

Plattform für crowdsourced Accessibility-Testing von Komponenten aus Web UI-Libraries

Bachelorarbeit

Herbstsemester 2022/2023

Autor: Mathias Lenz

Betreuer: Prof. Dr. Markus Stolze
OST IFS Institut für Software

Experte: Markus Flückiger

Gegenleser: Prof. Dr. Andreas Rinkel



Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Abstract

UI Libraries und interaktive UI-Komponenten werden häufig in Webapplikationen eingesetzt, sind aber bezogen auf Accessibility schwer einzustufen. Es gibt keine spezifischen Accessibility Standards für UI Libraries. Dazu kommt, dass UI Libraries sehr heterogen sind.

Das Ziel der vorliegenden Arbeit ist es, Projekte mit Anforderungen an Accessibility die Auswahl einer UI-Library zu vereinfachen, indem ein Prototyp einer geeigneten Plattform entwickelt wird. Diese Plattform soll es ermöglichen, UI Libraries crowdsourced zu testen und mit den Resultaten Auskunft über die Accessibility der getesteten UI Libraries und deren Komponenten geben.

Mit iterativem und agilen Vorgehen wurden im Rahmen der Analyse die Eckpunkte der Applikation gesteckt. Daraus wurde das Konzept erstellt und die Anforderungen spezifiziert. Die Applikation wurde als Client-Server-Anwendung umgesetzt. Als Client fungiert eine React-Single-Page-Application. Der Server ist mit Node.js und Express implementiert und verwendet eine MongoDB-Datenbank für die Persistierung. Die Programmiersprache ist in beiden Teilen Typescript. Als Schnittstelle fungiert eine REST-API.

Die entstandene Applikation wurde mit Usability Tests und einer Accessibility Evaluation geprüft und validiert. Als Prototyp kann sie damit den angepeilten Verwendungszweck erfüllen.

Die Grundlage für die Entwicklung einer erweiterten Version mit ausgebautem Kriterienkatalog und erweitertem Funktionsumfang für Administration, Deployment, Sicherheit und Performance wurde damit geschaffen.

Management Summary

Ausgangslage

Bei der Entwicklung von Webanwendungen werden häufig Bibliotheken mit vorgefertigten User Interface (UI) Komponenten verwendet, sogenannte UI Libraries (Abbildung 1). Diese werden meist aufgrund des Aussehens und Funktionsumfangs ausgewählt.

Für Projekte mit Anforderungen im Bereich Accessibility ist es schwierig, eine passende UI-Library auszuwählen, da UI Libraries sehr heterogen sind und es keine spezifischen Accessibility Standards für sie gibt. Von den beiden tangierenden Standards zielen die Web Content Accessibility Guidelines auf fertige Webseiten ab und sind darum nur in einzelnen Bereichen für UI Libraries anwendbar, die ARIA-Authoring Practices Guidelines fokussieren sich in erster Linie auf die Spezifikation und Anwendung von zusätzlicher Annotation von UI-Komponenten für assistive Technologie, regelt aber nicht das Verhalten der UI-Komponenten selber.

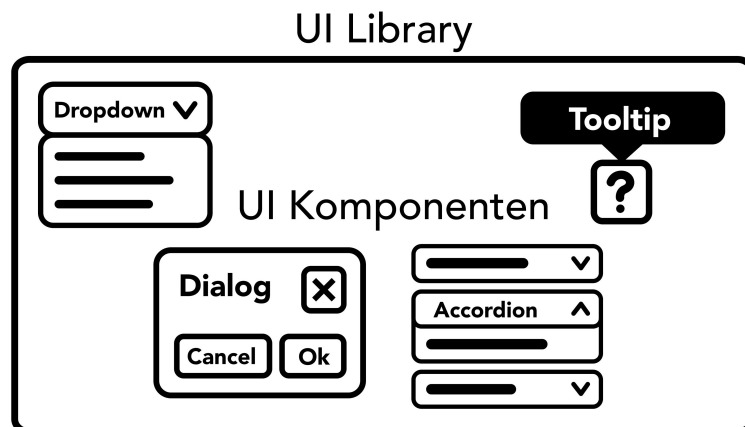


Abbildung 1: UI Libraries bestehen unter anderem aus verschiedenen UI-Komponenten.

