

A Survey on Web Testing: on the Rise of AI and Applications in Industry

Karthik Kapula
UiPath Solution Architect, ICS GLOBAL SOFT INC, Little Elm, Texas

ARTICLE INFO

Received: 12 Oct 2024
Revised: 18 Dec 2024
Accepted: 25 Dec 2024

ABSTRACT

Web testing is a critical component of software quality assurance, ensuring functionality, usability, and reliability of increasingly complex web applications. The integration of Artificial Intelligence (AI) techniques such as machine learning, natural language processing, and computer vision has significantly advanced the field by automating test generation, self-healing scripts, anomaly detection, and test optimization. This survey explores the current landscape of AI-powered web testing approaches, tools, and frameworks widely adopted across industries including e-commerce, finance, and healthcare in 2024. We analyze popular commercial and open-source solutions, discuss the challenges of dynamic web environments and flaky tests, and highlight emerging trends such as autonomous testing agents, multimodal testing, federated learning, and the need for standardization. Finally, the survey emphasizes the importance of collaboration between academia and industry to accelerate innovation and practical deployment of intelligent web testing methodologies.

Keywords: Web Testing, Artificial Intelligence (AI), Machine Learning, Natural Language Processing (NLP), Anomaly Detection, Autonomous Testing Agents, Federated Learning, Software Quality Assurance, Test Optimization

1. INTRODUCTION

Web applications have become ubiquitous, supporting critical functionalities in domains ranging from e-commerce and finance to healthcare and education. As their complexity and scale increase, ensuring their reliability, performance, and security is more challenging than ever. Web testing—encompassing the verification and validation of web-based systems—plays a central role in guaranteeing the quality of these applications. However, traditional web testing techniques often struggle to keep up with the dynamic nature of modern web development practices, particularly in agile and DevOps environments. In recent years, Artificial Intelligence (AI) has emerged as a transformative force in software testing, offering promising capabilities such as intelligent test generation, predictive defect analysis, and self-healing automation scripts. This survey explores the integration of AI into web testing, highlighting its current state, applications in industry, and emerging research directions.

1.1 Motivation

The traditional landscape of web testing is labor-intensive, error-prone, and often fails to scale with the growing complexity of web applications. Manual scripting and static test case design struggle to keep up with rapid development cycles and continuous deployment workflows. Additionally, maintaining test suites in the face of frequent UI changes is time-consuming and costly.

AI techniques offer a way to alleviate these challenges by introducing intelligence, adaptability, and automation to the testing lifecycle. The motivation behind this survey is to consolidate knowledge on how AI is transforming web testing and to provide a comprehensive understanding of its applications, benefits, limitations, and future potential. Given the increasing adoption of AI-driven tools in 2024, a timely survey is necessary to guide practitioners, researchers, and tool developers alike.

1.2 Scope and Objectives of the Survey

This survey focuses on the intersection of web testing and AI, particularly exploring developments up to the year 2024. The main objectives are:

- To review the foundational concepts of traditional web testing.
- To examine AI techniques relevant to web testing, such as machine learning, NLP, and computer vision.
- To survey existing AI-powered approaches and tools for web testing, including their applications and performance in real-world scenarios.
- To analyze case studies of industrial adoption across different sectors.
- To identify current limitations, emerging challenges, and promising directions for future research.

While the primary emphasis is on functional testing (e.g., UI, regression, test automation), non-functional aspects such as performance and security are also briefly discussed where relevant to AI applications.

1.3 Methodology

This survey was conducted through a systematic literature review of academic publications, white papers, and technical blogs from 2018 to 2024. Key sources include IEEE Xplore, ACM Digital Library, SpringerLink, and arXiv, as well as industrial documentation and product releases from AI-based testing platforms.

The selection criteria for included studies were based on relevance to AI applications in web testing, novelty of approach, and demonstrated impact or adoption in industrial settings. In addition to peer-reviewed research, the survey integrates insights from real-world implementations and practitioner feedback where available.

Tools were evaluated using publicly available documentation, feature matrices, and performance reports. Case studies were selected to represent a diverse range of industries and testing scenarios.

