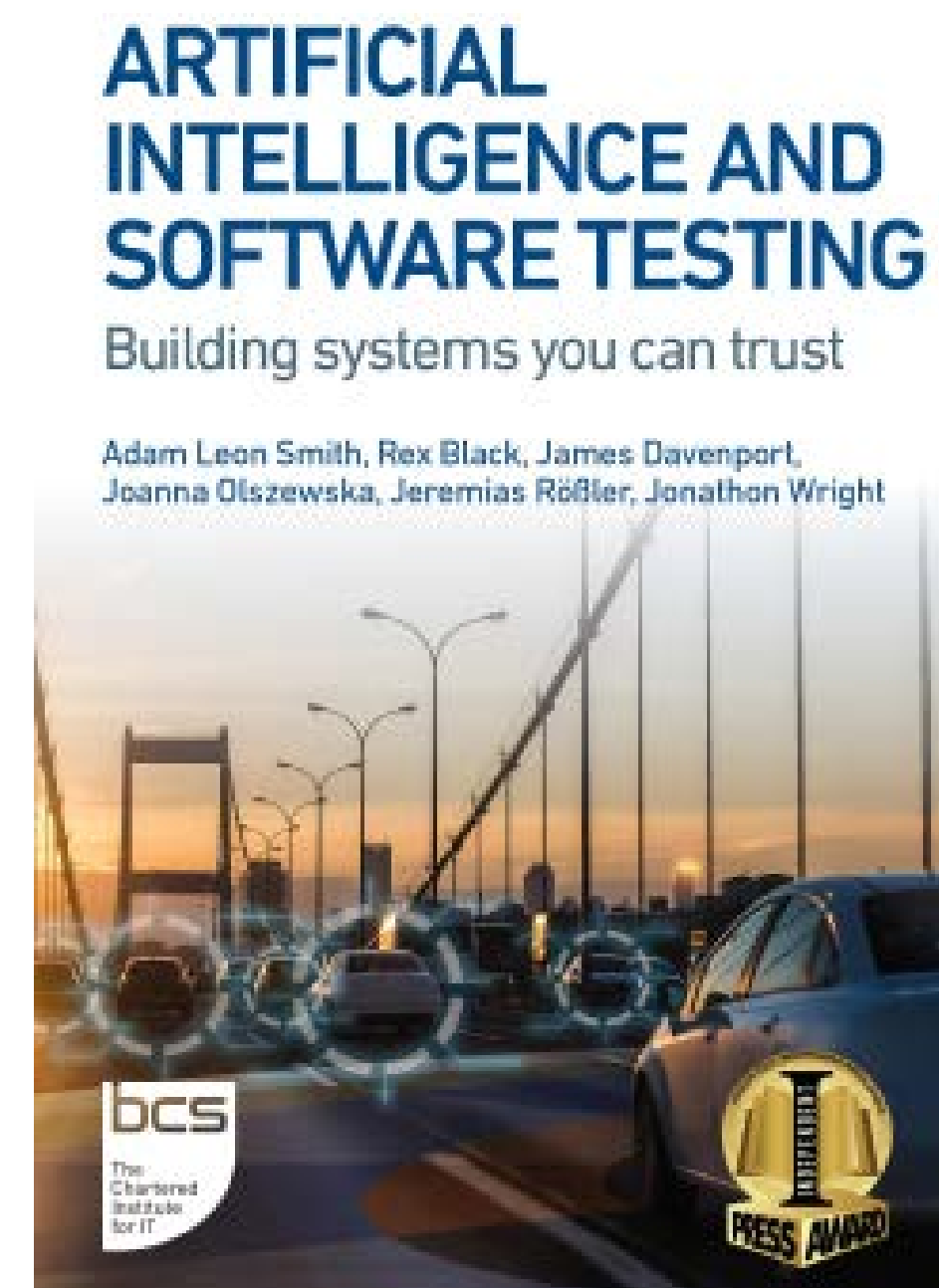