Vorgehen

Um Entwicklungsteams bei der Auswahl einer UI-Library zu unterstützen, wurde eine Webanwendung prototypisch entwickelt, die auf einem selbst entwickelten Kriterienkatalog für drei Typen von UI-Komponenten basiert.

Die Kriterien wurden anhand der erwähnten Accessibility-Standards aufgestellt und wurden pro Komponente in zwei Gruppen aufgeteilt. Einmal liegt der Fokus auf der Bedienbarkeit mit der Tastatur und einmal auf der Benutzbarkeit mit dem Screenreader. Die Bewertung wird



Welcome to Project Cactus



Collaborative



Accessibility



Components
from Web UI
Libraries



Testing



cactUS
That's what we
do!

The Mission

"Is my favourite UI library accessible?" - "What UI Frameworks could I choose for my project that gives me a headstart in accessibility?" These are the questions project cactus tries to answer!

[Check out the scored libraries](#)

Abbildung 2: Homepage der entwickelten Webanwendung

anhand der erfüllten und nicht erfüllten Kriterien vorgenommen. Die Anwendung bietet dafür einen Workflow, der ein schnelles und unkompliziertes Testen erlaubt. Jede Komponente und UI-Library erhält einen Accessibility-Score, anhand dessen man ablesen kann, wie gut eine Library im Bereich Accessibility aufgestellt ist.

Um die präsentierten Resultate glaubwürdig zu machen und um die grosse Menge an potentiell zu prüfenden UI Libraries bewältigen zu können, wird auf Crowdsourcing gesetzt. Das heisst, interessierte Personen können selber UI Libraries hinzufügen und Tests beisteuern und der Vorgang ist transparent und nachvollziehbar. Um Testresultate einzelner Testpersonen zu validieren, können Tests mehrfach durchgeführt werden.


Umsetzung

Die umgesetzte Webanwendung basiert auf einer Client-Server-Architektur. Der Client (Abbildung 2) ist für die Interaktion mit dem Benutzer zuständig und kommuniziert mit dem Server, der für die Verwaltung und Auslieferung der Daten zuständig ist. Besucher der Anwendung können sich die Bewertung und Testresultate der erfassten Libraries ansehen (Abbildung 3). Interessierte Besucher können sich engagieren, indem sie sich registrieren, um dann selber Tests erfassen zu können. Die Tests werden anhand des Kriterienkatalogs durchgeführt.


Material UI 100% Library Accessibility Score

[Homepage](#)
[Documentation](#)
 Version: v5.11.1

What do these numbers mean?
100% Accessibility Score (example)
 This is an average score over all the tests that were made for a component, testmode or library. It gives an idea about how good a library performs in terms of accessibility.

 **Dialog (or Modal, Prompt)**
 A dialog is a window overlaid over rest of the page. Users cannot interact with content outside of the dialog.

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

 **Accordion (or Disclosure)**
 An accordion is a vertically stacked set of headings each with a collapsible panel

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

Abbildung 3: Detailansicht einer UI-Library

Fazit

Die entwickelte Webanwendung »Project Cactus« bietet Besuchern Informationen zur Accessibility verschiedener UI Libraries. Somit kann in Projekten die Evaluation einer UI-Library unterstützt werden. Ebenfalls können durch die Webanwendung unterstützt Tests durchgeführt werden.

Der Prototyp wurde mittels Usability Tests und einer Accessibility Evaluation geprüft und konnte im Zuge dessen, abgesehen von verschiedenen kleineren Verbesserungen, validiert werden. Die Grundlage für die Entwicklung einer erweiterten Version mit ausgebautem Kriterienkatalog und erweitertem Funktionsumfang für Administration, Deployment, Sicherheit und Performance wurde damit geschaffen.

Danksagungen

Bedanken möchte ich mich herzlich bei:

Markus Stolze für die hilfreichen Ratschläge und Inputs während unserer Sitzungen.

Josua Muheim für die wertvollen Inputs zu Accessibility und die ermutigenden Worte.

Miro Dietiker für die detaillierte Rückmeldung zu meiner Anfrage.

Michael Gfeller für die ausführlichen und hilfreichen Code Reviews.

den Testpersonen der Usability Tests für Ihre Zeit und Hirnakrobatik.