2. BACKGROUND ON WEB TESTING

Web testing has evolved in parallel with the growth and transformation of web applications. From static HTML pages to complex single-page applications (SPAs) and progressive web apps (PWAs), the demand for rigorous, continuous testing has increased exponentially. This section outlines the evolution of web applications, summarizes traditional web testing techniques, and discusses the key challenges associated with modern web testing environments.

2.1 Evolution of Web Applications

The landscape of web applications has undergone several major transitions over the past two decades:

- **Static Web (Web 1.0):** The early web consisted of static HTML pages with limited interactivity. Testing was mostly manual and involved verifying links, forms, and basic layout compatibility.
- **Dynamic Web (Web 2.0):** With the rise of JavaScript, AJAX, and server-side scripting (e.g., PHP, ASP.NET), web applications became more dynamic and interactive. This necessitated functional testing, load testing, and cross-browser compatibility assessments.

- **Single-Page Applications (SPAs):** Frameworks like Angular, React, and Vue.js enabled the development of SPAs, where content updates dynamically without full page reloads. Testing SPAs introduced new complexities, such as asynchronous operations, DOM state tracking, and component-level testing.
- **Progressive Web Apps (PWAs):** PWAs combine the best of web and mobile apps, offering offline capabilities, push notifications, and installable interfaces. Testing now had to include service worker behavior, cache strategies, and mobile responsiveness.
- **Microservices and API-Centric Architectures:** Modern web apps often rely on distributed services accessed via REST or GraphQL APIs. Web testing must validate the integration between front-end and back-end services, making API testing and end-to-end testing essential.
- **DevOps and CI/CD Pipelines:** Continuous integration and deployment have transformed how applications are developed and released. Web testing is now integrated into automated pipelines and expected to deliver rapid feedback.

2.2 Traditional Web Testing Techniques

Web testing encompasses several types of testing activities, traditionally categorized as follows:

- **Functional Testing:** Ensures that features behave as expected, including form submissions, button actions, navigation, and input validation.
- **Regression Testing:** Checks that new changes do not break existing functionality. Regression suites are often maintained manually or via scripted automation.
- **Cross-Browser and Cross-Platform Testing:** Verifies consistency across different browsers (Chrome, Firefox, Safari, Edge) and platforms (Windows, macOS, Android, iOS).
- **Performance and Load Testing:** Measures system behavior under expected and peak traffic conditions using tools like Apache JMeter or LoadRunner.
- **Security Testing:** Identifies vulnerabilities such as XSS, CSRF, SQL injection, and insecure authentication flows.
- **Accessibility Testing:** Ensures compliance with accessibility standards (e.g., WCAG) to support users with disabilities.
- **UI Testing:** Verifies visual elements, layout correctness, and usability using automated tools or manual inspection.

Automation tools like Selenium, Cypress, and TestNG have long supported web testing, particularly for regression and functional tests. However, these tools often require high maintenance, especially when UI changes are frequent.

2.3 Challenges in Web Testing

Despite the maturity of web testing practices, several persistent challenges continue to hinder effectiveness and efficiency:

- **Test Script Fragility:** Automated scripts, particularly UI-based ones, are brittle and prone to failure with minor UI changes (e.g., altered DOM structure or element identifiers).

- **Scalability Issues:** As applications scale, so do the number of test cases. Managing large test suites becomes resource-intensive and often results in slower feedback loops.
- **Maintenance Overhead:** Frequent updates in agile and DevOps cycles require constant updates to test cases, increasing costs and reducing team productivity.
- **Incomplete Test Coverage:** Ensuring comprehensive coverage—both functional and non-functional—is difficult, especially when dealing with dynamic client-side rendering and third-party integrations.
- **Cross-Device and Cross-Browser Variability:** Ensuring consistent behavior across environments remains a significant challenge due to differences in rendering engines, device capabilities, and network conditions.
- **Asynchronous and Event-Driven Behavior:** Testing modern SPAs requires handling complex event sequences and race conditions that are hard to model with traditional tools.
- **Lack of Intelligence:** Traditional tools lack the ability to learn from past executions, predict defects, or adapt to changes autonomously—highlighting the need for AI integration.

