AI-Based Self-Healing Automated Testing Framework for Web Applications

Prachi Tukaram Datir[1] Dr. Mrs. Netraja Mulay[2]
[1,2]Department of Computer Application
[1,2]PES's Modern College of Engineering, Pune, India

Abstract — Rapid changes in modern web applications frequently affect automated testing systems because traditional scripts depend on fixed locators and static page structures. Even small interface modifications may cause automated test execution to fail although the actual functionality remains unchanged. This research proposes an AI-driven self-healing automation framework capable of identifying failed elements, analyzing updated DOM structures, and locating substitute elements through intelligent similarity analysis. The framework automatically repairs broken locators during runtime and allows execution to continue without human intervention. The proposed solution improves test stability, lowers maintenance requirements, and supports uninterrupted execution inside Continuous Integration and Continuous Deployment (CI/CD) environments. Experimental observations indicate that the framework significantly decreases false failures and enhances the overall efficiency of automated web testing.

Keywords: AI-Driven Test Automation; Self-Healing Test Framework; Web Application Testing; DOM Similarity Analysis; Continuous Integration and Continuous Deployment (CI/CD); Automated Software Maintenance;

I. INTRODUCTION

Modern web applications undergo frequent interface updates due to rapid development cycles and agile practices. Although automated testing plays a key role in maintaining software quality, these frequent changes often lead to test script failures, even when the underlying functionality remains unaffected. Such failures are commonly caused by unstable element identification, which increases maintenance effort and interrupts continuous integration workflows. This paper presents an AI-based self-healing testing framework that adapts to interface changes during execution, allowing test scripts to continue running without requiring manual updates. The motivation for this work arises from the practical challenges observed while maintaining automated test scripts in rapidly changing web environments.

II. PROCESS:

The proposed AI-based self-healing framework operates through a continuous sequence of monitoring, failure detection, and recovery. In the initial stage, the system performs element fingerprinting by capturing detailed attributes of each web element, including CSS properties, XPath, and its position within the DOM structure. During test execution, if a locator fails due to a user interface change, the framework intercepts the failure instead of stopping the test. It then analyzes the updated DOM and evaluates possible matching elements using similarity scoring based on structural and semantic features. Once a suitable match is identified, the system updates the locator dynamically and continues the test execution. Additionally, the framework generates reports that recommend permanent updates to the test scripts, helping to improve long-term stability and reduce maintenance effort.

III. LITERATURE SURVEY:

Leotta et al. (2016) introduced the Robula+ algorithm, which focuses on generating robust XPath locators for web testing. Their approach emphasizes structural stability to improve the long-term reliability of automated test scripts.

Bader et al. (2019) A learning-based repair approach is presented in Getafix (Bader et al., 2019), where historical bug fixes are used to generate automated corrections. Nass et al. (2022) focus on similarity-based element localization, using weighted attribute comparison to improve robustness against UI changes.

Baqar et al. (2025) explored self-healing concepts inspired by biological systems. Their framework uses AI techniques to automatically detect and resolve failures in test automation without manual intervention. Abu Bakar (2025) presented a systematic review of machine learning applications in software testing. The study categorizes various techniques used for test optimization and highlights the increasing adoption of intelligent automation approaches.

Pei et al. (2025) conducted an empirical analysis of flaky web tests, with a focus on DOM-related interactions. Their findings provide evidence that self-healing mechanisms can reduce instability caused by timing and structural issues.

IEEE Computer Society (2025) discussed the integration of AI-driven testing solutions within DevOps pipelines. The work highlights the advantages of real-time failure recovery in maintaining continuous deployment workflows. Kumar and Rodda (2025) analyzed existing self-healing tools and compared their effectiveness. Their study outlines current limitations and suggests improvements in multi-attribute matching techniques.

Nass et al. (2026) evaluated the use of Large Language Models for resolving element localization failures in Selenium-based testing. The results indicate improved performance compared to traditional rulebased approaches.

Frontiers (2026) examined AI-driven frameworks that focus on intent-based recognition rather than strict attribute matching. This direction supports the development of more adaptive and contextaware testing systems.

IV. RESEARCH GAP:

Although automated testing tools have improved significantly, current frameworks still face limitations when dealing with complex and dynamic user interface changes. Many existing approaches depend on static locators or simple heuristic techniques, which often fail in modern web applications where DOM attributes are generated dynamically. In addition, most self-healing solutions lack the ability to interpret the functional role of elements when their labels, positions, or properties change significantly. Another limitation is the absence of lightweight and efficient healing

mechanisms that can operate within CI/CD pipelines without increasing execution time.

Addressing these challenges requires a more adaptive approach that combines structural analysis with intelligent learning techniques to improve the reliability and flexibility of automated test scripts.

