AutonomQA: Self-Planning LLM Agents for High-Coverage Web QA

Mushfiqur Rahman, Istiak Ahmmed Rifti, Rakesh Debnath

ABSTRACT

Manual web testing is time-consuming, repetitive, and prone to human error—posing a major bottleneck in the modern software development lifecycle. To address this, we present **AutonomQA**, an autonomous multi-agent testing framework powered by large language models, designed to automatically execute high-coverage web tests from natural-language descriptions. Given a user-defined testing goal, the **Instruction Agent** decomposes it into unit cases. These unit cases are then handed over to the Execution Agent, which interactively carries out each step using LLM-based reasoning and control. If the **Execution Agent** encounters uncertainty or fails to proceed, a specialised Tool Agent is invoked to leverage external utilities. Finally, the **Reporter Agent** aggregates the outcomes into a human-readable summary, highlighting passed and failed cases for developer insight.

INTRODUCTION

- A **2023** survey shows developers waste **25%** of their time on manual testing, costing **$30–50K** per person/year.[1,2]
- Modern CI/CD pushes updates rapidly, but test suites fall behind due to fragile, hand-written scripts.[1,2]
- Minor UI changes cause test rot, demanding constant maintenance.
- Record-and-replay tools break on DOM changes and need developer fixes.
- High-level test intents stay constant, even as UI structures evolve.
- Manual web testing is time-consuming, repetitive, and error-prone—a major development bottleneck.
- We propose **AutonomQA**—a multi-agent framework powered by LLMs—to automate and adapt web testing.
- The **Instruction Agent** breaks down natural-language goals into granular test cases.
- The **Execution Agent** uses LLM reasoning to carry out steps, adapting dynamically when tests fail.
- The external tools are called when needed, and the **Reporter Agent** summarises results for developers.
- Enables fully automated, high-coverage web tests with minimal human intervention.

Manual Testing vs. AutonomQA

	Manual Testing	AutonomQA
Time Waste	25% of time wasted	Minimal human intervention
Update Speed	Falls behind CI/CD	Fully automated, high-coverage
UI Changes	Causes test rot	Adapts dynamically
Error-prone	Repetitive and error-prone	Automated, high-coverage
Test Creation	Hand-written selenium scripts	Natural-language goals to test cases
Tool Use	N/A	Invokes external tools
Reporting	Detailed reporting needs human effort	Summarizes results for developers

METHODOLOGY

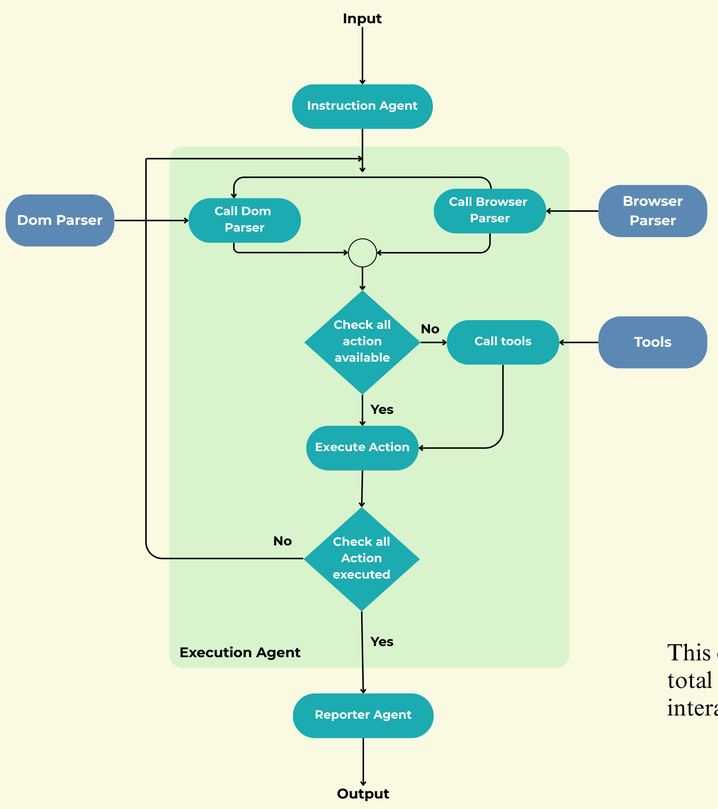

Figure: AutonomQA Multi-Agent Execution Workflow

This diagram illustrates the core workflow of AutonomQA's test automation framework. The Instruction Agent breaks the input goal into unit test cases. For each unit test case, the Main Agent delegates parsing tasks to the DOM and browser parsers. It then checks for actionable steps, invokes external tools if needed, executes the actions, and verifies whether all steps have been completed. The Report Agent compiles the outcomes into a structured output summary.