These challenges have paved the way for AI-based enhancements in web testing, where learning algorithms and intelligent automation can augment traditional approaches. The next section delves into how AI is transforming the web testing landscape.

3. AI IN WEB TESTING: FOUNDATIONS AND TECHNIQUES

Artificial Intelligence (AI), especially in the form of Machine Learning (ML), is increasingly being embedded into web testing workflows to improve automation, adaptability, and insight generation. Unlike traditional rule-based systems, AI enables predictive and data-driven decision-making, making web testing processes more resilient and intelligent. This section explores the foundational role of AI and ML in web testing, highlights core techniques used, and discusses how modern tools are leveraging these advancements.

3.1 Role of AI and Machine Learning

AI brings a paradigm shift to web testing by enabling systems to learn from data, adapt to changes, and even make autonomous decisions. In a typical testing pipeline, AI can be applied to prioritize test cases, detect UI anomalies, predict failures, generate scripts, and heal broken automation flows. Machine Learning models, trained on historical test data, logs, and defect reports, can uncover hidden patterns that indicate where defects are likely to occur or which tests provide the highest value. This predictive power is particularly valuable in continuous integration and deployment (CI/CD) environments where speed and quality must go hand in hand. Furthermore, AI reduces human intervention in routine tasks, making testing more scalable and aligned with agile development cycles.

3.2 Key AI Techniques for Web Testing

3.2.1 Supervised Learning for Bug Prediction

Supervised learning has been widely applied in software quality assurance to build predictive models that can classify or rank software components based on their likelihood of containing defects. In web testing, historical data from bug reports, code repositories, and execution logs are used to train classifiers such as decision trees, support vector machines, or neural networks. These models can identify high-risk modules or web components, enabling test teams to focus their efforts where they

matter most. For instance, a supervised model might predict that a new feature in a React component has a high chance of regression due to its dependency on a previously unstable module. These insights improve test coverage and reduce time spent on unnecessary tests.

3.2.2 Unsupervised Learning for Test Clustering

Unsupervised learning, particularly clustering algorithms like k-means or DBSCAN, is useful when labeled training data is scarce or unavailable. In web testing, unsupervised learning can be used to group similar test cases based on execution paths, feature coverage, or UI interactions. This helps in identifying redundant test cases, optimizing test suites, and improving execution efficiency. For example, clustering can reveal that a set of functional tests all interact with the same backend API endpoint, allowing for batch execution or prioritized monitoring. Additionally, anomaly detection—an unsupervised technique—can be applied to flag unusual behavior in test results or system logs, helping identify issues early.

3.2.3 Reinforcement Learning for Test Case Generation

Reinforcement Learning (RL) offers a powerful framework for dynamic and exploratory test case generation. In this setting, a test agent learns optimal actions (e.g., clicking buttons, entering data, navigating pages) through trial and error to maximize a reward signal such as code coverage or bug discovery. Over time, the RL agent becomes more proficient at generating test sequences that expose edge cases or unstable states. This is particularly valuable in exploratory testing of SPAs or systems with rich client-side interactions, where conventional test scripts often fall short. RL-based approaches can also adapt to UI changes and new workflows without human reprogramming, increasing test resilience.

3.2.4 NLP for Test Case Understanding and Generation

Natural Language Processing (NLP) plays a critical role in bridging human-readable test cases with machine-executable scripts. NLP techniques are used to interpret test requirements written in natural language and convert them into structured, executable formats using semantic parsing, intent recognition, and sequence-to-sequence models. Conversely, NLP can also be employed to summarize test outcomes or auto-generate documentation. Advanced models, including large language models (LLMs), can now generate Selenium or Cypress scripts directly from user stories or acceptance criteria. This not only accelerates test creation but also empowers non-technical stakeholders to contribute to testing by specifying behaviors in plain language.

