AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Marc Balcer
Instructor

Format: Live Instructor-Led Online through Zoom
Date: June 8 - 10, 2026
Time: 12:00 PM - 4:30 PM EDT
Price: $650 per person

To register:
Email Chris Remmert cremmert@nysforum.org and indicate the course title in the subject line.

Technology and Attendance Requirements:
Computer with a browser, Zoom, a microphone and speaker. For this workshop, camera should be on if possible and you must be actively participating.

AI for Software Testing is a practical, hands-on course for experienced testers who want to integrate generative AI into real testing work—responsibly and effectively. Rather than treating AI as a testing replacement or a standalone skill, the course positions AI as a collaborator that supports testing judgment, not a substitute for it.

You will work with a small, realistic application and use AI throughout the course to explore how it can assist with understanding system behavior, generating and refining tests, structuring test ideas, exploring features, reporting defects, and making informed decisions about automation. AI output is treated as input for evaluation, not as authoritative answers.

Throughout the course, you will engage in realistic exercises that reflect how testers actually work: observing behavior, forming hypotheses, designing tests, and refining understanding as information emerges. You will use AI to generate test ideas and explanations, then examine those outputs for assumptions, gaps, executability, and risk. The emphasis is not on producing more tests faster, but on learning how to guide, question, refine, and integrate AI-generated material into sound testing work.

By the end of the course, you will have a grounded understanding of where AI adds value in testing, where it creates risk or noise, and how to maintain tester accountability while benefiting from AI's speed and flexibility. The course is designed for testers, test analysts, and test leads who want to adopt generative AI in a professional, experience-based way—improving decision-making without compromising rigor or responsibility.

Why this course

This course stands out because it teaches experienced testers how to use generative AI as part of real testing work, not as a shortcut or a novelty. Instead of focusing on AI theory, tool features, or certification checklists, it follows the natural flow of testing—from observing behavior and forming understanding, through test design, exploration, defect reporting, strategy, and automation—showing where AI helps, where it fails, and how testers stay in control.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

The emphasis is on judgment, evaluation, and accountability. You learn how to review AI-generated tests critically, uncover hidden assumptions, decide what is runnable and what is not, and integrate AI suggestions into coherent testing approaches. This makes the course directly applicable to day-to-day testing work in a way most AI training does not.

Certification

This course will contribute 14 PMI® Professional Development Units (PDUs) towards your chosen certification (14 Business Acumen).

An AI for Software Testing digital badge will be available upon successful completion of the course from SoftEd.

Great for:

- Testers, Test Analysts and Developers wanting to utilize AI to automate and assess testing tasks and artefacts
- Project Managers, Business Analysts and leaders wanting to accelerate the testing process whilst balancing responsible and ethical oversight
- Anyone looking to be skilled in AI augmentation and innovation

Learning Outcomes:

1. Use generative AI to explore and understand system behavior when documentation is incomplete or unclear.
2. Write prompts that describe observed behavior, constraints, and test intent clearly.
3. Identify assumptions, gaps, and invented details in AI-generated test cases.
4. Turn AI-generated test ideas into executable tests with clear steps and observable outcomes.
5. Structure tests using partitions, boundaries, state, and sequences with AI support.
6. Use AI to generate exploratory testing ideas without losing tester control or focus.
7. Evaluate AI-generated nonfunctional test ideas for relevance and testability.
8. Write clearer bug reports and evaluate AI-assisted summaries and metrics.
9. Make informed decisions about what to automate and what not to automate.
10. Create a practical plan for integrating generative AI into your own testing work.

Prerequisites

To get the most out of this course, it is recommended that participants have foundational knowledge of software testing through formal training like our Software Testing Foundations or Agile Testing course or have relevant experience working in a software testing context.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 1 — Understanding AI's Role in Software Testing

Begin by exploring what generative AI actually does and how it applies to software testing in practice. You will compare outputs from different AI tools, observe how confident-sounding responses can still be incomplete or incorrect, and examine how AI changes testing work—not by replacing tester judgment, but by accelerating certain activities. This module establishes AI as a collaborator whose output must be evaluated, challenged, and corrected.

Objectives

- Describe the typical capabilities and limitations of generative AI in software testing.
- Distinguish between plausible-sounding output and testable information.
- Evaluate AI responses for accuracy, completeness, and relevance.
- Reflect on how AI changes tester responsibilities and decision-making.

Exercise

Reflect on your role as a tester, including your core activities, the artifacts you create, the inputs you receive, and the different perspectives that best fit your work. Then explore how AI could support you in that role. This involves writing a clear prompt that describes your work, inputs, outputs, and perspective, then asking how AI might help. Compare responses from multiple chatbots and use follow-up questions to refine and deepen the results.

Module 2 — Let's Test!

