IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

Visual Test Automation Frameworks Using Visual AI and Behavior-Driven Development

K. Suganthi, S. Sakthivel, S. Sathishkumar, S. Subash Chandra Bose, T. Thirumeni Selvan
Assistant Professor, Department of Computer Science and Engineering
Students, Department of Computer Science and Engineering
Anjalai Ammal Mahalingam Engineering College, Kovilvenni, Tamil Nadu, India
sathishkumar1164s@gmail.com

Abstract: *Modern web applications rely heavily on complex graphical user interfaces that evolve frequently during development cycles. Traditional automation frameworks primarily validate functional behavior through DOM-based assertions and element locators, which do not guarantee visual correctness from the user's perspective. As a result, visual defects such as layout misalignment, missing components, and styling inconsistencies may remain undetected. This paper proposes a Visual Test Automation Framework that integrates Behavior-Driven Development (BDD) with Visual Artificial Intelligence (AI) to enhance automated UI validation. The framework adopts a layered, configuration-driven architecture and introduces a Visual Test Manager component that centralizes visual validation logic. Selenium WebDriver performs browser automation while Visual AI analyzes screenshots to detect visual regressions. Experimental evaluation demonstrates improved defect detection capability and reduced maintenance effort compared with traditional automation approaches. The proposed framework provides a scalable and maintainable solution for modern web application testing.*

Keywords: Visual Testing, Visual AI, Test Automation, Behavior-Driven Development, Selenium WebDriver, Software Quality Assurance

I. INTRODUCTION

The rapid evolution of web technologies has led to the development of highly interactive and visually rich user interfaces. Modern software systems frequently release updates through Agile and DevOps practices, which significantly shorten development cycles. As a result, automated testing has become an essential activity for ensuring that applications maintain high levels of quality and reliability across frequent releases.

Automation tools such as Selenium WebDriver are widely used for functional testing because they enable interaction with web elements through programmatic control. However, these tools typically verify only functional aspects of applications, such as element presence, attribute values, or text content. They do not confirm whether the user interface appears visually correct to end users.

User interface defects—including misaligned elements, missing images, broken layouts, or incorrect styling—may not affect functional assertions but can significantly degrade user experience. In addition, locator-based automation scripts are highly sensitive to minor structural changes in the Document Object Model (DOM), resulting in fragile tests that require frequent maintenance.

To overcome these limitations, this research proposes a Visual Test Automation Framework that integrates Visual Artificial Intelligence with Behavior-Driven Development. The framework aims to validate both functional behavior and visual correctness while improving maintainability and collaboration in automation testing.

DOI: 10.48175/IJARSCT-32895

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

II. LITERATURE REVIEW

Recent research has investigated the use of artificial intelligence and computer vision techniques to improve the effectiveness of automated software testing. Visual regression testing has emerged as an important approach for identifying user interface defects that cannot be detected through traditional DOM-based assertions.

Zhang et al. [1] proposed an AI-assisted visual regression testing approach for web applications. Their study utilized computer vision techniques to compare application screenshots and detect layout inconsistencies across different versions of web interfaces. The results demonstrated that visual comparison methods can detect UI defects more effectively than conventional DOM-based validation techniques.

Balahadia and Ragel [2] introduced a visual testing framework that integrates Visual Artificial Intelligence with Behavior-Driven Development. Their work highlighted the importance of combining human-readable test specifications with automated visual comparison techniques to improve collaboration between developers and testing teams.

Chen and Zhao [3] explored deep learning techniques for graphical user interface testing using computer vision models. Their research demonstrated that machine learning algorithms can automatically identify UI components and detect visual anomalies such as misaligned elements and missing interface components.

Kumar et al. [4] investigated the role of artificial intelligence in automated testing within continuous integration environments. Their work emphasized that integrating intelligent testing techniques into automation frameworks improves regression testing efficiency and enables early detection of UI-related defects during software development cycles.

Li et al. [5] proposed a scalable visual testing architecture designed for cloud-native web applications. Their research focused on modular framework design and configuration-driven execution to support automated UI validation across multiple applications and environments.

Although these studies demonstrate the effectiveness of Visual AI and computer vision techniques for UI testing, most existing solutions focus primarily on visual comparison algorithms rather than the architectural design of scalable automation frameworks. Therefore, there is a need for an integrated automation framework that combines Visual AI-based validation, Behavior-Driven Development, and configuration-driven execution to improve maintainability and scalability in modern web application testing environments.

III. RESEARCH GAP

Although recent studies have demonstrated the effectiveness of Visual Artificial Intelligence and computer vision techniques for detecting user interface defects [1]–[5], several challenges remain in existing automated testing approaches.