3.3 AI-Augmented Test Automation Tools

Modern web testing tools are increasingly incorporating AI modules to enhance automation capabilities. Tools like **Testim**, **Mabl**, **Functionize**, and **Applitools** offer features such as self-healing test scripts, visual validation through computer vision, and predictive test analytics. Self-healing scripts use AI to automatically update locators or test steps when UI changes are detected, reducing maintenance overhead. Visual AI tools compare UI renderings at a pixel or DOM level, identifying subtle regressions that traditional functional tests might miss. Additionally, AI-based tools can prioritize tests based on historical failure patterns, user behavior analytics, or code change impact, thereby optimizing the execution schedule within CI pipelines. These capabilities illustrate how AI is not replacing testers but augmenting their efficiency, enabling teams to deliver faster and with higher confidence.

4. SURVEY OF AI-POWERED WEB TESTING APPROACHES

As AI technologies mature, their integration into web testing workflows has given rise to innovative techniques that extend beyond traditional automation. This section surveys six major categories of AI-

powered approaches that are transforming how web applications are tested in both research and industry.

4.1 Model-Based Testing with AI

Model-Based Testing (MBT) involves generating test cases from abstract models that represent the desired behavior of the system under test. Traditional MBT requires manual effort to create and maintain these models, which can be time-consuming and error-prone. AI enhances MBT by automating model creation and refinement using data-driven methods. For instance, machine learning algorithms can infer usage models from user interaction logs or code repositories. These inferred models can then be used to generate diverse and realistic test scenarios. AI also aids in updating models dynamically as the application evolves, ensuring that the test coverage remains aligned with the current system behavior. This fusion of AI and MBT leads to more adaptive and scalable testing strategies.

4.2 Visual Testing Using Computer Vision

Visual testing focuses on verifying the correctness of the user interface by comparing expected and actual visual outputs. Traditional pixel-by-pixel comparison methods are brittle and fail to handle minor, non-breaking UI changes. AI-powered visual testing uses computer vision and deep learning techniques to intelligently compare UI elements at a semantic level. Tools like AppliTools leverage convolutional neural networks (CNNs) to detect visual anomalies that are perceptible to users—such as misaligned elements, font changes, or color mismatches—while ignoring irrelevant differences like rendering quirks. This approach enhances the accuracy and robustness of UI validation, particularly across different browsers and screen resolutions. Moreover, AI can learn which visual changes are significant based on historical acceptance or rejection patterns, further reducing false positives.

4.3 Intelligent Test Case Generation

One of the most impactful uses of AI in web testing is the automated generation of test cases. Traditional approaches often require manual scripting or hard-coded templates, which are not scalable for large or rapidly changing applications. AI models—particularly those based on reinforcement learning, NLP, and genetic algorithms—can analyze application structure, user behavior data, and past test results to generate relevant and diverse test cases automatically. For example, a reinforcement learning agent might explore the UI to identify hidden or less-traveled paths, while an NLP model could translate user stories into executable test scenarios. These intelligent generation techniques not only accelerate the test design phase but also help uncover edge cases that might be missed by manual testers.

4.4 Self-Healing Test Scripts





In dynamic web environments, test scripts frequently break due to minor UI changes such as renamed buttons or altered element hierarchies. AI-powered self-healing mechanisms address this problem by automatically identifying and repairing broken selectors at runtime. These systems use AI models trained on historical locator data and DOM structure patterns to suggest alternative element locators when a test fails. For example, if an XPath expression fails to find a button due to a class name change, the system may fall back on other attributes such as label text, hierarchy, or visual position. Self-healing not only reduces the maintenance burden of automated tests but also enhances reliability in CI/CD workflows, where frequent updates are the norm.

4.5 Anomaly Detection in Test Results

Analyzing test results, especially in large-scale testing environments, can be tedious and time-consuming. AI-based anomaly detection techniques automate this process by learning the normal patterns of test execution and identifying outliers or deviations. These techniques, often based on unsupervised learning or time-series analysis, flag abnormal test durations, flaky test behaviors, or unexpected error messages that may not lead to test failures but indicate potential issues. For instance, a significant increase in the response time of a frequently tested API might be flagged as an anomaly even if the test technically passes. Such insights help teams detect regressions earlier, diagnose root causes faster, and maintain higher system quality over time.

