Requirements Documentation

ARGO UX Master Component Library

ABSTRACT

This requirements document outlines the design and implementation of a UX Master Component Library. The goal of this project is to provide a centralized repository of reusable UI components that follow established design patterns and principles. This library will serve as a reference for designers and developers, allowing them to easily access and incorporate standardized components into their projects. The UX Master Component Library will play a crucial role in improving the consistency and efficiency of the organization's product development process.

Contributors

Kevin Roa, Aliah De Guzman, Saud Baig,

Noah Turrin, Te'a Washington, Samuel Anozie,

Mark Bentsen

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

Yes

TABLE OF CONTENTS

ABSTRACT **1**
Contributors 1
TABLE OF CONTENTS **2**
LIST OF FIGURES **3**
LIST OF TABLES **3**
INTRODUCTION **4**
USE CASES FOR FUNCTIONAL REQUIREMENTS **5**
GRAPHIC USE CASE MODEL 5
UC1: Use Component 6
UC2: Customize Component 6
UC3: Refactoring Component 7
UC4: Create Component 8
UC5: View Component 9
UC6: Delete Component 10
RATIONALE FOR USE CASE MODEL 10
NON-FUNCTIONAL REQUIREMENTS **11**
1. Usability 11
2. Maintainability 11
3. Extensibility 11
CONFIGURATION MANAGEMENT **12**
REFERENCES **13**

LIST OF FIGURES

List Of Images

- Figure 1. Graphic Use Case Model

LIST OF TABLES

List Of Tables

- Table 1. Use Component
- Table 2. Customize Component
- Table 3. Refactor Component
- Table 4. Create Component
- Table 5. View Component
- Table 6. Delete Component
- Table 7. Configuration Management

INTRODUCTION

The purpose of this document is to describe the goals of the UX Master Library and its intended uses. It will elucidate the various scenarios in which the library will be used. This document will broadly cover various scenarios, however, it will not describe how they will be implemented.

The document first shows a use case diagram and demonstrates how the use cases are related to the end goal. It illustrates the system requirements and depicts the various interactions of the use cases.

It then goes on to describe the various use case scenarios for the system to handle. These scenarios include things such as the participating actors, entry and exit conditions, event flow, exceptions, and special requirements.

From there, it describes the most important non-functional requirements of the system. Each requirement is given a rationale which explains why it is valued within the system.

