SOLUTION BRIEF

Modernizing the Retail Customer Experience with Full-Stack Testing

KEYSIGHT

eggplant
Test Automation Software

The retail industry is always in a state of flux. To remain competitive, retailers must adapt to changing customer expectations, new digital services, and the demands for a seamless omnichannel experience. Some struggle to adjust because they rely on inflexible monolithic IT systems that are difficult to change, test, and scale.

Modern retailers are shifting from tightly coupled infrastructure to headless architecture and microservices. In a headless architecture, the presentation layer or front end of an application is separate from the back end. Automated software testing is critical to accelerating delivery cycles and keeping up with the pace of change.

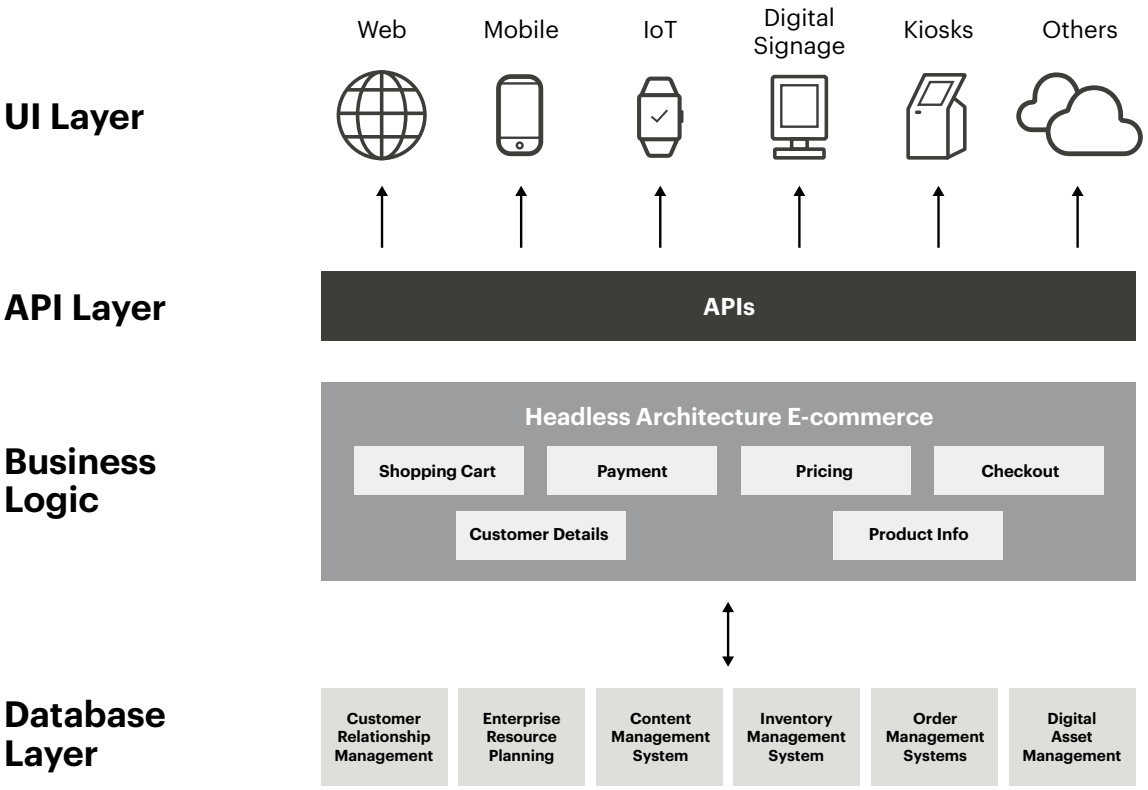

Figure 1. Multi-application functionality is combined into one API and delivered to various user interfaces with headless architecture.

Loosely coupled or decoupled IT architecture, such as a headless content management system (CMS), gives retailers more flexibility to keep up with the latest trends and technologies. So, when demands change, introducing new customer touchpoints, improving functionality, and scaling individual components instead of the entire application is quicker and easier.