4.6 Test Optimization and Prioritization Using AI

Given limited resources and tight release cycles, executing every test in the suite is often impractical. AI-driven test optimization and prioritization techniques help by selecting the most relevant tests based on risk, impact, and historical effectiveness. These systems use machine learning to analyze code changes, test coverage data, and historical defect trends to predict which tests are most likely to uncover bugs. Prioritization models can rank test cases in real-time as changes are committed, ensuring that high-risk areas are tested first. This leads to faster feedback, reduced test execution time, and more efficient use of testing infrastructure. In some implementations, AI can also suggest which tests can be safely skipped or deferred, enabling smarter regression testing strategies.

				
Platform	Web	Web & Mobile	Web	Web & Mobile
Purpose	Test Maintenance	Test Maintenance	Test Maintenance and Creation	Test Maintenance and Generation
Strength	Automated Testing with AI	Applications Involving Visual Testing	Synthetic Monitoring and AI for Smooth Change Adaption	Functional, Performance, and Load Testing
Free version availability	Yes (Community version)	Yes (Starter Edition)	Yes (Free Trial on request)	No

© Emtec, Inc.

5. TOOLS AND FRAMEWORKS IN 2024

The growing integration of AI into web testing has led to the emergence of several powerful tools—both commercial and open source—that enhance test efficiency, reliability, and adaptability. This section surveys leading AI-powered tools available in 2024, covering both industrial solutions and academic prototypes, and concludes with key criteria for evaluating these tools in real-world scenarios.

Tool	AI Features	Main Strength	User Focus	Deployment
Testim	Smart locators, self-healing	Reliable test maintenance	Agile/DevOps teams	Cloud / On-premises
Mabl	Visual AI, anomaly detection	Easy, low-code UI testing	QA & non-technical	Cloud
Functionize	NLP scripting, SmartFix	Scalable, natural language	Enterprises	Cloud
Applitools	Visual AI, Ultrafast Grid	Industry-leading visual UI	UI/UX and automation	Cloud / On-premises
Open Source / Academic	Genetic algorithms, RL	Research innovation	Researchers/Innovators	Mostly on-premises

5.1 Comparison of Popular AI-Powered Testing Tools

5.1.1 Testim

Testim is a widely adopted commercial test automation platform that leverages machine learning to enhance authoring, execution, and maintenance of UI tests. Its standout feature is **smart locators**, which dynamically identify elements based on multiple attributes and visual cues, making test scripts more resilient to UI changes. Testim also supports **self-healing** tests and **auto-grouping** of test steps to simplify modular test design. Its integration with CI/CD pipelines and test case management tools makes it ideal for agile and DevOps environments. As of 2024, Testim also includes AI-based **test prioritization**, allowing teams to optimize regression cycles based on code changes and test impact analysis.

5.1.2 Mabl

Mabl is another leading cloud-based AI testing platform that focuses on **intelligent end-to-end testing**. It uses machine learning to track application behavior across multiple test runs and identify anomalies. Mabl's **auto-healing** capabilities reduce test maintenance by adapting scripts to UI changes, while its **visual change detection** helps in identifying UI regressions. It provides deep integration with CI/CD tools, Jira, and Slack for seamless reporting. Unique to Mabl is its **low-code test creation environment**, enabling non-technical team members to contribute to automated testing. The platform's AI also helps with **flaky test identification** and performance trend analysis over time.

5.1.3 Functionize

Functionize offers a sophisticated AI-driven testing suite that combines **natural language processing (NLP)** and **machine learning** to simplify test creation and enhance scalability. Testers can write test cases in plain English, which the platform converts into executable tests using NLP models. Functionize's **adaptive test execution engine** automatically adjusts test flows in response to minor UI or data changes. The tool also includes **SmartFix**, which uses AI to recommend and apply fixes to broken test steps. In 2024, Functionize is recognized for its enterprise-grade scalability, supporting large-scale test suites with minimal manual oversight.