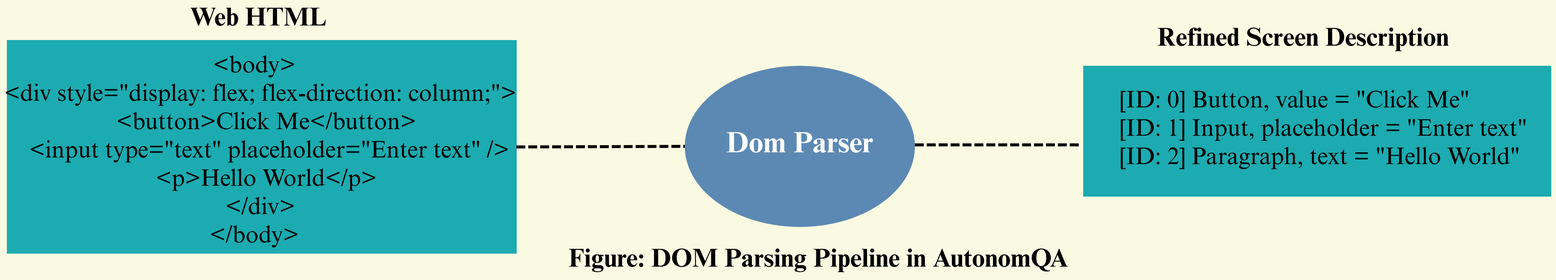

Figure: DOM Parsing Pipeline in AutonomQA

This figure illustrates how the DOM Parser processes raw HTML content. The HTML structure, containing buttons, input fields, and text elements, is analyzed and transformed into a structured, element-wise format. Each DOM element is assigned a unique identifier and simplified into a semantic description that enables downstream agents to reason about available actions in the web interface.

Browser Parser

- Current Tab: www.example.com/login
- Previous Tab: www.example.com/description1
- Total Tabs Opened: 2
- Tab Titles: "Login Page", "Product Details"
- Tab History Stack: www.example.com → www.example.com/description1 → www.example.com/login

Figure: Browser Parser State Overview

This component captures the browser's current state including active and previously visited tabs, total tabs opened, tab metadata etc. It enables multi-tab reasoning and efficient page-specific interaction planning for the testing agent.

Tools

List of tools:
1. Validate Login
2. Validate Form Fillup
3. Extract hidden HTML
4. Use external service etc.

Figure: Tool Agent Capabilities

The Tool Agent handles tasks that require specialized processing beyond standard DOM interaction. It supports functions such as login validation, form checking, hidden HTML extraction, and external service integration.

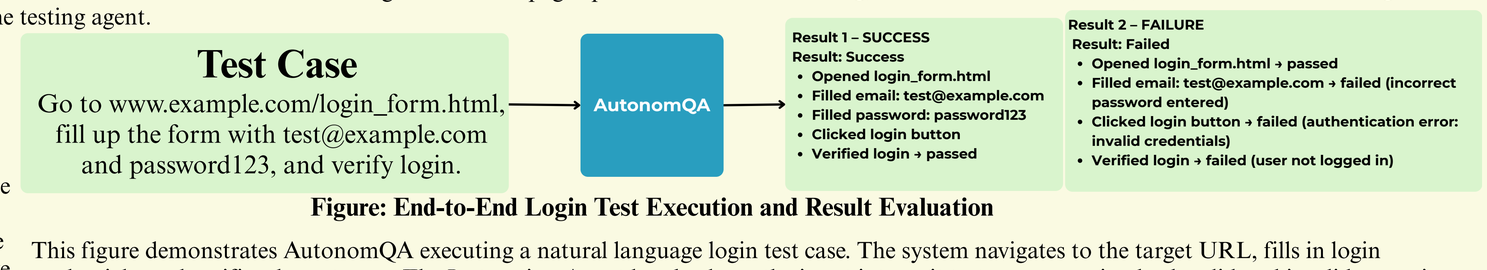

Figure: End-to-End Login Test Execution and Result Evaluation

This figure demonstrates AutonomQA executing a natural language login test case. The system navigates to the target URL, fills in login credentials, and verifies the outcome. The Instruction Agent breaks down the input into unit test cases covering both valid and invalid scenarios—two such cases are shown here. The results panel displays one successful and one failed execution, highlighting each step's outcome to support detailed debugging and validation.

