ISSN (Online): 3139-7689

Available online: www.ijsrmes.com

International Journal for Scientific Research in Modern Engineering and Science

Published in Conference Special Issue: Vol. 1, Special Issue 1, pp. 245–250, 2026

AN INTELLIGENT AI-ASSISTED FRAMEWORK FOR EARLY DETECTION OF OWNER-SIDE PAYMENT WORKFLOW FAILURES USING BROWSER AUTOMATION

Vaibhav Sahu[1]*, Shankar Sharan Tripathy[2]

[1]*Student, Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai*

[2]*Assisant Professor, Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai*

Abstract

Online payment workflow processes are considered to be among the most critical elements of contemporary web-based commerce systems. Even the slightest disruption in the user interface might result in considerable loss and compromise the trust of the user. Conventional monitoring systems are mostly based on backend logs and/or infrastructure metrics, and/or manual testing carried out on a periodic basis. Although these systems are quite efficient and useful for identifying failures on the server side, they are not very effective in identifying failures on the user interface side and the integration side, which are the most common sources of failures during the course of a transaction.

In this paper, we have proposed an intelligent AI-assisted system for the early identification of owner-side payment workflow failures through the use of browser automation. According to the proposed model, the system would simulate the user interface and validate the payment workflow up to certain safe points without actually executing any financial transactions. It would also have the ability to distinguish between customer-side payment failures and system-side failures, which might have resulted due to UI failures, integration failures, redirect failures, and/or workflow timeouts. It has also been experimentally proven that the proposed model would have the ability to detect disruptions at the user interface level, which might have gone undetected by conventional monitoring systems

Keywords: *Payment Workflow Monitoring, Intelligent Automation, Browser Automation, Anomaly Detection, UI-Level Failure Detection.*

1. Introduction

1.1 Background

Online payment workflow is an essential part of the infrastructure of modern e-commerce systems. The reliability of the workflow is critical in the generation of revenues as well as the satisfaction of consumers. A temporary halt in the workflow during the payment stage may result in the loss of consumers.

Most of the existing monitoring systems have been designed to provide backend observability. These systems include server logs as well as API checks. These systems have been found to provide efficient infrastructure failure detection. Nevertheless, the systems have failed to provide efficient mechanisms for the detection of user

interface level failures. This has led to situations where the backend is running efficiently, but the payment workflow is experiencing issues.

Advancements in automation technologies in the browser have made it possible to simulate the interactions of real users in modern web applications. These technologies can be used to provide efficient monitoring of the workflow.

1.2 Problem Statement

In most cases, the workflows fail due to user interface issues or integrations. These failures can be non-responsive UIs, failed redirects, or even the occurrence of timeout situations while the customer checks out.

Another challenge with the owner side of the application is that the failures do not trigger backend alerts. As a result, the owner side of the application will not be aware of the issues until the customer reports a failed transaction. Manual testing methods involve periodic testing, while backend monitoring lacks UI level visibility.

Another challenge with the application is the occurrence of customer-side payment failures due to issues like invalid card information or bank-side transaction failures. These types of failures should be classified as valid application failures since they occur during the course of a normal workflow. Without a proper system for the classification of application failures, the monitoring system will misinterpret the valid customer-side failures as application failures.

A need exists for an automated system that can continuously test the application workflow at the browser level while distinguishing between valid customer-side failures and application failures.

1.3 Research Objectives

The main goal of the proposed research is to design and evaluate an intelligent browser-based system for the early detection of owner-side payment workflow failures.

Specific objectives of the proposed research include the following:

- Developing a well-structured classification system for payment workflow failures
- Designing an automatic system for simulating real customer interactions up to well-defined safe checkpoints
- Developing a system for validating the workflow execution in a way that differentiates customer-side errors from owner-side system defects
- Collecting workflow execution data for systematic reporting of failures

1.4 Contributions

This research makes the following contributions:

- A classification framework to identify payment workflow failures on the customer side and the owner side
- A browser-based monitoring tool to validate payment workflows at the UI level

- Rule-based detection of payment workflow failures with evidence collection
- An evaluation of detection effectiveness through controlled automated execution of workflows

2. Methodology/ Approach

2.1 Failure Classification and Scope Definition

To allow for the proper detection of system defects, payment workflow outcomes have been classified into two categories.

Customer-Side Failures

Customer-side failures encompass payment workflow outcomes resulting from the customer or external financial systems. These outcomes do not include defects in the application.

Some examples of customer-side failures include the following:

- Invalid card information
- Insufficient account balance
- Incorrect CVV or expired card
- Bank-side transaction rejection
- Aborted payment transactions initiated by the customer

These outcomes have been classified as proper system behavior.

Owner-Side Failures

Owner-side failures encompass payment workflow outcomes resulting from defects in the application or the system. These types of failures directly impact the payment workflow.

Some examples of owner-side failures include the following:

- Failure of payment buttons to respond
- Failure of the page to render completely
- Failure of the system to redirect to payment gateways
- Failure of the system due to session timeouts during the payment workflow
- HTTP 4xx/5xx errors during payment processing
- JavaScript exceptions interrupting workflow continuity

The proposed system primarily addresses the detection of owner-side failures.