5.1.4 Applitools

Applitools specializes in **visual AI testing**, offering a visual validation engine powered by advanced computer vision and machine learning. Its flagship product, the **Eyes** platform, uses **Visual AI** to detect perceptual differences in UI components across browsers, devices, and screen sizes, while filtering out insignificant rendering variances. Applitools also supports **auto-maintenance** of visual

baselines and integrates with Selenium, Cypress, and other automation tools. Its **Ultrafast Grid** enables parallel execution across virtual devices, accelerating visual regression testing. In 2024, Applitools remains the industry benchmark for pixel-perfect UI validation, especially in complex responsive web applications.

5.2 Open Source and Academic Tools

While commercial tools dominate industry adoption, a growing number of **open source** and **academic tools** are pushing the boundaries of AI-driven web testing. Tools like **SmashedAvocado**, **ReTest**, and **EvoMaster** explore AI techniques such as genetic algorithms, reinforcement learning, and unsupervised clustering for test case generation and GUI exploration. The **AutoFL** framework, for example, integrates fault localization with deep learning to assist in test failure analysis. Academic research prototypes often focus on **explainable AI in testing**, model inference, and traceability between requirements and test cases. Though many of these tools are not production-ready, they provide valuable insights into the future of intelligent automation and influence the design of next-generation industrial frameworks.

5.3 Tool Evaluation Criteria

Evaluating AI-powered testing tools requires a multi-faceted approach that considers not only functional features but also usability, integration, and long-term maintainability. Key evaluation criteria include:

- **Adaptability:** The tool’s ability to handle UI changes through self-healing, intelligent locators, and flexible model updates.
- **Ease of Use:** Availability of low-code/no-code interfaces, NLP support, and intuitive test authoring environments.
- **AI Capabilities:** Type and maturity of AI techniques used—e.g., visual AI, anomaly detection, learning-based prioritization.
- **Scalability:** Performance under large test suites, concurrent execution, and support for distributed/cloud-based testing.
- **Integration:** Compatibility with existing CI/CD pipelines, version control systems, and defect tracking tools.
- **Reporting and Analytics:** Quality of dashboards, root cause analysis features, and predictive insights from test runs.
- **Community and Support:** Availability of documentation, community forums, vendor support, and long-term updates.

As AI continues to influence tool development, these evaluation dimensions help teams choose platforms that align with their testing maturity, domain complexity, and automation goals.

Table 1 : Distribution of AI Techniques Used in Web Testing

AI Technique	Approximate Usage (%)
Machine Learning (Supervised/Unsupervised)	40%
Natural Language Processing (NLP)	25%

Computer Vision	20%
Reinforcement Learning	10%
Federated Learning	5%

Table 2 : Market Share of Popular AI-Powered Web Testing Tools (2024)

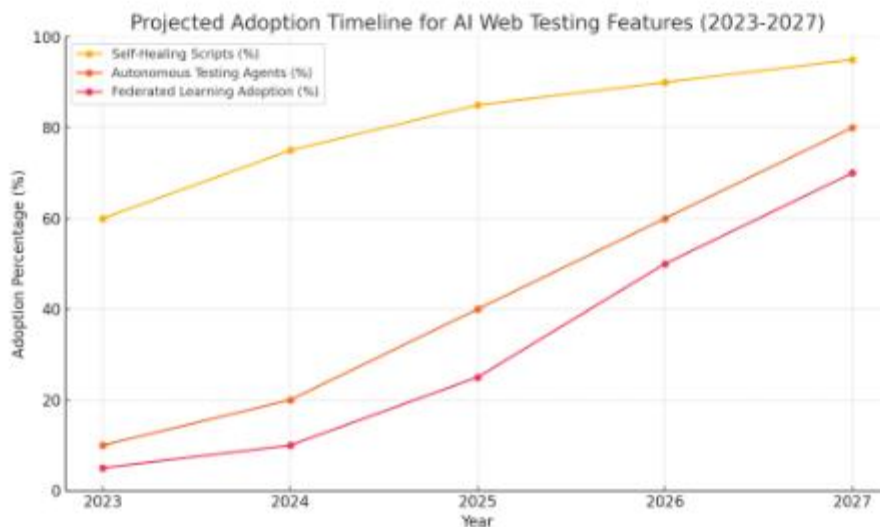
Tool	Market Share (%)
Testim	30%
Mabl	25%
Functionize	20%
Applitools	15%
Others (Open Source & Academic)	10%