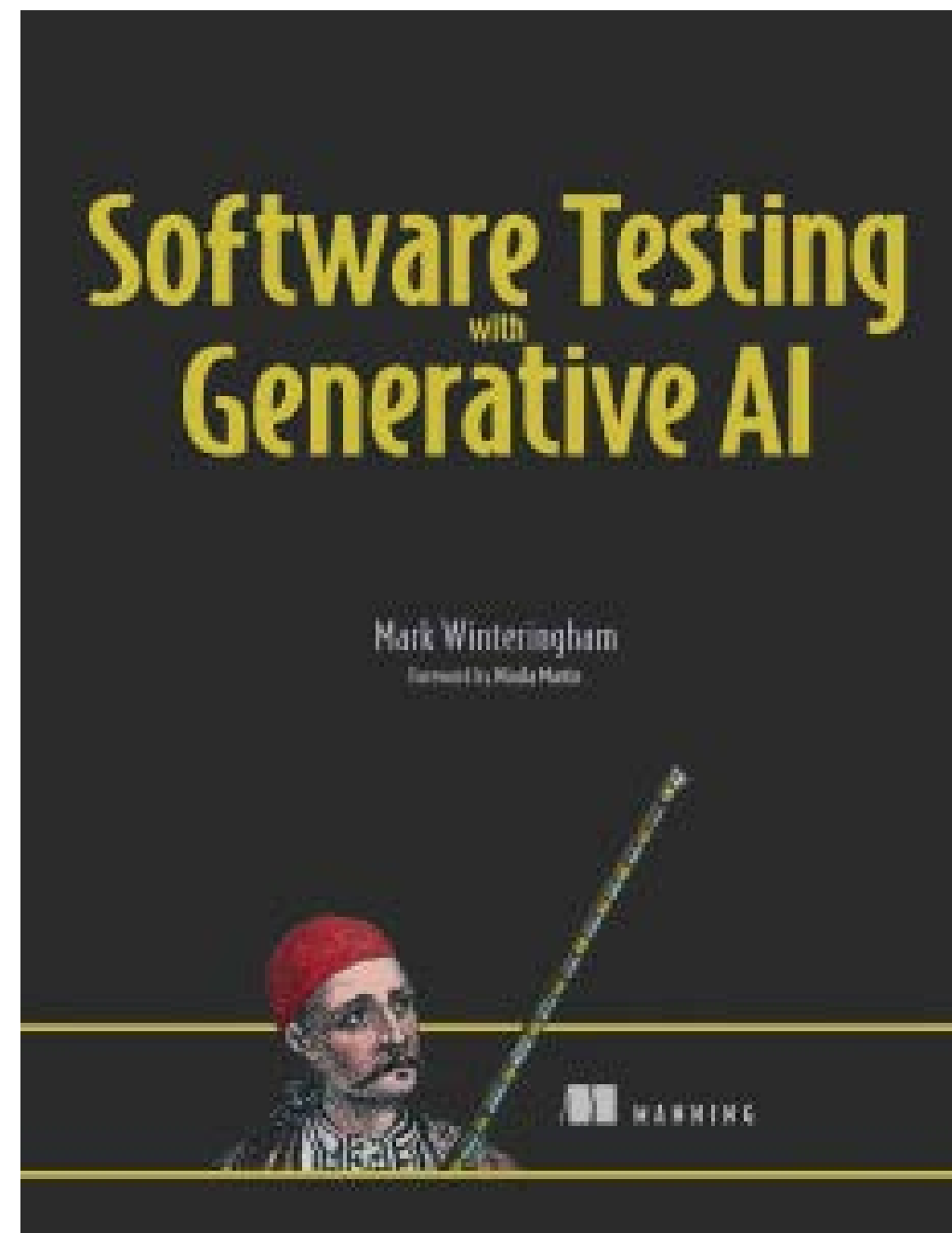
## Content