Inhaltsverzeichnis

Abstract	2
Management Summary	3
Aufgabenstellung	9
1 Einleitung	12
2 Analyse	13
2.1 Ausgangslage	13
2.2 Ziel der Arbeit	13
2.3 Übersicht zum Thema Accessibility	13
2.3.1 Accessibility Standards	14
2.3.2 Herausforderungen rund um Accessibility	15
2.4 UI Libraries und Komponenten	15
2.4.1 OpenUI	17
2.5 Lösungsansatz	17
2.6 Vision	18
2.6.1 Konkurrenz	18
2.6.2 Limitierungen	18
3 Design	19
3.1 Gesamtkonzept	19
3.1.1 Designentscheidungen	19
3.2 Konzept Kriterienkatalog	20
3.3 Scoring-Konzept	22
3.4 Applikation	25
3.4.1 Anforderungen	25
3.4.2 Benutzerführung	28
4 Umsetzung	32
4.1 Applikationsname und Logo	32
4.2 Architektur	32
4.3 Technologien	33
4.3.1 Coding Guidelines	33
4.4 Frontend	33
4.4.1 Verwendete Libraries	34
4.4.2 Evaluation AssisitivLabs	35
4.4.3 Visuelle Gestaltung	35

4.5	Backend	35
4.5.1	API	36
4.6	Datenbank	36
5	Qualitätssicherung	38
5.1	Auswertung Code-Reviews	38
5.2	Auswertung Systemtests	38
5.3	Auswertung Accessibility Evaluation	38
5.4	Auswertung Usability Tests	39
6	Projektauswertung	40
7	Ergebnisse	41
7.1	Evaluation der Requirements	41
7.2	Bezug zur Aufgabenstellung	41
7.3	Empfehlungen zur Weiterführung	41
7.3.1	Zusätzlicher Scope	41
7.3.2	Erweiterter Kriterienkatalog	42
7.3.3	Erweiterte Funktionen	42
7.3.4	Ausbau: Optionale Requirements	43
7.3.5	Weitere Gedanken	44
8	Schlusswort	45
9	Anhang	46
9.1	Systemtests	50
9.2	Accessibility Evaluation	58
9.3	Code Reviews	63
9.4	Code Reports	66
9.5	UI Library und Komponenten Tests	68
9.6	Wireframes	75

Aufgabenstellung Bachelorarbeit HS22 Informatik OST «Plattform für crowdsourced Accessibility-Testing von Komponenten aus Web UI-Libraries»

1. Betreuer

Prof. Dr. Markus Stolze, OST, IFS

markus.stolze@ost.ch

2. Praxispartner

OST, Institut für Software (IFS)

in Kooperation mit Josua Muheim, Accessibility Consultant @ BetterSource josh@josh.ch

3. Student (Einzelarbeit)

Mathias Lenz, mathias.lenz@ost.ch

4. Ausgangslage

Bei der Entwicklung von Web-Anwendungen werden häufig im Web frei verfügbare Webkomponenten wie zum Beispiel Mitteilungs-Dialoge oder gut konfigurierbare Komponenten zur Datumsauswahl genutzt. Die Verwendung dieser Komponenten ist nötig da die in Browsern verfügbaren Standardkomponenten häufig den Ansprüchen moderner Web-Anwendungen nicht genügen. Es fehlen häufig nachvollziehbare Informationen, die die Accessibility solch zusätzlicher Komponenten beschreiben. Gerade die Nachvollziehbarkeit zur Accessibility ist wichtig, da diese sehr unterschiedliche Aspekte, die unterschiedlich getestet werden müssen umfasst (z.B., Tastaturbedienung und Eignung für Screenreader).

Entwickler nutzen bevorzugt Web-UI-Komponenten die im Rahmen von Bibliotheken (Component Libraries, Frameworks, Design Systems) verfügbar gemacht werden. Verschiedene Webseiten bieten einen Überblick über freie und offene Bibliotheken und helfen bei der Auswahl für ein Entwicklungsprojekt (z.B. <https://open-ui.org/design-systems>). Sowohl diesen Übersichtsseiten als auch der Dokumentation vieler Bibliotheken fehlt es an detaillierten Informationen über die Accessibility der enthaltenen UI-Komponenten. Dies betrifft das erwartete Verhalten einer Komponente, zum Beispiel bei der Tastaturnavigation oder bei der Interaktion mit einem Screenreader, und Informationen ob die Komponenten der Bibliothek, in einer bestimmten Version, tatsächlich das erwartete Verhalten zeigen. Diese Informationen wären für die Auswahl von Web-UI-Komponenten aber relevant.