Table 3 : Top Challenges in AI-Powered Web Testing

Challenge	Severity (Scale 1–10)
Handling Dynamic UIs	9
Flaky Tests	8
Integration with Legacy Systems	7
Data Privacy & Security	6
Model Training Complexity	6

Table 4 : Projected Adoption Timeline for AI Web Testing Features

Year	Self-Healing Scripts	Autonomous Testing Agents	Federated Learning Adoption
2023	60%	10%	5%
2024	75%	20%	10%
2025	85%	40%	25%
2026	90%	60%	50%
2027	95%	80%	70%



Graph : Projected Adoption Timeline for AI Web Testing Features (2023-2027):

6. FUTURE DIRECTIONS

As AI continues to advance, web testing is poised for transformative innovations that will enhance automation, coverage, and reliability. This section highlights emerging trends and promising research avenues that are shaping the future landscape of AI-powered web testing.

6.1 Towards Autonomous Web Testing Agents

The future points toward fully autonomous testing agents capable of independently exploring web applications, generating test cases, diagnosing issues, and adapting to changes without human intervention. These agents will combine reinforcement learning, computer vision, and NLP to understand application logic, user workflows, and requirements at a semantic level. Autonomous agents will reduce reliance on manual scripting, continuously learn from new data, and operate seamlessly in dynamic environments. This shift promises to drastically reduce testing effort while improving defect detection in complex, evolving web systems.

6.2 Multimodal Testing (UI, API, UX)

Modern web applications are increasingly complex, involving rich user interfaces, backend APIs, and nuanced user experiences (UX). Future testing frameworks will integrate multimodal AI approaches that simultaneously validate UI correctness, API functionality, and UX quality metrics such as responsiveness and accessibility. This holistic testing perspective will leverage AI models trained on diverse data types—visual, textual, behavioral—and enable cross-layer defect detection. Multimodal testing will better align testing practices with real-world user interactions and system behavior.

6.3 Federated Learning for Web Testing

Privacy and data security concerns are growing, especially when testing involves sensitive user data across different organizations or geographies. Federated learning offers a promising approach, enabling AI models to train collaboratively on decentralized data sources without sharing raw data. Applying federated learning to web testing would allow organizations to jointly improve AI models for test

generation, anomaly detection, or failure prediction while preserving data confidentiality. This collaborative training can accelerate model improvements and broaden applicability across diverse application domains.

6.4 Standardization and Benchmarks

As AI-driven testing tools proliferate, there is an increasing need for standardized benchmarks, metrics, and evaluation protocols to compare their effectiveness rigorously. Future efforts will focus on creating shared datasets, test suites, and performance criteria that capture real-world challenges such as flaky tests, UI variability, and evolving software features. Standardization will enable objective tool comparisons, foster reproducible research, and guide industry best practices. Additionally, benchmarks tailored for AI-specific testing capabilities—like self-healing accuracy or anomaly detection precision—will become essential.

6.5 Collaboration Between Academia and Industry

Bridging the gap between academic research and industrial practice remains vital for accelerating innovation in AI-powered web testing. Future directions emphasize stronger partnerships that facilitate knowledge exchange, co-development of tools, and validation of academic prototypes in real-world settings. Joint initiatives can focus on addressing practical challenges such as scalability, explainability of AI decisions, and integration with legacy systems. Enhanced collaboration will help translate cutting-edge AI advancements into robust, deployable solutions that meet evolving industry needs.

7. CONCLUSION

7.1 Summary of Findings

This survey finds that AI integration in web testing is rapidly maturing, with self-healing scripts, visual AI, and predictive test prioritization increasingly common in tools like Testim, Mabl, Functionize, and Appltools. Machine learning dominates usage (40%), followed by NLP (25%) and computer vision (20%). Benefits include reduced maintenance, improved defect detection in dynamic UIs, and optimized regression cycles. However, challenges such as flaky tests, dynamic UI handling, and legacy integration persist. Emerging trends autonomous agents, multimodal testing, and federated learning offer high potential. AI is augmenting, not replacing, testers, enabling faster release cycles without compromising quality.

