applitools

SOLUTION BRIEF

Taking Component Testing to the next level with **Visual AI**

Applitools Visual AI gives software delivery teams all the benefits of component testing while also going layers deeper with the help of artificial intelligence.

Overview

Component testing is an important part of speeding up your testing pipelines while still delivering quality user interfaces (UIs) in software development. However, relying on component testing as the only testing of the GUI presents multiple shortcomings in a complete test strategy. Teams that solely test the UI/UX of their application through component testing can expect to ship more bugs, ultimately slowing down the delivery of a quality application that it was meant to speed up in the first place.

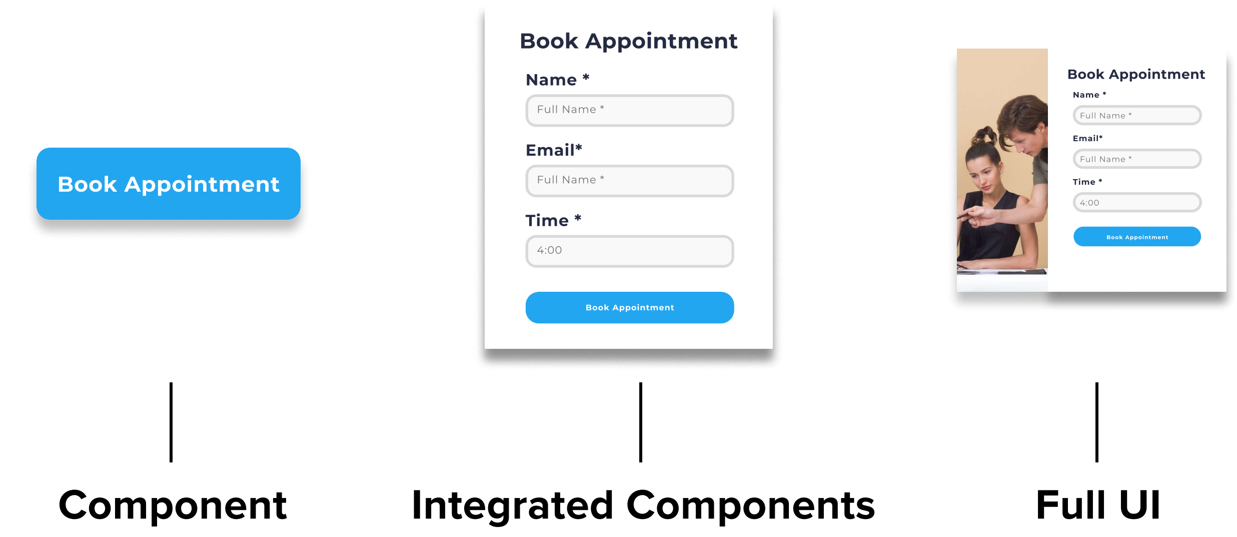
Visual AI, a test automation technology from Applitools, is the only way teams can reliably build a complete quality strategy. Visual AI enables total test coverage, from components to full-page experiences, while requiring even less resources and time.

What are UI Components?

Components are the building blocks of a modern frontend, enabling teams to build consistent UIs, quickly. These blocks are created in isolation and then are integrated into a full screen of an application. They include elements such as buttons, text boxes, check boxes, drop-down menus, images, and others. UI components play a crucial role in determining the look and feel of an application and provide both designers and developers a consistent and intuitive way for users to interact with an application.

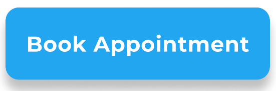
BENEFITS OF COMPONENTS

Reusability:	UI components can be reused across multiple pages or applications, reducing the amount of code that needs to be written and making development more efficient.
Modularity:	Components can be developed and tested independently, making it easier to isolate and fix issues.
Consistency:	By using the same components across an application, the user interface can be made more consistent and familiar, improving the user experience.
Separation of Concerns:	Components can be designed to manage specific tasks, such as data handling or presentation, which helps to separate the concerns of different parts of the application.
Better Maintenance:	With a component-based approach, updating a single component does not affect the entire application, making maintenance easier and less prone to errors.
Improved Scalability:	By breaking down an application into smaller, reusable components, it becomes easier to scale and adapt to changing requirements.
Ease of Collaboration:	Component-based development makes it easier for multiple developers to work on the same codebase, as they can focus on individual components without affecting other parts of the application.



Shortcomings of Component Testing

Component tests are best used as smoke tests that should be run early and often. These tests can cover a wide range of single components at once to help validate if any visual or even functional errors have appeared during regression testing. But they should only be a piece of your total quality strategy, as they miss broader scenarios.

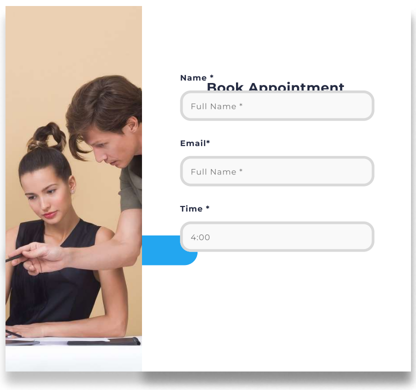


THIS COMPONENT WORKS!

CUSTOMERS INTERACT WITH INTEGRATED, FULL UIs

Component testing does not take into account the overall user experience (UX) of a UI; it fails to validate integration of components, bugs introduced outside of the specific component, and other layout defects that occur when components are combined.