Increased agility helps retailers adapt and deliver an enticing user experience, but it does introduce technical complexity. To take advantage of the benefits headless architecture and microservices offer, retailers need to adapt their testing framework and tools.

Headless Architecture and Microservices Need Full-Stack Testing

Headless architecture and microservices require thorough testing because they interact with multiple technology layers of an application, including these:

- Databases – Testing verifies data accuracy and creates complex queries to check schema, tables, and triggers.
- Application programming interfaces (APIs) – APIs link technology layers together to expose application functionality to the end user. Headless architecture turns front-end user interfaces (UI) into a collection of APIs so different applications and servers can gain access.
- Objects (web elements) – These represent user actions in the codebase, so testers must verify accuracy and functionality.
- UI (presentation layer) – Testing at the UI level validates how a piece of software works from the user's perspective. Interface data stored in APIs displays across multiple front ends for various formats (web, mobile, or the Internet of Things) for headless architecture.

Across all these layers, numerous communication paths, the back end to APIs to various UIs, and interactions built with business logic require testing to maintain the end-user experience.

APIs are vital for fast software development and a customer-focused experience

Retailers implementing an API-centric technology stack can choose best-of-breed technologies for third-party integrations to provide a customer-focused experience. In retail, developers can use APIs for inventory management, customer relationship management (CRM), payment, and product information systems. Incorporating these systems delivers a great user experience on an e-commerce website. APIs link all the UI elements with back-end logic and data sources.