USE CASES FOR FUNCTIONAL REQUIREMENTS

GRAPHIC USE CASE MODEL

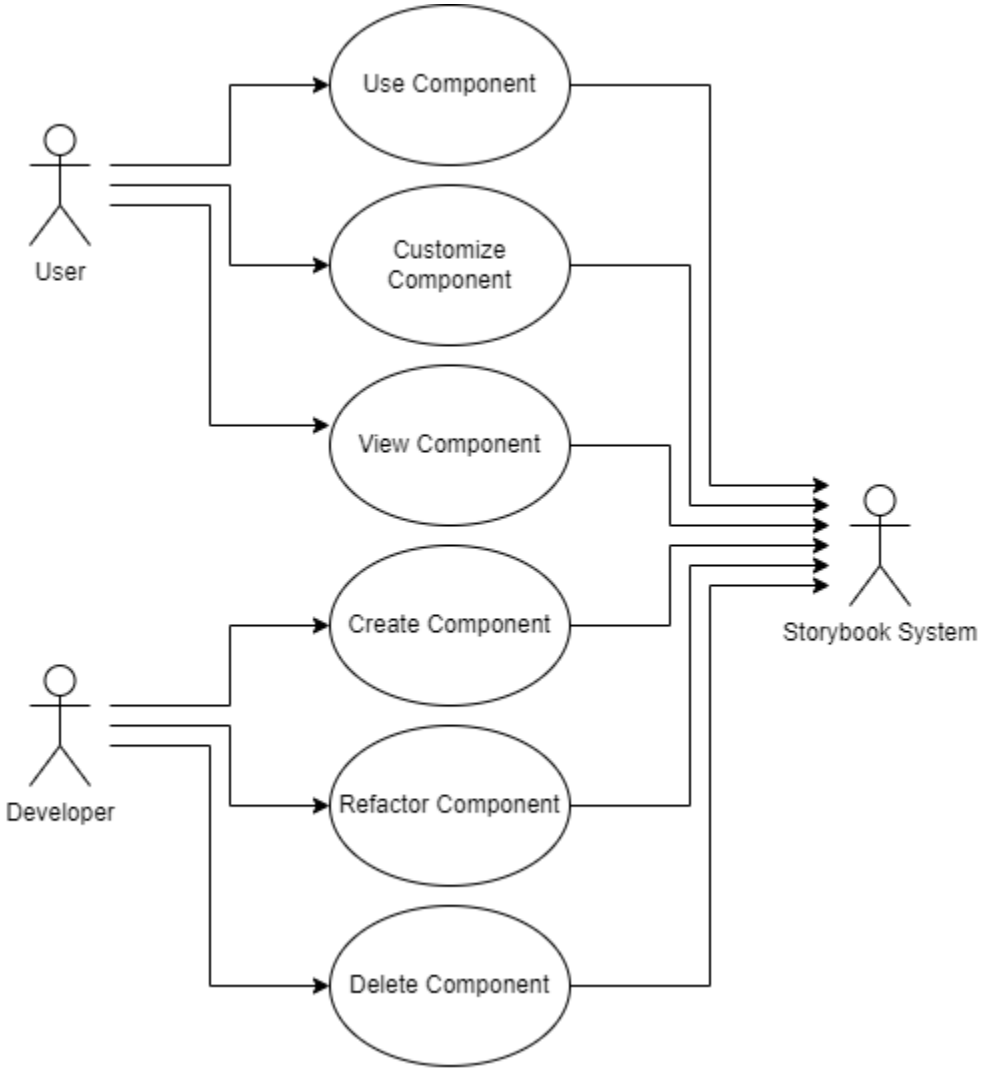

Figure 1. Graphic Use Case Model

UC1: Use Component	
Participating Actors	User, Storybook System
Entry Conditions	- Need reusable front-end component for an application - Component exists
Normal Flow of Events	1. User imports a component from the library 2. User includes the component in their usage 3. User customizes the component to suit their needs
Exit Conditions	- The component is fully functional and operates correctly within the application.
Exceptions	a. Component does not fit users' needs b. Certain aspect of component can't be customized
Extensions	a.1: See if another component handles the problem a.2: Have developer actor create new component for the library b.1: Have developer actor implement the customization capability to the library

Table 1. Use Component

UC2: Customize Component	
Participating Actors	User, Storybook System
Entry Conditions	- Has specific requirements for application front-end design that requires customizable components - Component exists
Normal Flow of Events	1. User imports a component 2. User adheres to implemented design guidelines to customize component
Exit Conditions	- Component is fully customized
Exceptions	a. There may be a time when a user may want to customize a component beyond the implemented design guidelines. b. The customization may not be possible with the current storybook component implementation
Extensions	b.1: See UC3

Table 2. Customize Component

UC3: Refactoring Component	
Participating Actors	Developer, Storybook System
Entry Conditions	- Change in requirements from the overall Argo organization related to front-end design - Developer has access to component design library codebase
Normal Flow of Events	1. Developer modifies files related to component that needs update 1. Developer tests changes 2. Developer requests code change on main component design library 3. Administrator reviews and approves code changes 4. Changes are pushed to live storybook library 5. Projects using the component update to latest version
Exit Conditions	- Component changes are consumed by existing and new users of said component
Exceptions	a. Component design library code may not compile for developer b. Changes introduced by developer can break component design library c. Administrator may not approve code change
Extensions	a.1: Likely caused by developer, diagnose issues with refactor. b.1: Ensure that any changes made to a component still function properly on any project it is used within before making it live. c.1: Various reasons; Work with the administrator to understand why the component was not approved.
Special Requirements	- Changes introduced to component library must adhere to usability and ensure that updated component is easy to use - Changes must also adhere to maintainability and extensibility in case future changes are needed