Auf den Webseiten von Standardisierungsgremien (z.B. WCAG: <https://www.w3.org/WAI/standards-guidelines/wcag/> und <https://www.w3.org/WAI/ARIA/apg/patterns/>) finden sich ebenfalls noch keine Informationen über die Accessibility konkreter UI-Komponenten. Es fehlen aktuelle Informationen, die beschreiben, ob eine Bibliothek bestimmte Richtlinien erfüllt und wie getestet werden kann, ob dies der Fall ist. Diese Informationen sind aber für Webentwickler:innen notwendig, die bei der Entwicklung einer Webanwendung entscheiden und verifizieren müssen, ob eine Komponente den Anforderungen in Hinblick auf Accessibility genügt.

5. Ziel der Arbeit

In dieser Arbeit soll ein Prototyp einer Web-Anwendung erstellt werden die

- ... exemplarisch für drei Typen von UI-Komponenten beschreibt, wie eine konkrete Instanz einer UI-Komponente dieses Typs aus einer Bibliothek auf einfache Art auf Accessibility getestet werden kann. Dabei soll der Fokus auf Bedienung mit der Tastatur und Bedienung mittels Screen-Readern liegen. Die Beschreibung sollte sich dabei auf Chrome als Browser, NVDA als Screenreader und Windows als Betriebssystem fokussieren. Es ist zu untersuchen ob eine in Emulation des Screenreaders in Assistiv-Labs (<https://assistivlabs.com/>) sinnvoll ist.
- ... die Möglichkeit gibt, dass sich Personen als Autoren registrieren können und als solche durch einen Test einer konkreten UI-Komponente einer Bibliothek (in einer definierten Version) geleitet werden und die Resultate dieses Tests (mit dokumentierter Version von OS, Browser und Screenreader) automatisch und für alle sichtbar hinterlegt werden.
- ... exemplarisch, unter Nutzung der Funktion für Autoren, dokumentiert wie es um die Accessibility von drei Bibliotheken und drei Typen von UI-Komponenten für eine Kombination von OS-Browser und Screenreader bestellt ist.
- ... eine Übersicht darstellt über die in der Anwendung vorhandenen Bewertungen von UI-Komponenten und Bibliotheken.
- ... Autoren die Möglichkeit gibt für weitere Typen von UI-Komponenten (oder auch OS-Browser-Screenreader-Kombinationen) Kriterien und Vorgehen zum Test der Accessibility zu erfassen. (Optional)
- ... Autoren die Möglichkeit gibt zu vorhandenen Testresultaten Kommentare hinzuzufügen bzw. einen vorhandenen Test nochmals in ihrem Kontext zu replizieren. (Optional)
- ... Personen die Möglichkeit gibt sich eine personalisierte Übersicht über vorhandene Test-Resultate zu erstellen und sich für eine gegebene Auswahl von UI-Komponenten-Typen und Bibliotheken eine personalisierte Vergleichstabelle generieren zu lassen. (Optional)

6. Auftrag

Die folgenden Teilaufgaben sind zu bearbeiten

- Analyse: Erhebung und Dokumentation der Anforderungen
 - Beschreibung der Ausgangssituation. Sammlung und Dokumentation relevanter Informationsquellen
 - Definition wichtiger Nutzungsszenarios
 - Beschreibung der Nutzer und Rollen
 - Funktionale Anforderungen
 - Nicht-Funktionale Anforderungen (z.B. Accessibility, Browser und Screenreader Unterstützung)
- Entwurf: Entwicklung Architektur und Vorschlag UX
 - Definition der drei Typen von UI-Komponenten
 - Definition der drei Bibliotheken und damit der neun zu testenden Instanzen von UI-Komponenten
 - Definition einer sinnvollen Architektur und der genutzten Libraries und Frameworks (mit Lizenzinfos)

- Prototyping UX (Paper, Scenario Walkthrough, ...)
- Definition eines MVP + Hypothesen zu Ausbausritten
- Demonstration Architektur-Prototyp und validiertem UX-Vorschlag
- Implementation und Test (mit Referenz auf FA/NFA)
 - Dokumentation des Erreichungsgrads der FA/NFA
 - Beschreibung weiter Limitation
 - User Tests mit Autoren und «Konsumenten» der Informationen.
 - Empfehlungen und Beschreibungen zu weiteren Ausbausritten mit Aufwandschätzung.
- Sinnvolle Dokumentation des Projektes und der prototypischen Implementation
 - Notwendige Dokumentation entsprechend «Leitfaden»
 - Developer-Dokumentation
 - Projekt-Dokumentation

7. Rahmenbedingungen

Die erstellte Software wird mit einer MIT Lizenz versehen. Der Zeitpunkt und die Modalitäten der öffentlichen Verfügbarmachung des Codes wird mit dem Betreuer abgesprochen. Alle beteiligten können den Code auch vor der Veröffentlichung in Projekten einsetzen.