Jump directly into test development by asking AI to generate test cases with minimal context. Attempt to execute those tests and discover where AI output helps, where it breaks down, and where assumptions have been silently introduced. By comparing results from different AI tools, you will learn to recognize the difference between tests that appear reasonable and tests that are actually executable and grounded in observed behavior.

Objectives

- Observe how AI generates test cases with limited information.
- Identify hidden assumptions and invented details in AI-generated tests.
- Recognize the difference between test appearance and executability.
- Critique and compare test outputs from multiple AI tools.

Exercise

Starting with little more than an image of a UI page or a short general description have AI generate a series of test cases for that page.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 3 — Tests as Specifications

AI was able to generate tests from little more than a snapshot of a UI. In this module, you work in the opposite direction: using tests and observed behavior to infer requirements and business rules. You will examine how AI fills gaps with assumptions, learn to separate confirmed behavior from speculation, and use requirement taxonomies to organize and correct AI-generated statements. The focus is on requirements as testable descriptions, not authoritative truths.

Objectives

- Infer requirements and business rules from observed system behavior.
- Separate confirmed behavior from assumptions and speculation.
- Identify and correct inaccuracies in AI-generated requirements.
- Organize requirements using established taxonomies to reveal gaps.
- Use AI to assist with traceability while maintaining tester judgment.

Exercise

Ask your AI chatbots to enumerate the requirements implied by the test cases. Are any of the requirements wrong or unclear? Prompt to correct them. Select one of the requirements classification taxonomies. Use AI to classify the requirements according to that taxonomy. Ask AI to construct a traceability matrix relating test cases to requirements.

Module 4 — Test Data

Real test cases require more than placeholder phrases such as “enter a valid value.” They require meaningful data choices that support coverage. In this module, you use equivalence partitioning and boundary value analysis as thinking tools to structure input domains and identify important conditions. AI is used to suggest partitions and boundaries, which you then evaluate for relevance, assumptions, and completeness before selecting representative data values.

Objectives

- Explain the role of test data in effective test coverage.
- Apply equivalence partitioning to group valid and invalid input conditions.
- Apply boundary value analysis to identify edge conditions.
- Distinguish between partitions based on observed behavior and those based on assumptions.
- Define preconditions and data dependencies explicitly.

Exercise

Have AI review and update the mileage calculator tests using equivalence partitioning and boundary value analysis. Define equivalence partitions for each input and identify boundary values for each partition. Have AI generate real data values for the inputs and identify any necessary pre-existing data.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 5 — Making Tests Executable

What does it actually mean for a test case to be executable? This module focuses on the information required for another tester to run a test without interpretation. Use AI to refine test cases to include clear steps, concrete data, and observable outcomes. AI is used to suggest improvements and rewrites, but emphasis is placed on recognizing what AI cannot infer and where human clarification is required.

Objectives

- Identify the information required to execute a test reliably.
- Rewrite test cases to remove ambiguity and interpretation.
- Define observable outcomes that support clear pass/fail decisions.
- Evaluate AI-generated edits for completeness and realism.
- Recognize details that AI cannot infer without domain knowledge.

Exercise

Have your chatbots review the test cases generated in the last exercise to evaluate whether or not they are clear and consistent enough to be run without interpretation. Ask AI to improve the tests by adding details and clear instructions. Then have AI translate these tests into Gherkin Given-When-Then notation.

Module 6 — Stories and Scenarios

User stories describe changes to a system, and acceptance criteria define the conditions under which those changes are considered complete. In this module, you examine how acceptance criteria, scenarios, and test cases relate in practice. You will explore the many-to-many relationship between them and use AI to draft and refine acceptance criteria while ensuring that scenarios remain grounded in realistic system behavior.

Objectives

- Review the purpose and structure of user stories.
- Understand the relationship between acceptance criteria, scenarios, and test cases.
- Recognize that acceptance criteria and scenarios map many-to-many.
- Use AI to draft acceptance criteria and scenarios critically.
- Identify gaps and overlaps between stories, criteria, and tests.

Exercise

Given a set of user stories, collaborate with AI to define the acceptance criteria for these stories. Then have AI write (new or enhanced) scenarios and test cases for these stories. Try testing each story independently and in combination. Finally, have AI create a mapping table between stories, acceptance criteria, and scenarios / test cases.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 7 — States and Coverage

Although tests for single pages and functions are a good start and can catch a lot of bugs, eventually we'll need to test larger units of functionality. These can be abstracted as different states and visualized graphically. Learn state-based thinking as a way to model sequences, transitions, and cumulative effects. Use AI to help sketch state diagrams and identify valid transitions, then reason about functional coverage in terms of states, transitions, and invariants rather than test counts.

