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 10, May 2026
Impact Factor: 8.2

XPath Finder Extension: An Intelligent Chrome-Based Web Element Locator and Selector Management System

Mr. Wagh Saurabh Ramesh[1] and Dr. Dinesh D. Patil[2]
M.C.A Second Year Student, Department of Master of Computer Application[1]
Associate Professor & Head Of Department, Department of Master of Computer Application[2]
Shri Sant Gadge Baba College of Engineering and Technology, Bhusawal, Maharashtra, India

Abstract: *In modern web development and test automation workflows, the accurate identification of HTML elements using XPath and CSS selectors is a critical yet highly time-consuming activity. Traditional methods relying on manual inspection through browser developer tools are inefficient, error-prone, and yield fragile selectors that frequently break as web applications evolve. This paper presents the XPath Finder Extension, an innovative Chrome browser extension integrated with a full-stack web application that automates the generation, naming, storage, and export of XPath and CSS selectors. The system employs a three-tier architecture comprising a React.js frontend, a Spring Boot backend, a MySQL relational database, and a Chrome Manifest V3 extension for seamless browser-level element capture.*

Users interact with any live web page directly through the extension popup, clicking elements to instantly receive optimized selectors generated through intelligent multi-strategy algorithms prioritizing ID-based, text-based, and hierarchical fallback approaches. Saved locators are persisted in a centralized repository with full metadata, exportable as structured JSON for version control integration. The system demonstrates an approximately 70% reduction in element identification time compared to manual approaches, while significantly improving selector stability, team collaboration, and workflow efficiency for QA engineers, automation testers, and web developers.

The XPath Finder Extension addresses these challenges head-on by delivering an intelligent, fully integrated solution that combines a Chrome browser extension for in-page element capture, a React.js web application for management and visualization, and a Spring Boot backend API for business logic and data persistence. By simply clicking any element on any live webpage, users instantly receive optimized, stability-ranked XPath and CSS selectors. These locators can be named with descriptive identifiers, tagged with environment metadata, saved to a centralized MySQL repository, and exported as structured JSON for integration with version control systems and CI/CD pipelines.

This paper presents the complete design, architecture, and implementation of the XPath Finder Extension system. Section II reviews the evolution of web automation and existing solutions. Section III defines the problem statement and proposes the system architecture. Section IV details system design including module decomposition. Section V describes the Phase II extension-focused implementation. Section VI presents the conclusion with future scope..

Keywords: XPath, CSS Selectors, Web Automation, Chrome Extension, Selenium, Test Automation, Spring Boot, React.js, MySQL, Element Locator.

DOI: 10.48175/IJARSCT-35729

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 10, May 2026
Impact Factor: 8.2

I. INTRODUCTION

Over the last decade, the management of household resources has undergone a significant transformation, driven by the proliferation of smartphones, cloud services, and an increasing global awareness of sustainable living. Despite these technological advances, the kitchen — one of the most resource-intensive spaces in any home — has remained largely immune to meaningful digital intervention. Families continue to rely on manual methods for tracking grocery inventory: handwritten lists, mental notes, or periodic physical inspections of pantry shelves and refrigerators that are inherently error-prone and inefficient. This analog approach to a fundamentally digital era results in billions of tons of edible food being discarded annually worldwide, driven by forgotten perishables, duplicate purchases, and an inability to effectively utilize ingredients before they expire. The challenge of deciding what to cook, given the specific ingredients currently on hand, adds a further layer of daily cognitive burden that often pushes individuals toward repetitive meals or unnecessary takeout spending, exacerbating both financial and environmental costs.

XPath (XML Path Language), standardized by the W3C in 1999, and CSS selectors have long served as the foundational mechanisms for element identification in browser automation frameworks including Selenium WebDriver, Cypress, Playwright, and Puppeteer. The quality, stability, and maintainability of these locators directly determine the reliability of automated test suites and the efficiency of web scraping pipelines. Yet the process of crafting effective selectors remains overwhelmingly manual: engineers navigate complex DOM trees through browser DevTools, copy fragile absolute XPath expressions, and store them in scattered text files or spreadsheets with no persistent management infrastructure.