Table 3. Refactor Component

UC4: Create Component	
Participating Actors	Developer, Storybook System, Development team
Entry Conditions	Need for new component to be stored
Normal Flow of Events	1. Developer creates a component for use in products 2. Developer tests changes 3. Developer requests code change on main component design library 4. Development team reviews and approves code changes 5. Changes are pushed to live storybook library 6. Projects using the component update to latest version
Exit Conditions	- Component is consumed by existing and new users of said component
Exceptions	a. Tests fail b. Review is denied c. Projects use old version of the library
Extensions	a.1: Fix bugs in component code and try again b.1: Conform to project standards and try again c.1: If projects need the new version, update dependency version and use latest
Special Requirements	- Changes introduced to component library must adhere to common design standards

Table 4. Create Component

UC5: View Component	
Participating Actors	User, Storybook System
Entry Conditions	- User should have access to the component library system - User has chosen a specific component to view
Normal Flow of Events	1. User enters the component library 2. User selects a component to view 3. The Component Library System displays the component along with information about it
Exit Conditions	- User is able to see the displayed component
Exceptions	a. If the user is not authorized to view the Component Library System, they will be denied access
Special Requirements	- The Component Library System needs to be able to verify developer identity - The Component Library System should contain 1 or more user interface components

Table 5. View Component

UC6: Delete Component	
Participating Actors	Developer, Storybook System
Entry Conditions	- Developer has ensured the component that they will delete is no longer required for any project - Component to be deleted is not a part of other reusable components
Normal Flow of Events	1. Developer opens the master component library 2. Developer selects the component they want to delete 3. Developer selects the option to delete the component 4. Developer confirms component deletion
Exit Conditions	- The component is removed from the master component library
Exceptions	a. The component is being used in a project
Extensions	a.1: Refactor the project to use an alternative component a.1.1: Remove the current component dependency a.2: Deprecate the component but keep it within the library for some time to give other projects time to migrate
Special Requirements	- The developer is required to get administrator(s) approval to remove a component - Once the developer selects the option to delete a component, it is removed from the library within 5 seconds

Table 6. Delete Component

RATIONALE FOR USE CASE MODEL

Use cases are split between “Users” and “Developers”. Users are individuals who would like to utilize components from the library within a separate project. The storybook system grants them the ability to view/import/customize components. Developers are individuals who maintain the component library; their role is to create/modify/delete components from the system. All use cases connect to the same storybook system because it is built to handle all of those types of interactions. Our job as group 4 is to utilize the storybook library to facilitate the interactions described within the use cases.

NON-FUNCTIONAL REQUIREMENTS

1. Usability

a. The system shall provide easy access for creating/viewing/modifying/deleting components.
b. The system shall provide the ability to easily use its components within other projects.
c. The system shall have documentation for each component deployed.

2. Maintainability

a. The system shall ensure that components are upgradable in the future.
b. The system shall house all components within the same repository.
c. The system shall be built using proper design standards.
d. The system shall be built using open source libraries.
e. The system shall automatically deploy new changes to the library.

3. Extensibility

a. The system shall be able to accept the creation of new components.
b. The system shall be able to allow the modification of old components.
c. The system shall feature components with easily modifiable parameters.
d. The system shall propagate changes across component implementations.

CONFIGURATION MANAGEMENT

Software: Google Sheets

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

Table 7. Configuration Management

REFERENCES

AltexSoft. “Non-Functional Requirements: Examples, Types, How to Approach.” *AltexSoft*, 21 Nov. 2019, www.altexsoft.com/blog/non-functional-requirements/.