Most current automation frameworks rely heavily on DOM-based element locators to verify application behavior. While these methods are effective for functional validation, they often fail to detect visual defects such as layout misalignment, missing interface components, and styling inconsistencies that affect the user experience. In addition, locator-based automation scripts are sensitive to structural changes in the Document Object Model (DOM), which increases maintenance effort and reduces test reliability.

Existing research primarily focuses on improving visual comparison algorithms or applying deep learning techniques for UI defect detection. However, limited attention has been given to designing scalable automation frameworks that integrate Visual Artificial Intelligence with Behavior-Driven Development and configuration-driven execution.

Therefore, there is a need for a comprehensive automation framework that combines visual validation techniques with readable test specifications and modular architecture. The framework proposed in this research addresses this gap by integrating Visual AI-based screenshot comparison, Behavior-Driven Development, and a layered configuration-driven automation architecture to improve maintainability and reliability in modern web application testing.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

IV. PROPOSED FRAMEWORK

The proposed system is a Visual Test Automation Framework designed to improve the reliability and effectiveness of automated user interface testing. The framework integrates Behavior-Driven Development (BDD) with Visual Artificial Intelligence (AI) to validate both functional behavior and visual correctness of web applications.

Conventional UI automation solutions commonly depend on DOM element locators such as XPath expressions, CSS selectors, or element identifiers to interact with user interface elements. Although these approaches are effective for verifying functionality, they often fail to detect visual defects such as layout misalignment, missing components, and styling inconsistencies. Furthermore, locator-based automation scripts are highly sensitive to minor structural changes in the Document Object Model (DOM), which leads to fragile tests and increased maintenance effort.

To address these challenges, the proposed framework introduces a visual validation approach that analyzes screenshots of web pages and compares them with baseline images using Visual Artificial Intelligence. Instead of relying solely on DOM structures, the framework evaluates the visual appearance of the interface from the user's perspective. This approach allows the system to detect layout regressions, rendering issues, and missing UI components more effectively.

The framework also incorporates Behavior-Driven Development to improve collaboration among developers, testers, and stakeholders. Test scenarios are written using Gherkin syntax, which allows requirements to be expressed in a natural language format. These scenarios act as executable specifications that connect high-level behavioral descriptions with underlying automation code.

Another important feature of the framework is its configuration-driven execution model. Execution parameters such as target website URL, browser type, viewport size, and visual match level are defined in external configuration files. This design eliminates hardcoded dependencies and allows the framework to run visual tests on different web applications without modifying the test scripts.

Overall, the proposed framework aims to provide a scalable and maintainable solution for automated UI testing by combining Visual AI validation, readable BDD specifications, and a modular layered architecture.

V. SYSTEM ARCHITECTURE

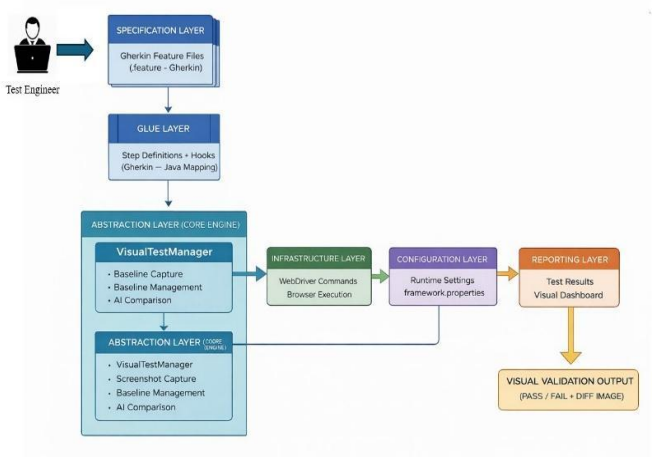

Fig. 1. Layered Architecture of the Visual Test Automation Framework

The proposed Visual Test Automation Framework follows a layered architecture that separates test specification, execution logic, visual validation, infrastructure management, configuration handling, and reporting. This modular design improves maintainability, scalability, and reusability while enabling the framework to perform visual validation across different web applications.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

A. Specification Layer

The Specification Layer contains Behavior-Driven Development (BDD) feature files written using Gherkin syntax. These files describe application behavior in a human-readable format using Given–When–Then statements. Test engineers define visual validation scenarios in this layer without interacting directly with automation code.

B. Glue Code Layer

The Glue Code Layer connects BDD feature files with executable automation logic. It consists of step definition classes and the test runner. Step definitions map Gherkin steps to Java methods, enabling the framework to interpret high-level specifications and translate them into executable test actions.