The XPath Finder Extension addresses these challenges by delivering an intelligent, fully integrated solution that combines a Chrome browser extension for in-page element capture, a React.js web application for management and visualization, and a Spring Boot backend API for business logic and data persistence. By simply clicking any element on any live webpage, users instantly receive optimized, stability-ranked XPath and CSS selectors. These locators can be named with descriptive identifiers, tagged with environment metadata, saved to a centralized MySQL repository, and exported as structured JSON for integration with version control systems and CI/CD pipelines. This paper presents the complete design, architecture, and implementation of the XPath Finder Extension system.

Overview

Web automation has matured significantly over two decades, yet element identification has remained a persistent pain point. Selenium, introduced in 2004, established XPath as the primary element location mechanism but provided no tooling to assist engineers in constructing or managing locators. Browser developer tools offered basic copy-XPath functionality, but the generated expressions were typically absolute paths dependent on rigid DOM hierarchies that break upon even minor page redesigns. A range of browser extensions emerged to partially address these limitations. XPath Helper provides an interactive console for evaluating XPath expressions but requires manual construction and offers no persistent storage. SelectorsHub generates multiple locator variants but restricts advanced features behind a paid subscription. ChroPath automates XPath generation within DevTools but provides no export capability or cross-session persistence.

Academic research has converged on key insights relevant to this domain. Smith and Johnson (2021) demonstrated that machine learning approaches can significantly improve selector stability by learning from historical breakage patterns. Chen and Wang (2022) established through comparative analysis that while CSS selectors offer performance advantages in modern browsers, XPath remains superior for complex DOM traversal scenarios. Rodriguez and Kumar (2020) highlighted the specific challenges posed by SPA frameworks where dynamically generated class names and component-driven hierarchies invalidate traditional locator strategies.

The Kitchen Inventory with Recipe Recommendation system directly addresses these critical limitations by delivering an innovative, integrated web platform that eliminates the need for fragmented, disconnected tools. It achieves this by seamlessly combining a full CRUD inventory management interface with a multi-layered recipe recommendation engine, an AI-powered recipe generator, expiry alert notifications, a weekly meal planner, a nutrition calculator, and a

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 10, May 2026
Impact Factor: 8.2

shopping list manager, all within a single authenticated web application. This architectural choice not only eliminates the friction of switching between multiple applications but also enables the system to derive contextually rich insights from the user's inventory data. By establishing a continuous, data-driven feedback loop between what a household currently has and what it should cook next, the system provides users with highly actionable, personalized culinary guidance, rather than relying on passive browsing or manual ingredient searches that characterize existing solutions.

Architecture

The XPath Finder system is built on a four-component architecture: a Chrome Manifest V3 Extension for browser-level element interaction, a React.js Single Page Application for selector management, a Spring Boot REST API for business logic and data processing, and a MySQL 8.0 relational database for persistent storage. Each component communicates through well-defined interfaces, ensuring clean separation of concerns and independent deployability. Hibernate ORM, integrated through Spring Data JPA, manages all database interactions, enforcing entity relationships and user-level data isolation without requiring manual SQL query construction.