2 Lessons | 1h 00m

 Introduction to Generative AI - Art of the Possible

<https://explore.skillbuilder.aws/learn/courses/17176/introduction-to-generative-ai-art-of-the-possible>

@ludovicobesana



# How is testing evolving with AI?



# THE State of Software Quality 2025 REPORT



PRESENTED BY  
 Katalon

# 3 Take Aways



AI is an ally,  
not a shortcut



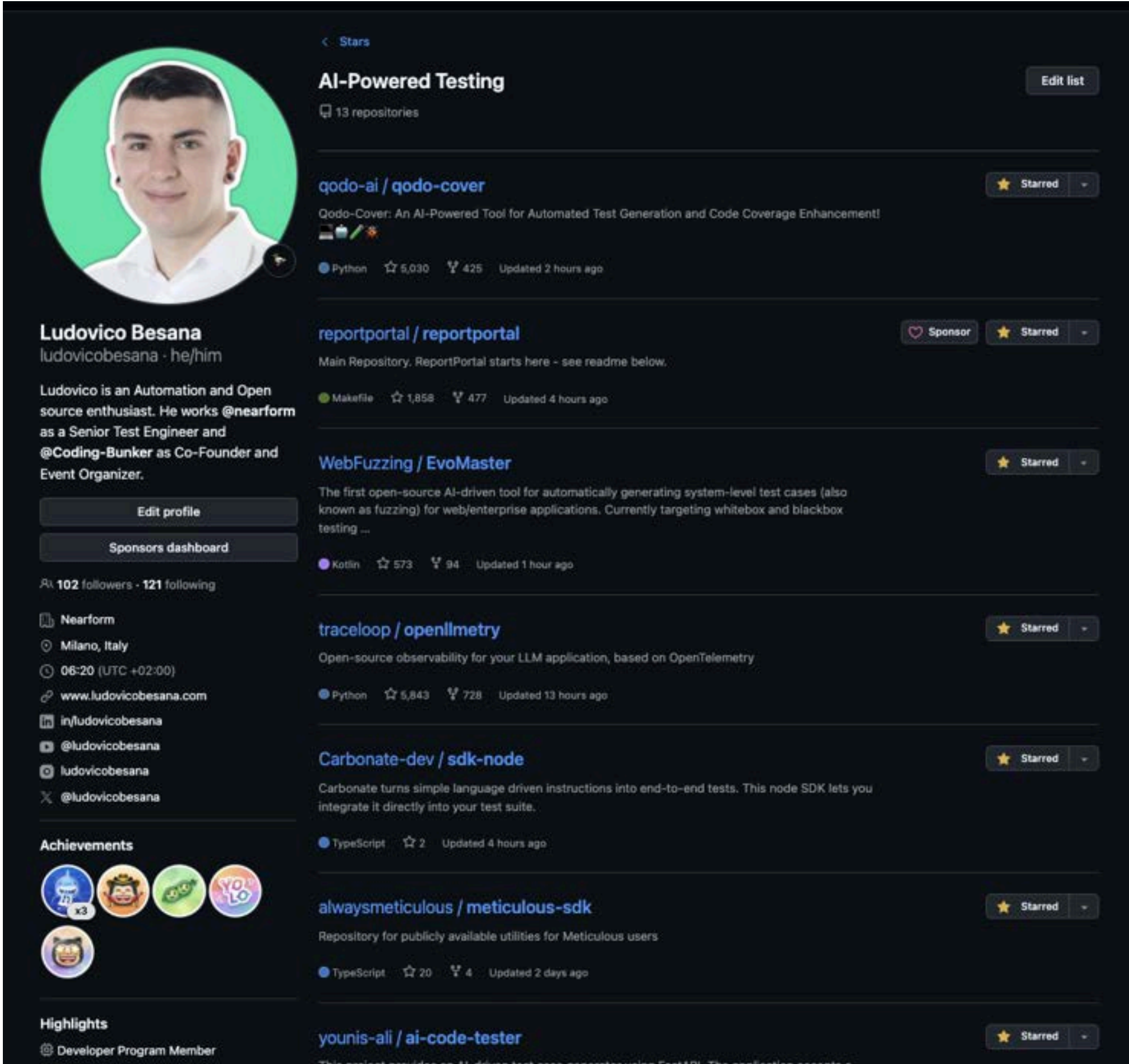
Start with small,  
measurable experiments



Value comes from context  
and asking the right questions

# Thank you

# Q&A



<https://github.com/ludovicobesana>



@ludovicobesana



# What's new

Starting today, you can follow me  
on YouTube too!