C. Visual Test Manager (Abstraction Layer)

The Visual Test Manager represents the core component of the framework and forms the Abstraction Layer. This component encapsulates all visual testing operations including baseline management, screenshot capture, and AI-based comparison. By centralizing visual validation logic, the framework separates test implementation from visual validation mechanisms, improving maintainability and reusability.

D. Infrastructure Layer

The Infrastructure Layer provides the underlying technologies required for automation execution. Selenium WebDriver performs browser automation and interacts with web elements, while the Applitools Eyes SDK provides visual validation capabilities. The Visual AI engine analyzes captured screenshots and compares them with baseline images to identify visual differences.

E. Configuration Layer

The Configuration Layer manages runtime parameters and framework settings. These include browser configuration, viewport sizes, target URLs, and visual comparison settings. Using a configuration-driven approach allows the framework to execute tests on multiple applications without modifying the core automation code.

F. Reporting Layer

The Reporting Layer generates detailed execution reports and visual validation results. Test outcomes, visual differences, and screenshots are presented through automated reports and visual dashboards. These reports help testers quickly identify visual regressions and analyze UI defects detected during automated execution.

VI. IMPLEMENTATION

The proposed Visual Test Automation Framework was implemented using a combination of modern automation and visual testing technologies. The implementation integrates browser automation, visual validation, configuration management, and reporting components to support scalable UI testing across different web applications.

A. Development Environment

The framework was developed using the Java programming language and the Maven build automation tool. Selenium WebDriver was used to automate browser interactions, while Cucumber was employed to implement Behavior-Driven Development (BDD). Visual validation was performed using the Applitools Eyes SDK, which applies artificial intelligence techniques to compare screenshots and detect visual differences.

The framework was developed and executed in the IntelliJ IDEA integrated development environment. Execution results and logs were generated using Allure and Cucumber reporting utilities.

B. Test Specification Using BDD

Test scenarios were written using Gherkin syntax in Cucumber feature files. These scenarios describe application behavior using natural language statements such as Given, When, and Then. This approach improves collaboration between testers, developers, and stakeholders by making test specifications easy to understand.

An example of a visual validation scenario is shown below:

Feature: Visual Validation

Scenario: Validate full-page layout consistency

Given a website URL is provided dynamically

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

When the website is opened
Then the visual appearance should match the baseline
The step definitions implemented in Java translate these Gherkin statements into executable automation commands.

C. Visual Test Manager Component

The core functionality of the framework is implemented through the VisualTestManager class, which encapsulates all visual validation operations. This component manages baseline screenshots, performs visual comparison, and interacts with the Applitools Eyes SDK.

The VisualTestManager provides methods for different validation types, including:

Full-page visual validation
Region-based visual validation
Responsive viewport testing
Dynamic element ignoring
Visual defect detection

A simplified code fragment used for visual validation is shown below:

```
eyes.check("Full Page Validation", Target.window().fully());
```

This operation captures a full-page screenshot and compares it with the stored baseline image using Visual AI.

D. Configuration-Driven Execution

The framework follows a configuration-driven execution model in which runtime parameters are defined in external configuration files. These settings include browser type, viewport size, Applitools API key, and target website URL. By separating configuration from test logic, the framework can execute visual tests across multiple applications without modifying the automation code.

E. Visual Defect Detection

The framework supports multiple visual defect detection scenarios, including layout inconsistencies, missing user interface components, and content overflow issues. During execution, screenshots of the current application state are compared with baseline images. Any visual differences are highlighted and reported through visual dashboards.

This approach allows testers to identify UI defects that traditional DOM-based automation techniques may fail to detect.

VII. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed Visual Test Automation Framework, several experiments were conducted on publicly available web applications. The objective of the experiment was to analyze the ability of the framework to detect visual defects such as layout inconsistencies, missing components, and content overflow issues.

A. Test Environment

The experiments were conducted in the following environment:

Parameter	Configuration
Operating System	Windows 11
Processor	Intel Core i5
RAM	8 GB
Programming Language	Java
Automation Tool	Selenium WebDriver
Visual Validation Tool	Applitools Eyes
Build Tool	Maven

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

Test Framework	Cucumber BDD
Browser	Google Chrome

B. Test Scenarios

Multiple visual validation scenarios were executed to evaluate the framework's capability in detecting different types of user interface defects. These scenarios were designed to simulate real-world UI testing conditions.

The following types of tests were performed:

Full Page Visual Validation – verifies the entire page layout against a stored baseline.