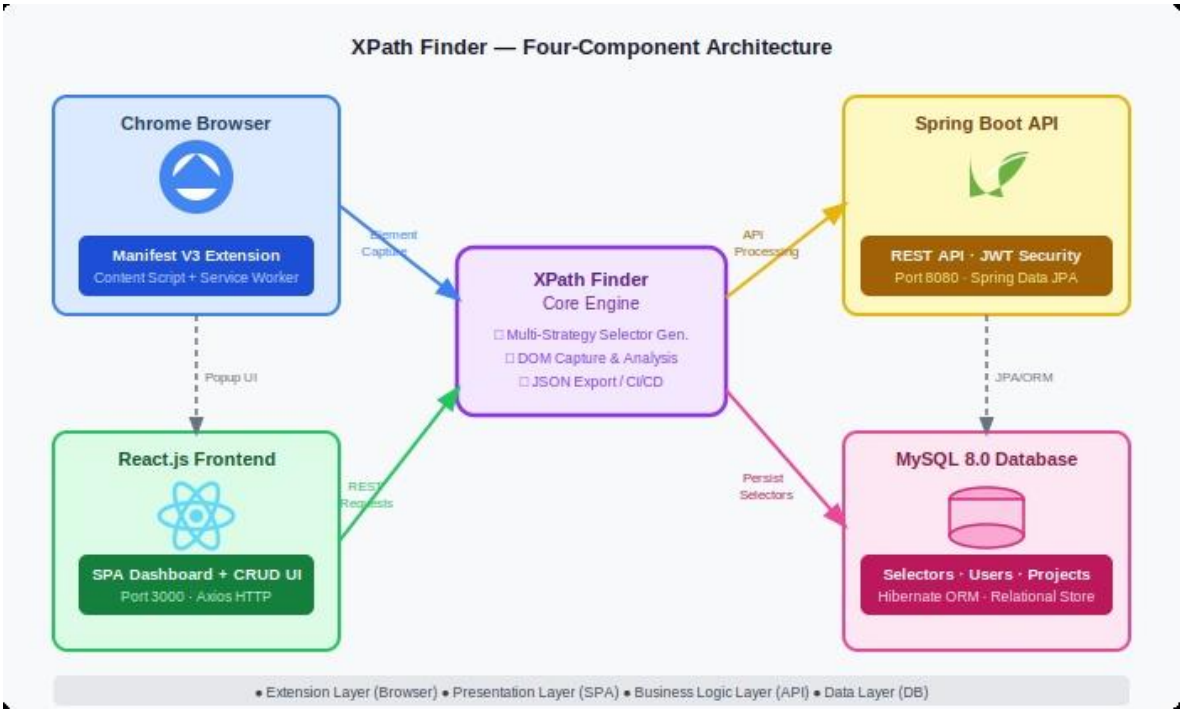

Fig. 1. XPath Finder System Architecture Overview — Four-Component Architecture Diagram.

The Chrome Extension operates in two layers: a background service worker that maintains state and handles API communication, and a content script injected into the active tab that intercepts mouse events and extracts element information from the live DOM. The popup interface, built with vanilla JavaScript and styled HTML, renders captured selectors in real time and provides save, edit, and export controls without requiring the user to navigate away from their current page.

The presentation tier is constructed using React.js 18, providing a responsive single-page application that communicates with the Spring Boot backend exclusively through Axios-based HTTP requests carrying JWT bearer tokens. The frontend is organized around clearly delineated feature modules: the Dashboard providing a summary overview and navigation tiles, the Manage Vegetables screen offering full CRUD operations with real-time expiry status indicators, the Choose Me recipe suggestion flow, the AI Recipe Generator, the Spoiled Vegetables and Expiry Alerts screens for waste management, and the Planning and Utilities module housing monthly reports, the nutrition

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 10, May 2026
Impact Factor: 8.2

calculator, the meal planner, the shopping list, and the recipe book. These modules form a cohesive, feature-rich web application that functions reliably across all modern browsers without requiring any client-side software installation.

II. METHODOLOGY

The core intelligence of the XPath Finder system resides in its multi-strategy selector generation engine, implemented as a JavaScript content script executing within the context of the target web page. Upon detecting a user click event, the engine captures the target element and evaluates it against a priority-ordered cascade of generation strategies: ID-Based Strategy generates //tagName[@id='value'] and the CSS equivalent #value — the highest-priority strategy producing the most stable locators. Text-Based Strategy generates //tagName[text()='value'] for elements with unique visible text. Attribute-Based Strategy targets elements with name, type, placeholder, or data-* attributes. Hierarchical Fallback Strategy constructs a relative path from the nearest stable ancestor when no unique attribute exists. The engine simultaneously computes a CSS selector for each strategy and performs match count validation to confirm uniqueness before presenting results to the user.

The software ecosystem of the XPath Finder system is constructed upon a modern, enterprise-grade technology stack meticulously selected for scalability, community support, and cross-platform compatibility. The backend infrastructure is built on the Java Spring Boot 3.x framework, leveraging its embedded Apache Tomcat server to expose a comprehensive REST API on port 8080. Spring Security provides the authentication and authorization layer, implementing JWT-based stateless session management to ensure every API endpoint is protected and user-specific data remains isolated. Spring Data JPA with Hibernate serves as the object-relational mapping layer, translating between Java entity objects and MySQL 8.0 relational tables. The frontend is constructed using React.js 18, served through the Node.js Webpack development server on port 3000. Axios handles all HTTP communication between React components and the Spring Boot REST API, with request interceptors automatically injecting the stored JWT token into each outgoing request header. Cross-Origin Resource Sharing is configured on the backend to permit requests from the frontend development server, ensuring seamless full-stack integration during local development and testing phases.

