Testing Plan

ARGO UX Master Component Library

ABSTRACT

This testing plan document covers a comprehensive approach for testing multiple system use cases and user interface components. Test scenarios are designed to validate system functionality and ensure that user interface components conform to specifications. The plan categorizes test cases based on whether they relate to the system or individual components. Methods for generating tests are also discussed. Overall, this testing plan document provides a comprehensive framework for testing multiple system use cases and user interface components.

Authors

Kevin Roa, Aliah De Guzman, Saud Baig,

Noah Turrin, Te'a Washington, Samuel Anozie,

Co-Authors

Mark Bentsen

Has every member of your group contributed to this document and participated in the project meetings?
Yes

TABLE OF CONTENTS

ABSTRACT **1**
Authors 1
Co-Authors 1
TABLE OF CONTENTS **2**
LIST OF TABLES **3**
INTRODUCTION **4**
REQUIREMENTS & SPECIFICATIONS BASED SYSTEM LEVEL TEST CASES **5**
UI Test Cases 5
System Test Cases 8
TRACEABILITY OF TEST CASES TO USE CASES **9**
TECHNIQUES FOR TEST GENERATION **10**
CONFIGURATION MANAGEMENT **11**
REFERENCES **12**

LIST OF TABLES

List Of Tables

- Table 1. Component Test Cases
- Table 2. System Test Cases
- Table 3. Test Case to Use Case Map
- Table 4. Configuration Management

INTRODUCTION

The purpose of this document is to describe how the components within the UX Master Library will be tested to ensure validity and functionality. Additionally, it describes how the components and tests align with the system use cases

The document first describes the various test cases each component should go through to validate its functionality. In order for a component to be considered adequate, it must align with the described test cases. If the component does not pass all test cases then the component is considered inadequate and need further work to be accepted.

The document then describes the various system level test cases. These are functionalities that the system should provide in order to be considered fully functional.

The document goes on to map how the UI and System test cases coincide with the system use cases described within the requirements documentation.

Finally, there is a brief discussion on the methods in which test cases may be generated for the various components. From a broad perspective, test cases are formed based on the unique design characteristics displayed on the Figma design document, as well as each components' unique parameters.

REQUIREMENTS & SPECIFICATIONS BASED SYSTEM LEVEL TEST CASES

UI Test Cases

ID	Component	Test Cases
C1	Accordion	A. Title is displayed properly B. Contents shows/hides on click
C2	Card	A. Title is displayed properly B. Titlebar displays custom content properly C. Body content is displayed properly D. Footerbar displays custom content properly E. Card displays tabs properly F. Apply custom styles properly
C3	Form	A. Title is displayed properly according to variant B. Displays correct button color based on color prop C. Width of form component rendered correctly D. Padding around form component rendered correctly E. Individual form components rendered correctly in the right sequence
C4	Header	A. Title is displayed properly B. Title font weight adjusts title properly C. Searchbar is aligned properly D. Tab is aligned properly E. Tab displays correct custom labels F. Button is aligned properly G. Menus are aligned properly H. Menu options are displayed correctly I. Time is updated accurately J. Each tab is mapped to open its respective menu
C5	Image	A. Text rendered correctly for appropriate variant B. No text rendered for image only variant C. Image aligned correctly according to variant D. Image size displayed according to height and width props E. Header text displayed with black and bold font F. Subheader text displayed with gray font G. Body text displayed with black font H. Text aligned correctly according to align prop I. Padding around image component rendered correctly J. If available, badge rendered correctly in blue color scheme

C6	Login	A. A login form is displayed B. Username input accepts text C. Password input accepts text D. Project title/logo is displayed properly above the inputs E. Login button is aligned to bottom right of the card F. Changing the accentColor is reflected properly
C7	LoginPage	A. Login component is rendered in the center of the page B. Login component properly displays title/logo C. Properly displays different background color properties D. Properly displays different accentColor properties
C8	Alert	A. Alert icon properly renders on top of child component B. Value is properly displayed within the icon C. Icon can be aligned to the top/bottom of child component D. Icon can be aligned to the left/right of child component E. Value shows + if above the designated max F. Icon is hidden if showZero is enabled while value is 0 G. Icon background displays correct color scheme H. Value text color displays correct color scheme
C9	Tooltip	A. Tooltip displays correctly B. Label displays correctly C. Tooltip can be moved D. The conditions to display the tooltip can be modified
C10	Badge	A. Displays correct color scheme for each variant B. Displays correct label
C11	Button	A. Label is displayed properly B. Variant changes how the button is displayed C. The color of the button is customizable D. Button can be disabled
C12	Checkbox	A. Checkbox is unchecked by default B. Clicking on checkbox changes state C. Displays correct label by checkbox D. Displays correct color of checkbox E. Medium-sized checkbox by default F. State is not disabled by default G. Renders checkbox and label font color as gray when disabled H. Unable to change state when disabled
C13	Dropdown	A. Displays correct label B. Drops down user-specified items in list C. Supports 2 different sizes D. Allows custom width to be set
C14	FilterChip	A. Value is properly displayed within the chip B. Clicking on the X triggers the onClose function C. Clicking elsewhere on the chip triggers the onClick function