Objectives

- Understand the importance of state and sequence in system behavior.
- Model system behavior using states and transitions.
- Identify valid and invalid transitions between states.
- Reason about functional coverage using states, transitions, and invariants.
- Use AI to assist with modeling while validating accuracy manually.

Exercise

Use your chatbots to create a state model illustrating the different states of the sample application and the behaviors that cause transition from page to page. Use AI to determine if the test cases from the last exercise fully cover the paths through that state model. If not, what tests need to be added? As a bonus, have AI generate a UI navigation map for an extended version of the application.

Module 8 — Validating Quality Attributes

Functional correctness is not enough. Systems must also meet quality expectations such as usability, accessibility, performance, and security. Use AI to evaluate nonfunctional requirements and to convert these from vague aspirations into solid testable objectives. Design tests that ensure systems meet quality criteria essential for user satisfaction and system reliability

Objectives

- Identify relevant nonfunctional requirements for a given system.
- Distinguish testable quality attributes from vague or non-actionable ones.
- Evaluate AI-generated nonfunctional test ideas for relevance.
- Prioritize quality attributes based on risk and impact.
- Design tests that validate selected quality attributes.

Exercise

Use AI to improve loosely-defined nonfunctional requirements, make them testable, and identify techniques for testing these requirements.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 9 — Test Strategy and Planning

Individual tests do not add up to a test strategy. Learn how AI can help you to make deliberate testing choices aligned with business goals and risk. Use a Test Strategy canvas to outline and plan a comprehensive testing strategy. See how AI can draft inputs and challenge assumptions, while responsibility for prioritization and trade-offs remains with the tester.

Objectives

- Distinguish between test cases and test strategy.
- Identify key risks and priorities that drive testing decisions.
- Use planning frameworks to structure testing efforts.
- Evaluate AI-generated strategy inputs critically.
- Align test strategy with business and technical goals.

Exercise

Use AI to create a Test Strategy Canvas—a single concise view of the overall approach to testing a full application.

Module 10 — Bug Analysis and Reporting

Bug reports are both technical findings and communication artifacts. Practice using AI to analyze bug reports to distinguish real failures from expected behavior or misunderstandings, assess impact and risk, and communicate findings effectively to different audiences. Although AI is used to assist with summarization and categorization, learn to evaluate where this help improves clarity and where it distorts meaning or priority.

Objectives

- Distinguish bugs from misunderstandings of expected behavior and feature changes.
- Analyze bugs to identify impact, risk, and root causes.
- Evaluate AI-generated bug summaries and categorizations critically.
- Decide when AI-assisted analysis adds value and when it does not.
- Write clear, actionable bug reports for different stakeholders.

Exercise

Ask AI chatbots to come up with list of potential bugs, different kinds of risks, and a set of sample bug reports. Then take a list of actual bug reports and have the AI analyze and classify these into an actionable report.

AI for Software Testing

14 PMI PDUs | 14 IIBA CDUs

Content:

Module 11 — Test Automation

Automating test execution promises to make it easier and cheaper to run large test suites more frequently—a practice essential to modern agile development and DevOps practices.
Explore the different kinds of automated tests. See how AI can generate automated tests for different levels and architectures. Use AI to evaluate existing systems' suitability for automation and to check and improve existing automation suites.

Objectives

- Evaluate whether a test scenario is suitable for automation.
- Identify prerequisites for effective automation that AI cannot infer.
- Critically assess AI-generated automation scripts.
- Recognize common automation anti-patterns and risks.
- Decide when manual or exploratory testing is the better option.

Exercise

While no programming or automation tool experience is necessary, get a sense of the power of automated testing by observing demonstrations of different kinds of automation. Ask AI chatbots for advice about automating different test scenarios.

Module 12 – Integrating AI Into Your Testing

So what does all of this mean? Now it all comes together. You'll assess where AI can add the most value in your own practice, plan small experiments, and design an ethical framework for responsible adoption. The focus is on practical next steps—how to pilot, measure, and lead change as an AI-empowered tester.

Objectives

- Identify high-potential areas to pilot AI within your testing processes.
- Plan change-management and measurement strategies for AI adoption.
- Develop an ethical framework for responsible use of AI tools and data.
- Create a personal or team roadmap for integrating AI as a productivity and quality multiplier.

Exercise

Reflect on your own testing practice and identify areas where AI could realistically add value. Use AI to brainstorm small, low-risk pilot experiments and success measures.

Lecturing is kept to the minimum necessary where most of the learning is achieved by applying the practices and techniques in group exercises. Our LiveOnline delivery is over three days (each four and a half hours in duration). The instructor is 100% live and interaction and learning objectives are the same as our in-person classes with the added benefit of being able to take this course from your home, your office or your home office. Since this class is delivered over half-days it allows for greater flexibility and leaves you with time each day for other work or activities.