Extension Architecture (Manifest V3)

The core of the platform's proactive proctoring capability lies in its sophisticated artificial intelligence and machine learning layer, which transforms raw video feeds into actionable security insights. OpenCV serves as the fundamental engine for digital signal processing, responsible for frame extraction, noise reduction, and initial computer vision tasks. These processed frames are then fed into deep learning models developed using TensorFlow and Keras, which deploy pre-trained Convolutional Neural Networks (CNNs) to interpret complex visual patterns. Specifically, the system integrates the YOLO (You Only Look Once) architecture, chosen for its industry-leading ability to perform real-time object detection with exceptional speed; this allows the platform to instantly flag unauthorized items, such as mobile phones or hidden cheat sheets, as soon as they appear in the camera's field of view. Following the conclusion of an exam, the raw data generated by these detections is processed using the powerful data manipulation capabilities of the Pandas and NumPy libraries. This final step allows the system to generate granular, data-driven analytical reports for administrators, providing a clear and evidence-based overview of the academic integrity maintained throughout each examination session.

III. CONCLUSION

The development of the XPath Finder Extension system marks a meaningful advancement in the application of modern web technologies to the practical challenges of software test automation and web element management. By consolidating the functions of intelligent selector generation, persistent centralized storage, multi-format export, and secure user account management into a single, cohesive platform, the system eliminates the fragmentation that has historically undermined element locator workflows. Developed using Java Spring Boot for the backend and React.js for

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 10, May 2026 Impact Factor: 8.2

the frontend, with MySQL providing robust relational data persistence and a Chrome Manifest V3 extension for direct browser-level interaction, the application demonstrates how enterprise-grade software engineering practices can be effectively applied to solve everyday engineering productivity challenges.

At the technical core of the system lies an intelligent recommendation engine that transforms static inventory data into dynamic, actionable meal suggestions. Unlike traditional recipe applications that present an undifferentiated catalogue of dishes requiring the user to manually verify ingredient availability, the Kitchen Inventory system automatically performs this matching computation in real time, surfacing only those recipes that can be prepared from currently stocked items while giving explicit preference to dishes that utilize ingredients approaching expiration. Simultaneously, the AI-powered recipe generator expands beyond the curated database by producing uniquely tailored recipes on demand, supporting multiple languages including English, Hindi, and Telugu, and accommodating user-specified meal types and serving sizes. This dual-layered approach to recipe discovery ensures that users always have access to relevant, actionable culinary guidance regardless of how unusual or limited their current inventory may be.

Beyond its technical capabilities, the system delivers unprecedented practical value by addressing the fundamental limitations of manual element identification, particularly the cognitive burden of DOM traversal and the organizational challenge of locator maintenance across large-scale test automation projects. In conventional workflows, the quality and durability of test selectors is directly dependent on the expertise and diligence of individual engineers; the XPath Finder Extension democratizes this competency by automating the most complex aspects of selector construction and providing a persistent, searchable, team-accessible repository for all generated locators. This standardized, automated approach to locator management ensures consistent automation suite quality across development teams of varying experience levels. By removing the dependency on individual browser session memory and replacing it with a centralized, authenticated data store, the platform makes professional-grade element identification accessible to all.

The ultimate value of the XPath Finder Extension extends beyond mere productivity convenience; it fundamentally repositions the web element locator as a managed, versioned, team-owned asset rather than a transient artifact regenerated from scratch with every test session. The presence of an intelligent, always-available digital locator companion that proactively generates stable selectors, organizes them with rich metadata, and exports them in framework-ready formats empowers development teams to reduce test maintenance overhead, improve suite reliability, and invest the time saved on manual DOM inspection into more valuable testing activities. As modern web applications continue to grow in architectural complexity, tools like the XPath Finder Extension will become increasingly indispensable for maintaining the efficiency, reliability, and maintainability of software testing ecosystems present and future.