8. Schlussbestimmungen

Im Weiteren gelten die Bedingungen zu den Informatik Bachelorarbeiten entsprechend dem aktuellen «Leitfaden für Bachelor- und Studienarbeiten Bachelorstudiengang Informatik an der OST»

Das Kriterien-Raster zur Bewertung wird den Studierenden zusammen mit dieser Aufgabenstellung verteilt. Allfällige Anpassungen müssen bis Mitte der Arbeit einvernehmlich beschlossen werden.

Prof. Dr. Markus Stolze

(Dokument ohne Unterschrift)

1 Einleitung

Accessibility im Web ist noch immer ein Thema, das vernachlässigt wird. Dabei haben zwischen 5% und 20% der Bevölkerung ein temporäres oder dauerhaftes Handicap und können von der Förderung von Accessibility profitieren.

Für Webapplikationen werden häufig vorgefertigte, interaktive Web UI-Komponenten aus frei verfügbaren Bibliotheken verwendet. Damit kann Zeit und Aufwand gespart werden.

Es existiert bisher kein Standard und keine verlässliche Informationquelle, die Auskunft über die Accessibility solcher UI-Komponenten geben kann. In der vorliegenden Bachelorarbeit wird ein Prototyp einer Webanwendung entwickelt, der diese Lücke schliessen soll.

Das Kapitel *Analyse* betrachtet die aktuelle Situation der Accessibility und der UI Libraries, um anhand dessen die Anforderungen an die Webanwendung und die Vision ebener zu konkretisieren. Im Kapitel *Design* wird die Anwendung konzeptuell ausgelegt und es werden die Anforderungen definiert. Dazu wird ein User Interface in Form von Wireframes erarbeitet. Dann wird im Kapitel *Umsetzung* dargelegt, wie die Webanwendung aufgebaut ist und welche Technologien verwendet werden. Das Kapitel *Qualitätssicherung* zeigt auf, wie die Webanwendung geprüft wurde und wie die Resultate sind. Zum Schluss wird das Projekt ausgewertet, die Ergebnisse betrachtet und Empfehlungen zur weiteren Entwicklung zu geben.

2 Analyse

Im Rahmen der Analyse wird der Kontext der Arbeit untersucht und nach Faktoren gesucht, die die Rahmenbedingungen der zu erstellenden Anwendung festlegen.

2.1 Ausgangslage

Angenommen, in einem neu gestarteten Projekt soll eine Webseite oder Webapplikation entwickelt werden, und dabei wird Accessibility als Anforderung aufgenommen. Für die Applikation wird also nach einer UI-Library gesucht, die nicht nur die benötigten Funktionen und Komponenten bietet, sondern auch das Thema Accessibility behandelt.

Hier entstehen dann Probleme. Einerseits ist Accessibility noch immer ein Nischenthema. Accessibility ist etwas unübersichtlich und nicht sehr leicht zugänglich, da es ein Thema ist, das unterschiedliche Bedürfnisse versucht unter einen Hut zu bringen. Auf die andere Seite ist die UI-Library-Landschaft sehr vielfältig und ebenfalls unübersichtlich.

Versucht man, etwas über die Accessibility einer konkreten Library oder einzelnen Komponenten einer Library herauszufinden, ist man limitiert durch die Angaben der Hersteller/Entwickler, die meist eher waage und aufgrund fehlender Standards kaum vergleichbar sind. Die Herausforderungen beginnen bereits bei der Heterogenität der UI Libraries, da gibt es unterschiedliche Bezeichnungen für Komponenten mit gleichem Funktionsumfang und gleiche Bezeichnungen für Komponenten mit unterschiedlichem Funktionsumfang.