Region-Based Validation – validates specific UI components such as headers and navigation sections.

Missing Component Detection – checks whether critical UI elements are removed or missing.

Content Overflow Detection – identifies UI elements that exceed container boundaries.

VIII. RESULTS AND DISCUSSION

The experimental results demonstrate the effectiveness of the proposed Visual Test Automation Framework in detecting both functional and visual defects in web applications. Unlike traditional automation approaches that rely solely on DOM-based validation, the proposed framework leverages Visual Artificial Intelligence to analyze the rendered user interface and identify visual inconsistencies.

The overall test execution results are shown in Fig. 2. The Allure report provides a detailed summary of test outcomes, including total test cases, pass percentage, and defect categorization. A total of 13 test cases were executed, with a pass rate of 69.23%. The report clearly distinguishes between passed, failed, and broken test cases, enabling efficient analysis of system behavior and defect identification.

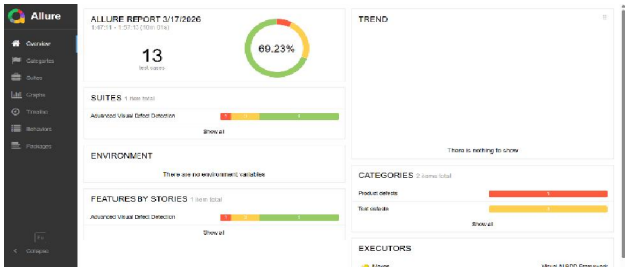

Fig. 2. Allure Test Execution Report Showing Overall Results

The framework successfully identified visual defects that are not detectable using conventional automation techniques. One such example is shown in Fig. 3, where a content overflow issue is detected. This defect occurs when a user interface element exceeds the viewport width, leading to layout distortion and poor user experience. Traditional tools such as Selenium WebDriver cannot detect such issues because they focus only on functional validation. In contrast, the proposed Visual AI-based approach captures and analyzes screenshots to identify such visual anomalies effectively.

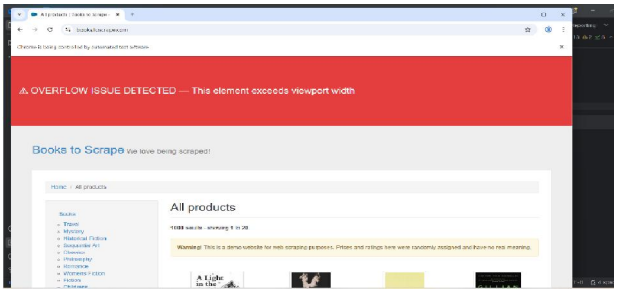

Fig. 3. Detection of Content Overflow Issue Using Visual AI

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

In addition to defect detection, the framework provides graphical insights into test execution results. Fig. 4 illustrates the distribution of test statuses and defect severity levels. The status chart shows the proportion of passed, failed, and broken test cases, while the severity chart categorizes defects based on their impact. These visualizations help testers quickly understand system quality and prioritize defect resolution.

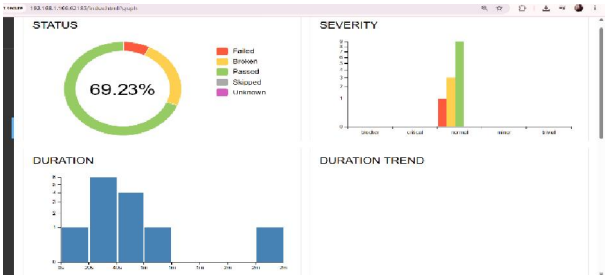

Fig. 4. Graphical Analysis of Test Status and Defect Severity

The experimental results are summarized in Table I.

TABLE I: VISUAL TESTING RESULTS

Test Scenario	Defect Introduced	Detection Result
Full Page Layout Validation	No defect	Passed
Header Region Validation	No defect	Passed
Missing Header Section	Header removed	Detected
Missing Navigation Section	Navigation removed	Detected
Footer Region Validation	Layout shift	Detected
Content Overflow Test	Large overflow element	Detected

Discussion

The results clearly indicate that the proposed framework is capable of detecting visual defects such as missing components, layout shifts, and overflow issues that traditional automation frameworks may overlook. By using screenshot-based comparison and Visual AI analysis, the system evaluates the user interface from the end-user perspective rather than relying solely on DOM structures.

Furthermore, the configuration-driven architecture reduces maintenance effort, as test scripts are less affected by minor UI changes. This improves test reliability and scalability across different web applications. Overall, the integration of Visual Artificial Intelligence with Behavior-Driven Development significantly enhances the effectiveness of automated UI testing.