This is because Component testing focuses on testing individual components in isolation, rather than how they interact with each other and with the user. This means that issues with the overall flow and usability of a UI may not be caught until later stages of testing or even until it hits production, and the customer experience has already been damaged.

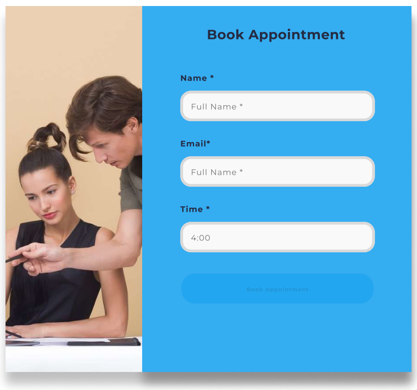


THAT BUTTON LOOKS SLOPPY!

ACCESSIBILITY TESTING IS INCOMPLETE BEFORE INTEGRATION

Another shortcoming of only doing component testing is that it does not allow you to fully test for accessibility issues. Accessibility is an important consideration for UIs, as it ensures that the software can be used by people with disabilities and sensory impairments.

While it may be used in isolated environments and issues, component testing does not typically allow testers to fully validate accessibility issues, such as color contrast, which can only be tested when components are combined into a full UI.

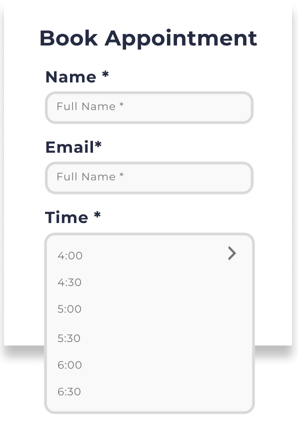


I'M UNABLE TO SEE THE BUTTON!

FUNCTIONAL TESTING OF COMPONENTS IS SHALLOW

While some frameworks do allow for functional testing of components, it's based around "unit" testing concepts and only verifies shallow functional events. Component testing is really best for quickly validating visual regressions, rather than validating full functionality. Sure, it can cut time down for the way that Unit tests get executed quicker, but it won't significantly improve test coverage.

This is why component testing is little more than Unit testing, which while necessary, wouldn't be the only testing strategy used by high-quality development teams. Teams should rely on integration testing and end-to-end testing to make sure they have full-coverage for all customer scenarios.



I'M UNABLE TO CLICK THE BUTTON AND THE DROPDOWN WON'T GO AWAY!

An example would be when clicking a button, the user is supposed to be redirected to a specific URL to expose a Dashboard or Settings page. How would component testing be able to tell if the navigation of the web was successful? It can't.

Why Visual AI

In order to ensure the overall quality and usability of a UI, it is important to also do other types of testing in addition to component testing. Visual AI allows you to do functional, visual, and accessibility testing all at once on both components, full-page, and even end-to-end scenarios. Many times, teams move to component testing to reduce the noise, or false positives, of pixel-to-pixel matching tools, but Visual AI gives you the best of both worlds.

Visual AI identifies, groups, and maps each individual element used in a UI — whether it's a component, a region of a UI, or the complete page. That means Applitools Eyes can scan and validate thousands of elements across components or web, mobile, & desktop applications in just seconds.

By grouping these elements and understanding the context of their connection, Visual AI also enables automatic analysis and maintenance. When an error is found in a component across different browsers, or when a bug is found in a component that is used in another - it will grouped together and presented as one failure, needing only a single update.

applitools

Benefits of Visual AI

Greater Test Coverage
Visual AI tests each element in a UI individually and as a whole - giving you fast test coverage of your entire app.

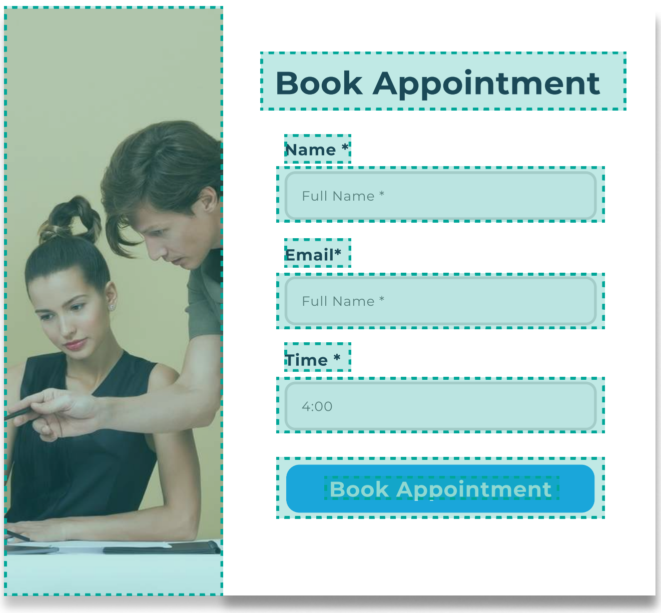
Functional, Visual, Accessibility
With a single test of a UI, Applitools is able to validate correctness with functionality, accessibility, and UX.

Less False Positives
Accurately test for changes that affect the customer experience, not just every single pixel that shifts.

Less Test Flakiness
Visual AI can intelligently heal tests when slight differences in the UI or DOM occur due to frontend changes.

Automated Test Maintenance
Group together similar bugs across different browsers and devices or across multiple integrate components.

Root Cause Analysis
Quickly pinpoint where exactly regressions happened with full context of the DOM.



```
eyesCheck.('form')
```

Validate every element and experience on the page with a single command.

applitools

Sign up for Applitools Eyes today to get started with Visual AI

Get Started