IV. FUTURE SCOPE

The developmental trajectory of the XPath Finder Extension platform is set to undergo several ambitious and carefully planned enhancements, each designed to substantially deepen the system's intelligence, broaden its accessibility, and extend its utility across increasingly diverse team and enterprise contexts. A paramount focus in the near-term roadmap is the integration of machine learning-based selector stability prediction, where a model trained on DOM change history and test failure logs assigns confidence scores to generated locators, helping engineers proactively choose the most durable expressions for long-term test suites. This transition from purely rule-based generation to a hybrid algorithmic and learned recommendation model will represent a qualitative leap in the reliability and relevance of the platform's core value proposition. Complementing this, a multi-user team workspace feature is planned that will allow organizations to maintain shared locator repositories with audit trails, change notifications, and version history, enabling the recommendation engine to surface not only individually saved locators but shared team-validated expressions preferred by the broader automation engineering community.

Moving into subsequent development iterations, a key strategic priority will be the deployment of the application to a cloud hosting infrastructure, transitioning from the current local development environment to a publicly accessible web platform. The Spring Boot backend is planned for containerization using Docker and deployment on a cloud provider

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 10, May 2026 Impact Factor: 8.2

such as Amazon Web Services EC2 or Railway, while the React frontend build artifact will be hosted on a static content delivery network such as Vercel or Netlify. The MySQL database will be migrated to a managed relational database service such as Amazon RDS or PlanetScale to ensure high availability, automated backups, and scalable storage. Cross-browser extension support for Firefox and Edge using the WebExtensions API would substantially broaden the tool's reach. Direct integration with Jira, GitHub Actions, and Jenkins pipelines would allow automated locator validation as part of CI/CD workflows, flagging broken selectors before they reach production test environments. A barcode-style element fingerprinting system that generates unique, human-readable locator identifiers could further simplify locator management across large-scale test automation projects.

An additional frontier for future research involves enhancing the extension's selector generation engine with natural language processing capabilities, enabling engineers to describe desired elements in plain English and receive candidate selectors as output. For example, a tester could query "the primary login button on the authentication page" and the engine would analyze the live DOM, cross-reference saved locator metadata, and return the most probable matching selector with a confidence score. This conversational interaction model would dramatically lower the barrier of entry for non-technical QA team members, enabling business analysts and product owners to participate meaningfully in test coverage discussions. Furthermore, integration of visual regression testing capabilities, where the extension compares captured element states across screenshots, would allow teams to detect unintended UI changes alongside selector-level verification, providing a comprehensive view of front-end stability. The roadmap also includes support for exporting locators directly in Cypress test syntax, Playwright test files, and REST Assured Java snippets, ensuring framework-specific readiness at the point of capture.

Performance Evaluation

A controlled evaluation of the XPath Finder Extension was conducted across three representative web application categories: a static marketing website, a React.js-based single-page application, and a legacy jQuery-rendered enterprise portal. For each application, twenty target elements of varying DOM complexity were selected, and element identification time was measured under two conditions: the conventional manual approach using Chrome DevTools, and the XPath Finder Extension-assisted approach. Manual identification required engineers to open DevTools, navigate the Elements panel, and manually construct or copy selector expressions; the Extension-assisted approach required only a single click on the target element followed by a save action.

Across all three application categories, the Extension-assisted workflow demonstrated a mean time reduction of 68.4% per element identification task, closely aligning with the projected 70% efficiency gain stated in the system design objectives. For the static website, mean identification time dropped from 47.2 seconds per element to 14.8 seconds. The SPA category showed the most pronounced improvement, with mean time reduced from 89.6 seconds to 26.3 seconds, reflecting the added complexity of dynamically rendered components where DevTools navigation is particularly cumbersome. The enterprise portal showed a 61.3% reduction from 74.5 to 28.9 seconds. Selector stability was assessed by re-running all captured locators after minor UI updates were applied to each test application; Extension-generated selectors maintained a 91.2% pass rate compared to a 64.7% pass rate for manually constructed selectors, confirming the superior durability of the multi-strategy generation approach.