7.2 Implications for Practice and Research

For practitioners, these findings highlight the value of adopting AI-powered testing to enhance coverage, reduce maintenance overhead, and prioritize high-risk areas in fast-paced release cycles. Selecting tools with strong self-healing, visual AI, and predictive analytics can deliver measurable efficiency gains. For researchers, the results point to opportunities in improving flaky test mitigation, explainability of AI-driven decisions, and integration with legacy systems. Emerging domains like autonomous testing agents, multimodal validation, and federated learning warrant deeper exploration. Bridging academic innovation with industrial needs will be critical to ensure scalable, reliable, and transparent AI-driven testing solutions in increasingly complex web environments.

7.3 Final Thoughts

AI is redefining the landscape of web testing, shifting from experimental concepts to essential components of modern quality assurance. The progress in self-healing automation, intelligent test generation, and visual AI demonstrates clear value in accelerating release cycles without sacrificing reliability. However, sustainable success requires addressing persistent challenges, establishing industry standards, and ensuring transparency in AI-driven decisions. Continued collaboration between academia and industry will be pivotal in advancing robust, scalable solutions. By embracing AI's potential while maintaining human oversight, organizations can create adaptive, high-quality testing practices that meet the demands of evolving, complex web applications.

REFERENCE

- [1] Balsam, S., & Mishra, D. (2024). Web application testing—Challenges and opportunities. *Journal of Systems and Software*, 112186.
- [2] Li, T., Huang, R., Cui, C., Towey, D., Ma, L., Li, Y. F., & Xia, W. (2024). A Survey on Web Application Testing: A Decade of Evolution. *arXiv preprint arXiv:2412.10476*.
- [3] Garousi, V., Mesbah, A., Betin-Can, A., & Mirshokraie, S. (2013). A systematic mapping study of web application testing. *Information and Software Technology*, 55(8), 1374-1396.
- [4] Ricca, F., & Stocco, A. (2021). Web test automation: Insights from the grey literature. In *SOFSEM 2021: Theory and Practice of Computer Science: 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25–29, 2021, Proceedings 47* (pp. 472-485). Springer International Publishing.
- [5] Li, Y. F., Das, P. K., & Dowe, D. L. (2014). Two decades of Web application testing—A survey of recent advances. *Information Systems*, 43, 20-54.
- [6] Mridha, N. F., & Joarder, M. A. (2023). Automated web testing over the last decade: A systematic literature review. *Systematic Literature Review and Meta-Analysis Journal*, 4(1), 32-44.
- [7] Sampath, S., & Sprenkle, S. (2016). Advances in web application testing, 2010–2014. In *Advances in Computers* (Vol. 101, pp. 155-191). Elsevier.
- [8] Kertusha, I., Assress, G., Duman, O., & Arcuri, A. (2025). A Survey on Web Testing: On the Rise of AI and Applications in Industry. *arXiv preprint arXiv:2503.05378*.
- [9] Chen, G., Chen, G., Wu, D., Liu, Q., Zhang, L., & Fan, X. (2021, December). A selenium-based web application automation test framework. In *2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)* (Vol. 2, pp. 257-261). IEEE.
- [10] Akbulut, S., Gebreyesus, Y. T., Mishra, A., & Yazici, A. (2019, November). Systematic mapping on quality in web application testing. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)* (pp. 1-5). IEEE.
- [11] Ali, K., & Xiaoling, X. (2019). A Reliabel and an efficient web testing system. *arXiv preprint arXiv:1903.01221*.
- [12] Aydos, M., Aldan, Ç., Coşkun, E., & Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. *Journal of King Saud University-Computer and Information Sciences*, 34(9), 6775-6792.

- [13]Fuad, M. M. N., & Sakib, K. (2023, May). WebEV: A Dataset on the Behavior of Testers for Web Application End to End Testing. In 2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC) (pp. 79-83). IEEE.
- [14]Loubiri, O., & Maag, S. (2022, July). Automated Web Testing using Machine Learning and Containerization. In 2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC) (pp. 113-121). IEEE.