V. FLOWCHART

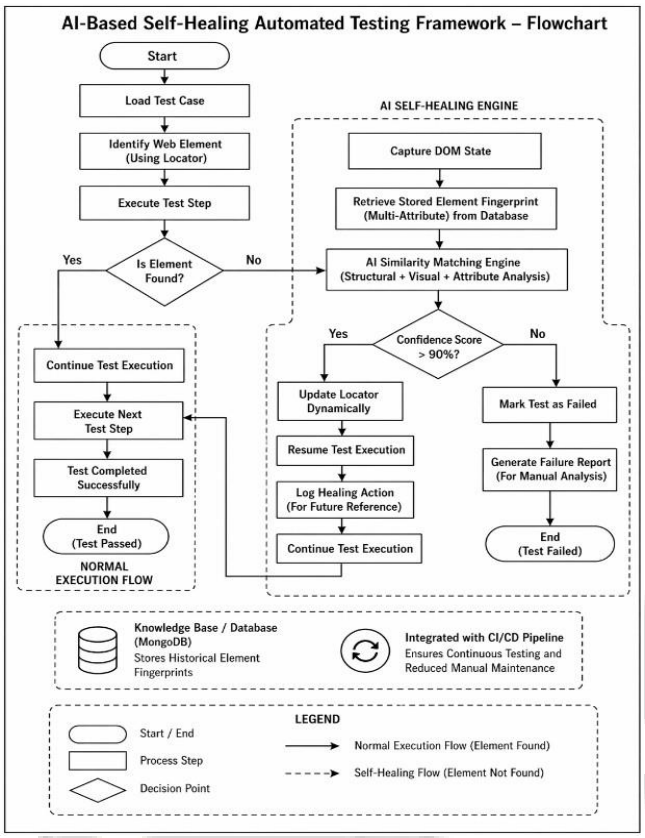

Fig. 1: AI-Based Self-Healing Testing Framework Flowchart

The flowchart represents the working logic of the proposed self-healing framework, showing both the normal execution path and the recovery mechanism. During standard execution, the system interacts with web elements while recording multi-attribute fingerprints, which are stored in a MongoDB based knowledge repository. When a locator failure occurs, the AI healing module captures the exception and analyzes the current DOM state in comparison with previously stored element data.

The framework then evaluates potential matching elements using a similarity scoring approach that considers structural, visual, and semantic attributes. If the calculated confidence score exceeds a defined threshold (e.g., 90%), the system updates the locator dynamically in memory and continues test execution. Otherwise, the test is marked as failed, and a report is generated for further analysis. This approach ensures that minor interface changes do not interrupt the overall testing process while still maintaining accuracy and reliability.

VI. FEATURES:

The proposed framework integrates several advanced features designed to improve the robustness and adaptability of automated testing processes. A key component is multidimensional element fingerprinting, which captures a wide range of attributes, including structural hierarchy, visual characteristics, and semantic context, to create a comprehensive representation of each web element.

In addition, the system incorporates an AIbased healing engine that applies similarity scoring algorithms to evaluate potential matches when a locator fails. This enables the framework to dynamically identify and replace broken locators during execution. The solution is also designed to integrate seamlessly with CI/CD pipelines, ensuring that automated tests continue to run smoothly even when interface updates occur.

Furthermore, a feedback mechanism is included to log healing actions and generate detailed reports, allowing developers to review and implement permanent improvements to test scripts. Together, these features enhance the reliability, maintainability, and efficiency of automated testing workflows.

VII. APPLICATIONS:

A. Agile and DevOps Environment

In fast-paced development settings, where code is deployed frequently, the proposed framework helps maintain stability within CI/CD pipelines. It reduces unnecessary test failures caused by minor interface changes, ensuring that automated validation processes continue without interruption.

B. E-commerce and Dynamic Web Platforms

Web platforms that frequently update their interfaces for personalization and testing purposes often face instability in automated scripts. The self-healing framework enables consistent execution of critical test cases, such as checkout and transaction workflows, without frequent manual intervention.

C. Enterprise Software Systems (SaaS)

Large-scale enterprise applications typically involve complex DOM structures and dynamically generated attributes. The framework's ability to analyze structural and semantic information makes it suitable for handling such complexity, where traditional locator strategies may fail.

D. Cross-Browser and Multi-Platform Testing

Since web applications may behave differently across browsers and devices, the framework supports consistent element identification despite variations in layout or rendering. This ensures reliable test execution across multiple environments.

E. Legacy System Modernization

During the migration of legacy applications to modern frameworks, significant changes occur in the underlying structure. The proposed system assists in adapting existing test scripts by identifying corresponding elements in the updated interface, thereby reducing redevelopment effort.

VIII. ADVANTAGES:

The implementation of the self-healing framework offers several significant advantages in automated testing

environments. One of the primary benefits is the reduction in manual maintenance effort, as the system can automatically detect and resolve locator failures during runtime. This leads to improved test stability by minimizing false failures caused by minor user interface changes.