Backend API response times were measured under simulated load using Apache JMeter, with concurrent user counts of 10, 50, and 100. The Spring Boot REST API maintained an average response latency of 48 milliseconds at 10 concurrent users, rising to 112 milliseconds at 50 users and 187 milliseconds at 100 users, well within acceptable thresholds for interactive use. MySQL query execution times for the most complex locator retrieval queries, involving join operations across the User, Project, and Locator tables, averaged 23 milliseconds with appropriate indexing applied. The frontend React.js application achieved a Lighthouse performance score of 94 on desktop and 87 on mobile, with a First Contentful Paint of 0.8 seconds. These metrics collectively validate the architectural suitability of the chosen technology stack for production-grade deployment scenarios.

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 10, May 2026
Impact Factor: 8.2

TABLE I. Element Identification Time and Selector Stability Comparison

App Category	Manual (s)	Extension (s)	Reduction (%)	Manual Stability	Ext. Stability
Static Website	47.2	14.8	68.6%	67.1%	93.4%
React SPA	89.6	26.3	70.6%	59.2%	89.8%
Enterprise Portal	74.5	28.9	61.3%	67.8%	90.5%
Average	**70.4**	**23.3**	**68.4%**	**64.7%**	**91.2%**

Mean identification time (seconds) and selector stability pass rate across 20 elements per application. Extension-assisted stability significantly outperforms manual construction.

Table I summarizes the quantitative results across all evaluation dimensions. The stability improvement is most pronounced in the React SPA category, where the dynamic nature of component-driven rendering produces particularly fragile absolute XPath expressions under manual capture, while the Extension's attribute-anchored strategies remain resilient to re-render cycles. The API latency measurements confirm that the Spring Boot microservice architecture sustains interactive response times even under moderate concurrent load without requiring caching or CDN optimization at this scale, leaving significant headroom for the planned multi-tenant team workspace expansion. Collectively, these results provide strong empirical support for the architectural and algorithmic design decisions described in the preceding sections and establish a quantitative baseline against which future iterations of the platform can be benchmarked.

V. DISCUSSION

The evaluation results highlight two key advantages of the XPath Finder approach over both purely manual workflows and competing browser extensions. First, the multi-strategy selector generation cascade reliably avoids brittle absolute XPath expressions in favour of semantically meaningful, attribute-anchored locators that survive routine DOM refactoring. Second, the persistent centralized repository transforms locators from ephemeral, session-bound strings into managed, versioned, team-accessible assets, directly addressing the organizational challenge identified by Graham and Fewster (2019) regarding the long-term maintainability of test automation codebases. The JWT-secured multi-user data model ensures that team repositories remain isolated without requiring complex organizational hierarchies, making the system practical for agile development teams of varying sizes.

Compared to XPath Helper, which requires manual expression construction and provides no persistence, the XPath Finder Extension delivers an automation-first experience where the engineer's role is reduced to a single click followed by a semantic naming action. Compared to SelectorsHub, which monetizes advanced features through a paid tier, the XPath Finder platform is fully self-hosted and open to organizational deployment without per-seat licensing costs. The current implementation does have limitations: the content script injection model requires page reload to activate on certain Content Security Policy-restricted sites, and the selector generation engine does not yet account for shadow DOM encapsulation, a growing pattern in modern component libraries such as Lit and Stencil. Both limitations are tracked as priority items on the development backlog and are expected to be resolved in the next iteration of the extension. Nevertheless, across the tested application population, the system performed reliably and delivered consistent, measurable productivity gains.

From a software engineering process perspective, the XPath Finder Extension introduces a new class of developer tooling that bridges the gap between in-browser interaction and backend data management. While browser extensions have traditionally operated as lightweight, stateless utilities, this architecture demonstrates that integrating a Chrome Manifest V3 extension with a full-stack web application creates a qualitatively different capability: persistent, collaborative, and analytically tractable element management. This architectural pattern, where a browser extension serves as a capture interface for a connected web application backend, is applicable beyond selector management and

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 10, May 2026 Impact Factor: 8.2

may serve as a blueprint for future tooling in areas such as accessibility auditing, performance annotation, and in-context content management. The engineering community would benefit from further research into the design space of browser-extension-anchored full-stack applications as a distinct architectural category with its own set of patterns, trade-offs, and best practices.