IX. CONCLUSION

This paper presented a Visual Test Automation Framework that combines Visual Artificial Intelligence with Behavior-Driven Development to enhance automated user interface validation. Conventional automation tools primarily focus on DOM-based functional verification, which may overlook visual inconsistencies in web applications. The proposed framework introduces screenshot-based validation that analyzes the visual appearance of web pages and compares them with baseline images to identify UI defects.

The framework architecture separates specification, automation logic, visual validation, configuration management, and reporting into distinct layers, which improves modularity and maintainability. Experimental evaluation demonstrated that the framework is capable of detecting various visual issues, including missing elements, layout shifts, and overflow problems. These findings indicate that integrating Visual AI techniques into automation frameworks can significantly improve the effectiveness of UI testing in modern web applications.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, April 2026
Impact Factor: 8.2

X. FUTURE WORK

Although the proposed framework successfully demonstrates the advantages of integrating Visual AI with automation testing, several enhancements can be explored in future research.

First, the framework can be extended to support mobile application visual testing across different device resolutions and operating systems. Second, the framework may incorporate parallel execution across multiple browsers and devices to improve scalability in large-scale testing environments.

Another potential improvement involves integration with CI/CD pipelines, enabling automated visual testing during continuous integration and deployment processes. This would allow visual regressions to be detected early during software development cycles.

Future work may also explore cloud-based visual testing infrastructure to support large-scale distributed execution. Additionally, machine learning techniques could be applied to automatically classify detected visual defects and prioritize issues based on severity.

REFERENCES

[1] Y. Zhang, L. Chen, and H. Wang, "AI-Assisted Visual Regression Testing for Web Applications," Proc. IEEE International Conference on Software Testing, Verification and Validation (ICST), 2023.

[2] F. F. Balahadia and R. K. C. Ragel, "Visual Test Framework: Enhancing Software Test Automation with Visual Artificial Intelligence and Behavioral Driven Development," Proc. IEEE HNICEM Conference, 2023.

[3] J. Chen and M. Zhao, "Deep Learning Based GUI Testing Using Computer Vision," Proc. IEEE International Conference on Software Engineering and Artificial Intelligence (SEAI), 2024.

[4] R. Kumar, S. Patel, and A. Sharma, "AI-Driven Testing in Continuous Integration Pipelines," Proc. IEEE International Conference on Software Maintenance and Evolution (ICSME), 2024.

[5] L. Li, X. Sun, and T. Wu, "Scalable Visual Testing Architecture for Cloud-Native Applications," Proc. IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2025.

[6] H. Park, J. Kim, and S. Lee, "Automated Visual UI Testing Using Deep Learning Techniques," Proc. IEEE International Conference on Big Data and Smart Computing (BigComp), 2023.

[7] S. Gupta and P. Sharma, "Computer Vision Based UI Testing for Web Applications," Proc. IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2024.

[8] K. Tanaka and M. Suzuki, "Visual Regression Testing Using AI-Based Image Analysis," Proc. IEEE International Conference on Software Quality, Reliability and Security (QRS), 2024.

[9] A. Singh, R. Patel, and D. Shah, "Improving UI Test Automation Using Visual Artificial Intelligence," Proc. IEEE International Conference on Intelligent Systems and Applications (INTELLI), 2023.

[10] M. Brown and J. Wilson, "Behavior-Driven Development for Scalable Test Automation," Proc. IEEE International Conference on Software Engineering and Knowledge Engineering (SEKE), 2023.

[11] P. Zhao and X. Liu, "Automated GUI Testing with AI-Based Visual Recognition," Proc. IEEE International Conference on Machine Learning and Applications (ICMLA), 2024.

[12] S. Karthik and R. Ramesh, "Visual Regression Detection in Web Applications Using Deep Learning," Proc. IEEE International Conference on Emerging Technologies in Computing (ICETC), 2025.

[13] D. Wang, H. Li, and Y. Chen, "AI-Based Automated UI Testing Framework for Web Systems," Proc. IEEE International Conference on Smart Computing (SMARTCOMP), 2023.

[14] T. Nguyen and P. Tran, "Intelligent Web UI Testing Using Computer Vision Techniques," Proc. IEEE International Conference on Software Engineering Research and Practice (SERP), 2024.

[15] J. Lee, K. Park, and S. Choi, "Cloud-Based Visual Testing Framework for Web Applications," Proc. IEEE International Conference on Cloud Engineering (IC2E), 2025