C15	List	A. Displays correct list based on variant B. List is just text by default C. Renders list items correctly in sequential display D. Displays correct button color for checklist/ radiolist/ toggle list E. Displays correct background color F. Radio list allows for selection of only one value at a time G. Checklist/radiolist/toggle list buttons work correctly based on state changes
C16	Radio	A. Able to change checked or unchecked state through props B. Correctly renders correct value of radio button C. Displays correct color of radio button D. State is not disabled by default E. State is not error by default F. Renders radio button gray when disabled G. Unable to change state when disabled H. Renders radio button red when error I. Medium-sized radio button by default
C17	SearchBar	A. Displays text field with search icon correctly B. Displays outlined search bar by default C. Changes search bar display based on variant D. Displays correct label in search bar E. Renders search bar gray when disabled F. Unable to change state when disabled G. Renders search bar red when error H. State is not disabled or error by default
C18	TextField	A. Displays correct label if available B. Displays correct helper text if available C. Allows multiple rows of text using multiline and rows props D. Renders text field gray when disabled E. Unable to change state when disabled F. Renders text field red when error G. State is not disabled or error by default H. Displays error when length surpasses maxLength prop I. Changes margin appropriately based on margin prop value
C19	Toggle	A. Displays correct color of toggle button B. State is not disabled by default C. Renders radio button gray when disabled D. Unable to change state when disabled E. Medium-sized toggle by default
C20	LeftNav	A. Nav takes full height of parent container B. All list items are displayed properly C. ListItem icon is displayed properly and is centered D. ListItem text is displayed properly below the icon E. Clicking any ListItem triggers the onChange function F. Clicking any ListItem displays the selection accordingly

		G. Footer button is displayed at the bottom of the nav H. Clicking the footer button triggers the onFooterClick function
C21	Menu	A. Displays the provided options as menu items B. Displays the correct number of options C. Menu closes when the backdrop is clicked D. Default menu closes when a menu item is clicked E. Displays the correct component next to each menu item F. Displays the correct component color
C22	Tab	A. The number of tabs is equivalent to the number of labels B. Renders the correct label for each tab C. Sets the active tab when a tab is clicked D. Displays no tabs when the labels array is empty E. The selected tab has proper indicator styling
C23	Table	A. Supports user-defined columns B. Supports user provided rows C. 3 density values of “comfortable,” “spacious”, and “ “compact” D. Enabling and disabling of various visual elements of the table

Table 1. Component Test Cases

System Test Cases

ID	Test case
T1	Importing and using components in all of the user interfaces
T2	Importing components and passing props to them to customize behavior
T3	Workflow of making change, pushing it, making release, sending to NPM, and installing the newest package version in UIs.
T4	Component is created and sent to the storybook library
T5	User accesses the deployment and is able to view their desired component

Table 2. System Test Cases

TRACEABILITY OF TEST CASES TO USE CASES

A mapping of UI and System test cases to System use cases.

Test Case	System Use Case
ALL UI test cases	Add all system use cases
T1	UC1: Use Component
T2	UC2: Customize Component
T3	UC3: Refactoring Component
T4	UC4: Create Component
T5	UC5: View Component

Table 3. Test Case to Use Case Map

TECHNIQUES FOR TEST GENERATION

First, tests are generated by the appearance of components in the Figma design document [1]. The library component is visually compared to its reference in the company-provided Figma document. These tests check to make sure that the component bears sufficient resemblance to the provided reference. A component fails the test if it cannot be easily identified or matched to its reference.

Second, tests are generated via the component's parameters. Each component has a number of parameters that can be modified. Tests are created for each parameter to test for both clarity of naming as well as ensuring it functions as intended (ex. A "color" parameter should change the color of the component). A parameter fails the tests if its name does not convey its function, or if it does not function properly.

CONFIGURATION MANAGEMENT

Version In	Version Out	Changes	Reviewed By
0.0	1.0	Initial Document Creation	All Group Members

Table 4. Configuration Management

REFERENCES

[1] ARGO. “UTD BHCC Design Library.” *ARGO*. https://www.figma.com/file/vyxYsmCyu5jTVY7wBpkG25/UTD-BHCC-Design-Library-(Copy)-(Copy)?node-id=6%3A2&t=6q0hlEemn9xrZG5o-0 .