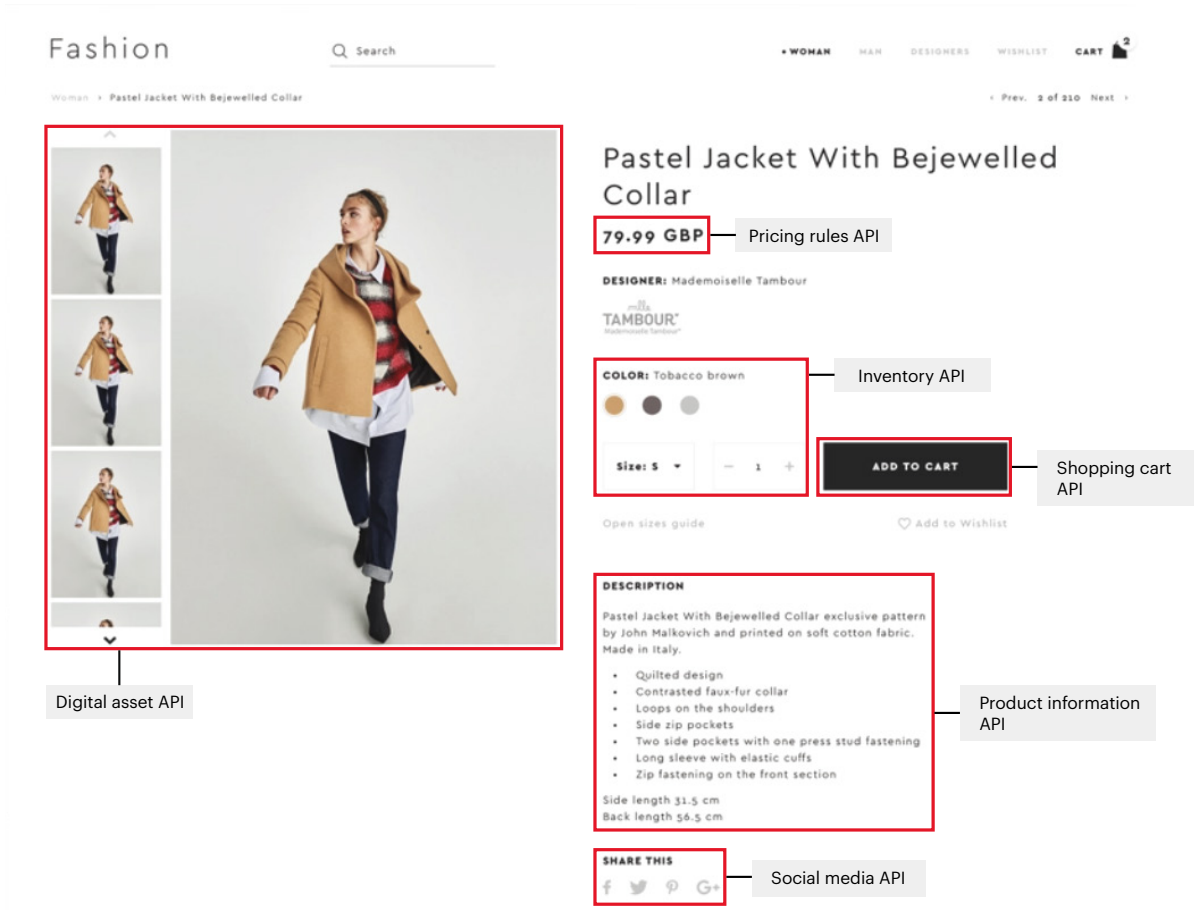

Figure 2. Modern ecommerce websites are built with numerous APIs to provide a customer-focused experience.

For headless architecture, APIs can connect a single back end to third-party integrations and multiple front ends. So, if you are using a headless CMS, APIs can separate content stored in the back end and publish it to various customer channels. That results in faster delivery of content specific to websites, mobile apps, in-store digital signage, and even smartwatches, providing a unified omnichannel experience.

By bridging the gap between technology layers, APIs drive innovation and increase business value. Via API connectivity, developers can interface with any programming language, database, and object layer to modernize applications and deliver new services and functionality to the UI layer.

Using APIs helps retailers implement modular IT infrastructure that delivers agile software development, faster time to market, and a better customer experience. However, interacting with so many different technology layers, systems, and applications makes full-stack software testing complex for quality assurance (QA) teams.

Full-Stack Testing Challenges

Full-stack testing of multiple layers of an application built on diverse technologies is complex. APIs often connect databases, objects, and UI layers in modern applications, but these end-to-end paths are often tested in isolation or by separate teams.

Some teams test only a specific API call or a request from the database layer to verify data accuracy — checking stock levels in an inventory management system, for example. Other testers may validate only how the UI layer visually represents the API call, such as confirming that the number of jeans in stock displays correctly on an e-commerce site. However, if a defect occurs at any layer, the entire data journey will fail, creating an error-strewn user experience.

QA teams must combine verifying the accuracy of API calls to and from back-end systems and validating the correct response through the object layer to ensure that the UI is accurate regardless of the customer touchpoint. APIs don't have a graphical user interface (GUI), so testing every layer together is the only way to confirm whether the end-user visual representation is correct.

Multiple technology layers also combine business logic, custom workflows, and processes. Any instability severely affects how an organization serves its customer and impacts the bottom line.

Testing numerous technology layers requires multiple tools, which quickly causes problems. Some tools will only fire off API calls to check that the response from a back-end system is accurate and that no random events occur. Other tools can only verify the information from the API at the object layer but will not validate the visual representation at the UI, resulting in a poor user experience. Testing the UI across various devices, browsers, and operating systems requires another set of tools, increasing the number of test cases and maintenance. QAs may even need separate tools to test web and mobile applications.

Implementing multiple tools for every test case is possible, but maintaining the scripts quickly becomes a barrier to optimizing workflows and productivity. Spiraling recurring costs and tools that no one uses because of skills gaps will also slow software development cycles and increase time to market.

Optimize Full-Stack Testing with Eggplant DAI

Keysight's Eggplant Digital Automation Intelligence (DAI) is one solution that combines multiple technology layers and tests them together. Eggplant DAI orchestrates end-to-end testing by creating scenarios that verify API requests from a database through the object layers. It then validates the visual response at the UI.