RESULT AND DISCUSSION

Evaluation Setup:

- Applications: 10 different web-based applications (e.g., e-commerce, email, dashboard, and forms).
- Browsers: Chrome and Firefox on desktop environments.
- Test Case Complexity: Varied in terms of the number of steps required to complete each scenario (e.g., login, search, checkout).

Accuracy and Efficiency of AutonomQA:

- AutonomQA successfully completed **88%** of browser-based test cases using **claude 4.**
- It handled natural language test descriptions without any predefined scripts.
- The average execution time per test case was 15 seconds, demonstrating high efficiency.
- AutonomQA achieved an **83%** error recovery rate using **claude 4.**
- It autonomously detected failed interactions like missing elements or wrong clicks.
- Upon errors, it effectively navigated backward and chose the correct alternative actions.
- We also used fine tuned LLama models and it also achieves comparable performance.

Future Works:

- Integrate computer vision into the system to help the LLM better interpret and understand on-screen code structures and visual layouts.
- Introduce a dedicated Thinking Agent alongside the Execution Agent to reason through complex workflows and improve execution efficiency.
- Integrate complex memory into the system so that the agent can retain contextual knowledge across multiple test steps, recall past interactions, track execution history, and make informed decisions based on that

Model	Test Execution Success Rate	Error Recovery Rate	Execution Time per Step
Claude 4	88%	83%	15.0s
GPT-4o	73%	78%	11.8s
LLaMA-3.1 8B (Base)	60%	10%	11.0s
LLaMA-3.1 8B (Fine-tuned)	65%	55%	11.0s

Table 1: Test Execution Performance Metrics for different LLM

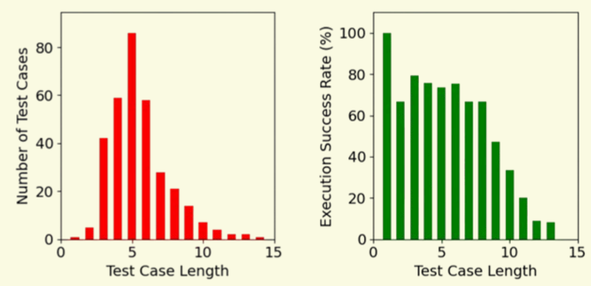

Table 2: The left histogram shows test case length distribution, with most cases in the 3–10 step range. The right histogram shows that shorter test cases succeed more often, while longer ones have lower success due to increased interactions and error risks

CONCLUSION

- **Robust & Accurate**: AutonomQA hits an **88 %** success rate and **83 %** self-recovery on 10 very different web apps in both Chrome and Firefox. Even when the DOM shifts or a button label changes, it still finds its way—saving developer from brittle scripts and surprise test breaks.
- **Major Productivity Boost**: By turning natural-language goals into working tests that repair themselves, AutonomQA hands back **≈ 25 %** of every developer's week—worth **$30–50 K** per engineer each year. Fewer hours spent chasing flaky tests means more time for shipping features (and maybe finally taking that coffee break).
- **Scalable Impact on CI/CD:** Its multi-agent, LLM-powered design lets test suites heal themselves as code evolves, so projects QA keeps pace with rapid releases instead of slowing them down.

REFERENCES

1. Wang et al., "Software Testing with Large Language Models: Survey, Landscape, and Vision," IEEE Transactions on Software Engineering, 2023. arXiv:2307.07221
2. Sherifi, B., Slhoub, K., & Nembhard, F. (2024). The Potential of LLMs in Automating Software Testing: From Generation to Reporting. arXiv:2501.00217. https://doi.org/10.48550/arXiv.2501.00217

Associated With

Associate partner

E-learning partner

Contributor

Media partner

TV partner

Photographic partner