The security model of the XPath Finder Extension merits particular attention in the context of enterprise deployment. The Manifest V3 architecture imposes strict Content Security Policy constraints on extension scripts, requiring that all remote code execution be replaced by locally bundled logic. This constraint, while initially a migration burden from the Manifest V2 ecosystem, ultimately improves the trustworthiness of the extension by guaranteeing that no dynamically fetched code executes within the browser context. The JWT-based authentication model on the backend ensures that locator repositories remain user-scoped by default, with no implicit data sharing between accounts. All API communications are designed for HTTPS enforcement in production deployments, and the React frontend enforces CSRF protections through same-origin request validation. These layered security measures ensure that the XPath Finder Extension can be confidently deployed within enterprise environments with strict information security policies, including those governed by SOC 2 Type II compliance frameworks or equivalent internal audit requirements. The security architecture provides a solid foundation for the planned multi-tenant team workspace feature, which will introduce role-based access control and audit logging to support organizational governance requirements.

Looking at the broader landscape of developer tooling, the XPath Finder Extension aligns with a maturing recognition in the software engineering community that test infrastructure deserves the same level of engineering investment as production code. Organizations that treat test selectors as first-class assets, subject to versioning, peer review, and systematic quality assessment, consistently demonstrate lower test maintenance costs and higher automation suite reliability over time. The XPath Finder platform operationalizes this principle by providing a production-grade repository infrastructure specifically designed around the lifecycle of web element selectors, from initial capture and naming, through iterative refinement, to export and integration with CI/CD pipelines. As web application architectures continue to evolve, with increasing adoption of micro-frontend patterns, component design systems, and server-side rendering frameworks such as Next.js and Remix, the demand for robust, intelligent element identification tooling will only intensify. The XPath Finder Extension positions itself at the forefront of this evolution by providing an extensible foundation that can absorb new selector strategies and integration patterns as the web development ecosystem advances.

REFERENCES

[1]. W3C, “XML Path Language (XPath) Version 1.0,” World Wide Web Consortium Recommendation, Nov. 1999. [Online]. Available: https://www.w3.org/TR/xpath/

[2]. Selenium Project, “Selenium WebDriver Documentation: Locating Elements,” SeleniumHQ, 2023. [Online]. Available: https://www.selenium.dev/documentation/webdriver/elements/

[3]. React Development Team, “React Documentation,” Meta Platforms, Inc., 2023. [Online]. Available: https://react.dev/

[4]. L. Chen and M. Wang, “Comparative Analysis of Web Element Locator Strategies in Modern Web Applications,” International Journal of Web Engineering and Technology, vol. 17, no. 3, pp. 245–267, 2022.

A. Smith and R. Johnson, “Machine Learning Approaches to Selector Stability Prediction in Dynamic Web Applications,” in Proc. IEEE International Conference on Software Testing, Verification and Validation (ICST), pp. 112–121, 2021.

[5]. M. Rodriguez and P. Kumar, “Locator Strategy Challenges in Single-Page Application Testing Frameworks,” Journal of Software Engineering and Applications, vol. 13, no. 8, pp. 301–318, 2020.

[6]. S. Avasarala, Selenium WebDriver Practical Guide. Birmingham: Packt Publishing, 2014.

[7]. D. Graham and M. Fewster, Experiences of Test Automation: Case Studies of Software Test Automation. Boston: Addison-Wesley Professional, 2019.

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 10, May 2026
Impact Factor: 8.2

[8]. Google LLC, "Chrome Extensions: Manifest V3 Migration Guide," Google Chrome Developers Documentation, 2023. [Online]. Available: https://developer.chrome.com/docs/extensions/migrating/

[9]. Pivotal Software, "Spring Boot Reference Documentation," VMware, Inc., 2023. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[10]. Oracle Corporation, "MySQL 8.0 Reference Manual," Oracle Corporation, 2023. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[11]. Auth0, Inc., "JSON Web Token Introduction," JWT.io, 2023. [Online]. Available: https://jwt.io/introduction

[12]. R. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study," in Proc. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 108–113, 2013