The Eggplant automation engine normalizes result sets used for test execution, including XML, JSON, and YAML. It simplifies the file extension by automatically detecting the API request. Not only can Eggplant query and manipulate databases through SQL commands, but it can also handle the data locally in your script for use later in the test case. Eggplant can then execute different API calls and validate the data transfer and UI functionality through testing, regardless of the programming language.

Eggplant incorporates all data journeys across every layer using a model-based testing approach to re-create business-driven scenarios and replicate what the user sees in production. It helps you identify any potential issues at the source, ensuring that testing is precise, reliable, and repeatable. By executing automated full-stack testing in a single framework, the Eggplant solution accelerates development cycles, improves the user experience, and reduces the cost associated with multiple tools.

With the Eggplant test automation solution, QA teams only need to deal with one code layer to test the back end, through the object layer, to the front-end UI. Eggplant DAI simplifies full-stack testing across every technology layer of an application and the end-to-end data journey and facilitates accelerated software delivery.

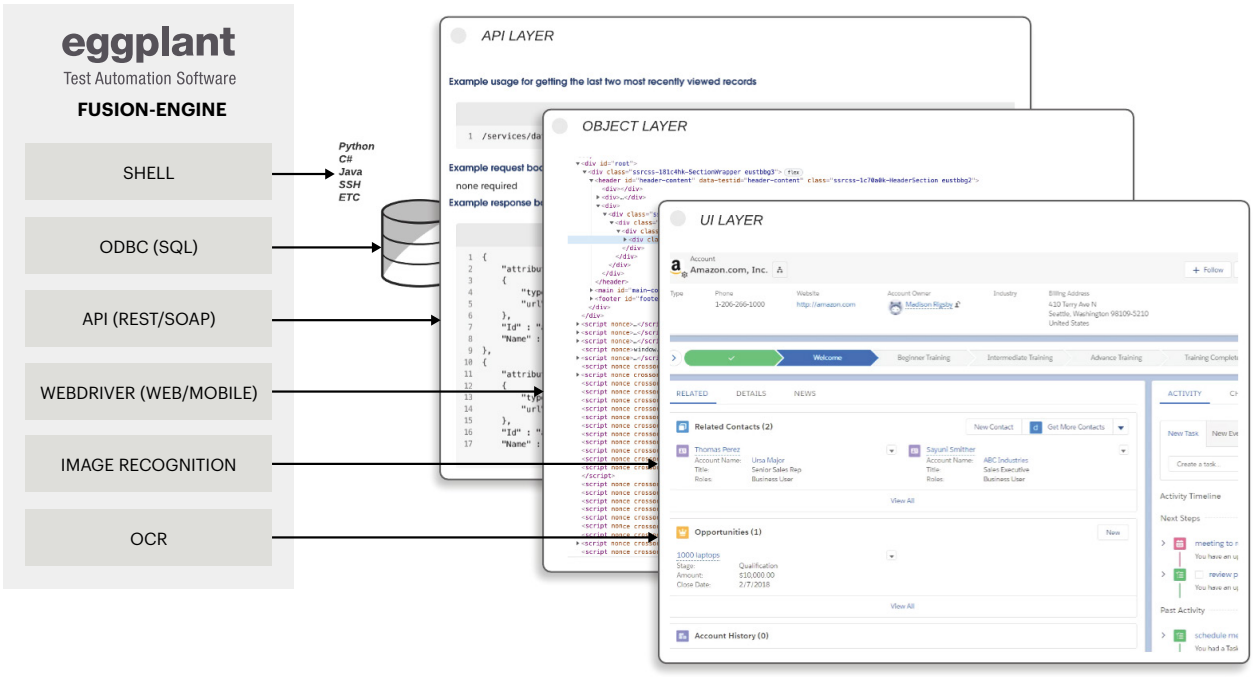

Figure 3. The Eggplant automation engine can test every layer of an application.

To discover how Keysight's Eggplant DAI can optimize full-stack testing for your organization, sign up for a free trial today.