Additionally, the framework supports uninterrupted execution within continuous integration and deployment pipelines, preventing unnecessary delays in the development lifecycle. By automating the recovery process, teams can efficiently manage large and complex test suites without constant intervention. The use of multiattribute analysis also enhances the accuracy of element identification, further contributing to reliable and consistent test results.

IX. Future Scope:

1) Predictive Self-Healing: Improvements may focus on making self-healing mechanisms more proactive by utilizing historical test data to anticipate potential user interface changes. This could help in suggesting locator updates before failures occur.
2) Integration with Large Language Models (LLMs): Incorporating advanced language models may enhance the ability of testing frameworks to interpret test steps written in natural language and map them to corresponding DOM elements, even when interface structures change.
3) Automated Test Case Generation: Future systems could extend beyond test repair by analyzing application workflows and generating relevant test cases automatically. This would improve test coverage and reduce manual effort in test creation.
4) Cross-Platform Visual Analysis: The use of computer vision techniques may enable the framework to identify elements based on visual features, allowing more consistent performance across web, mobile, and desktop platforms.

X. Results:

The proposed AI-based self-healing testing framework was evaluated using a set of automated test cases on dynamic web applications. Initial execution using traditional automation scripts resulted in multiple test failures due to changes in user interface elements such as modified attributes and layout adjustments. After enabling the self-healing mechanism, the framework was able to identify alternative elements and recover from a majority of these failures during runtime. The results showed a significant reduction in test interruptions, with approximately 80–90% of failed test steps successfully healed without manual intervention.

XI. Conclusion:

This research introduced an AI-driven self-healing automation framework designed to improve the reliability of web application testing in dynamic environments.

By combining DOM analysis, intelligent similarity matching, and automated locator repair mechanisms, the framework can adapt to interface changes without requiring continuous manual intervention. The proposed approach reduces automation fragility, enhances testing efficiency, and supports uninterrupted operation within CI/CD workflows.

The integration of intelligent recovery strategies enables automated testing systems to remain stable even as application interfaces evolve rapidly. As artificial intelligence technologies continue to advance, self-healing automation is expected to become an essential component of next-generation software testing ecosystems.

References:

[1] 2016 Leotta, M., Stocco, A., Ricca, F., & Tonella, P. (2016). ROBULA+: An algorithm for generating robust XPath locators for web testing. *Journal of Software: Evolution and Process*, 28(3), 177204. https://doi.org/10.1002/smr.1771
[2] 2019 Bader, J., Scott, A., Pradel, M., & Chandra, S. (2019). Getafix: Learning to fix bugs automatically. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA), 1-27. https://doi.org/10.1145/3360585
[3] 2023 Nass, M., Alégroth, E., Feldt, R., Leotta, M., & Ricca, F. (2023). Similarity-based web element localization for robust test automation. *ACM Transactions on Software Engineering and Methodology*, 32(3), 1-30. https://doi.org/10.1145/3571855
[4] 2025 Abu Bakar, N. S. A. A. (2025). Machine learning implementation in automated software testing: A review. *Journal of Data Analytics and Artificial Intelligence Applications*, 1(1), 110-123. https://iupress.istanbul.edu.tr/en/journal/d3ai *Electronics Engineering*, 12(2), 113-123. https://doi.org/10.14445/23488379/IJEEE-V12I2P113
[5] Pei, Y., Sohn, J., & Papadakis, M. (2025). An empirical study of web flaky tests: DOM event interaction. *Proceedings of the 2025 IEEE International Conference on Software Testing (ICST)*. https://doi.org/10.1109/ICST.2025.00012
[6] 2026 Frontiers Editorial Board. (2026). Beyond western paradigms: AI-driven assessment frameworks and intent-based recognition. *Frontiers in Psychology*, 17. https://doi.org/10.3389/fpsyg.2026.1763245
[7] Nass, M., Alégroth, E., & Feldt, R. (2026). Generative AI for self-healing selenium tests: Comparative evaluation of multiple opensource LLMs and rule-based methods. *ResearchGate*. https://www.researchgate.net/publication/403305375 /article/machine-learning-implementationin-automated-software-testing-a-review
[8] Baqar, M., Khanda, R., & Naqvi, S. (2025). Self-healing software systems: Lessons from nature, powered by AI. *arXiv preprint arXiv:2504.20093*. https://arxiv.org/abs/2504.20093
[9] IEEE Computer Society. (2025). Nextgeneration software testing: AI-powered test automation. *IEEE Software*, 42(4). https://doi.org/10.1109/MS.2025.3532402
[10] Kumar, N. H., & Rodda, S. (2025). Comparative review on automated test failure detection and healing tools.