2.2 System Workflow

The system simulates a realistic payment workflow through automated browser interactions. The workflow includes the following steps:

1. Launch automated browser session
2. Navigate to the e-commerce website
3. Select a product and add it to the cart

4. Proceed to checkout
5. Fill delivery information
6. Select payment method
7. Initiate payment workflow
8. Validate predefined checkpoints
9. Terminate execution at a safe checkpoint prior to transaction confirmation

The termination checkpoint ensures that no real financial transactions occur during monitoring.

3.3 Validation and Anomaly Detection Logic

Validation logic is applied at specific checkpoints during workflow execution. These checkpoints verify:

- Page load completion within defined time thresholds
- Visibility and responsiveness of critical UI elements
- Successful redirect to the payment interface
- Continuity of workflow transitions

Two types of validation rules are applied:

- **State-based validation:** checks UI element presence and interaction capability
- **Threshold-based validation:** verifies acceptable response times

Failures are classified deterministically using rule-based logic to ensure reproducibility and interpretability.

4.4 Data Collection and Evidence Generation

During each execution cycle, the system collects structured operational data including:

- Execution timestamp
- Workflow stage
- Validation results
- Execution duration
- Failure classification (if applicable)

When an anomaly is detected, the system captures diagnostic artifacts such as:

- Screenshot of the UI state
- Execution logs
- Workflow stage where failure occurred

These artifacts support root-cause analysis and operational reporting.

3. RESULT AND DISCUSSION

3.1 Failure Detection Performance

The proposed framework's effectiveness was also verified by repeating the automated workflow for the payment process. Both the normal operating cases and the cases with deliberately induced failure were analyzed.

The system successfully detected various owner-side failures, including:

- Non-responsive payment initiation buttons
- Failed or delayed redirects to the payment interface
- Timeout cases for the workflow
- Absence or improper rendering of UI components

In cases where deliberately induced customer-side failures were included, such as cases with invalid input data, the system successfully classified the outcomes as valid workflow behavior rather than system failures, thereby validating the effectiveness of the structured failure classification system.

Comparative observations indicate that the automated workflow monitoring allows for the detection of system-side disruptions much earlier compared to the reactive methods of detecting system-side disruptions through customer complaints or testing.

Execution Run	Workflow Duration (s)	Failure Detected	Failure Stage	Evidence Generated
Run 1	11.8	No	–	–
Run 2	14.2	Yes	Payment redirect stage	Screenshot + log
Run 3	12.6	No	–	–
Run 4	15.1	Yes	Payment interface loadi	Screenshot + log
Run 5	11.4	No	–	–

Table 1: Automated Workflow Execution Results

3.2 Reliability and Operational Safety

This framework operates within a set of defined safe checkpoints that stop the workflow from executing before the actual confirmation of the financial transactions. This ensures that the monitoring activities do not inadvertently execute any financial transactions.

Repeated workflow execution showed the detection to be consistent for the same failure conditions. This proves the reliability of the system.

However, the effectiveness of the system depends on the proper definition of the validation checkpoints. Misconfiguration can result in false positives or false negatives, especially for web interface dynamism.

Despite the limitations, the system represents a good practice for the proactive monitoring of financial workflows.

4. Conclusion

This research proposed an intelligent framework of automation that is geared towards the early detection of owner-side payment workflow failures in web-based commerce systems. This is done by simulating realistic user interactions and using structured validation rules at predetermined checkpoints.

The proposed approach is successful in differentiating between actual system defects and expected customer-side failures while at the same time providing evidence for operational analysis. From the observations made during the experiments, it is clear that browser-level workflow validation can enhance detection latencies for critical financial transactions.

There is a possibility of improving the proposed framework in the future by incorporating machine learning techniques into the proposed framework, as well as predictive analysis using historical execution data and expanding the proposed framework to cover other critical user workflows beyond payment workflow failures.

References

[1] M. Moń and B. Pańczyk, "A comparative analysis of web application test automation tools," Journal of Computer Sciences Institute, vol. 35, pp. 159–165, 2025.

[2] A. Antonczak and B. Bylina, "Analysis of end-to-end test automation tools based on the examples of Selenium WebDriver and Playwright," Proceedings of the Conference on Computer Science and Intelligence Systems, 2024.

[3] A. J. R. Cos, "Comparative reliability analysis of Selenium and Playwright: Evaluating automated software testing tools," Asian Journal of Research in Computer Science, 2025.

[4] B. García and J. C. Dueñas, "Automated functional testing based on the navigation of web applications," arXiv preprint arXiv:1108.2357, 2011.

[5] P. K. Adapala, "Evaluating synthetic monitoring tools for financial applications," International Journal of Engineering Research Publications, 2023.

[6] H. Lenke, "Synthetic monitoring and end-to-end testing: Two sides of the same coin," USENIX Login, 2024.

[7] Catchpoint Monitoring Guide, "Synthetic transaction monitoring and proactive failure detection," 2025.

[8] Paessler Monitoring Blog, "Synthetic monitoring and proactive detection of website failures," 2025.

[9] Microsoft, "Playwright: Fast and reliable end-to-end testing for modern web applications," Official Documentation, 2025.