2.2 Ziel der Arbeit

Im Rahmen dieser Arbeit soll ein Prototyp einer Webapplikation entwickelt werden. Die Webapplikation ist dafür gedacht, dass sich Entwickler und interessierte Personen über die Accessibility von UI Libraries informieren können. Sie soll eine direkte Beteiligung der Nutzer durch das Mitwirken an den Tests ermöglichen. Mithilfe der Applikation werden dann drei Typen von UI-Komponenten aus drei Libraries getestet und damit die Resultate zur Verfügung gestellt.

2.3 Übersicht zum Thema Accessibility

Unter dem Begriff (Web) Accessibility versteht man die Bemühung, das Internet und seine Inhalte barrierefrei zugänglich zu machen. Konkret werden vier Hauptkategorien von Behinderungen unterschieden, für die es Lösungen zu finden gilt. Das wären gehörbezogene, visuelle, motorische und kognitive Behinderungen. In der Tabelle 2.1 werden diese vier kurz vorgestellt,

zusammen mit den wichtigsten Massnahmen, die es Personen mit diesen Einschränkungen ermöglichen oder mindestens erleichtert, sich im Internet zu bewegen.

Einschränkung	Wichtigste Massnahmen
Gehör	Untertitel und Transkripte
Visuell	Ausreichende Kontrastwerte, Skalierbarkeit, Alternativtexte für Bilder
Motorisch	Tastaturbedienbarkeit, Fokushervorhebung, sinnvolle Tabreihenfolge
Kognitive	Einfache Sprache, hohe Fehlertoleranz, Ablenkungen vermeiden u.v.m.

Tabelle 2.1: Unterschiedliche Behinderungs-Typen zusammen mit den wichtigsten Massnahmen, um Betroffenen Informationen zugänglich zu machen.

Um einen Eindruck von der Anzahl betroffenen zu kriegen, bietet das Bundesamt für Statistik hilfreiche Zahlen [9]. Rund 1'551'000 Personen und damit 22.2% der Schweizer Bevölkerung gaben an, von einer Behinderung betroffen zu sein. Eine andere wichtige Kennzahl ist die Anzahl IV-Bezüger mit 218'000 Personen, und damit 4% der Bevölkerung. Ausgenommen der Gehbehinderten sind all diese Personen potentiell auf assistierende Technologie beziehungsweise barrierefreie Webangebote angewiesen.

Es gibt verschiedene Organisationen, die sich für Accessibility im Web engagieren, international ist dies beispielsweise die WebAIM Non-profit Organisation und in der Schweiz gibt es die Stiftung »Zugang für alle«. Dass das Thema im noch Aufholbedarf hat, zeigt der *WebAIM Million Report 2022* [8]. Im Rahmen des Reports werden eine Million der populärsten Webseiten automatisiert auf Accessibility-Probleme untersucht. Über die Million Webseiten wurden insgesamt rund 50 Millionen Fehler gefunden. 96.8% der Webseiten enthielten Fehler. Das wirft kein gutes Licht auf den aktuellen Stand der Accessibility.

2.3.1 Accessibility Standards

Das wichtigste Standardisierungsgremium ist die *Web Accessibility Initiative* (WAI) vom W3C, dem *World Wide Web Consortium*. Die WAI publiziert verschiedene Standards und Guidelines. Für diese Arbeit wichtig sind die *Web Content Accessibility Guidelines* (WCAG) und der *Accessible Rich Internet Applications* (WAI-ARIA) *Authoring Practices Guide* (APG).

Daneben gibt es international verschiedene Gesetze und Richtlinien, die zu einem Teil auf den WCAG basieren [6]. In der Schweiz gibt es den *eCH-0059 Accessibility Standard V3.0* [5]. Dieser stützt sich auf die WCAG 2.1 und gilt für Webseiten der öffentlichen Hand, für staatsnahe Betriebe und für öffentliche Institutionen wie Universitäten oder Spitäler.

Web Content Accessibility Guidelines (WCAG)

Die WCAG [7] sind ein technischer Standard für Web Accessibility und unterstützen Entwickler zum Beispiel bei der Entwicklung von Webseiten oder Webseitenbearbeitungstools wie Content Management Systemen.

Die aktuelle Version der WCAG ist 2.1 (Die neue, noch nicht offizielle WCAG Version 2.2 wurde bei dieser Arbeit nicht berücksichtigt). Der WCAG 2.1 enthält 13 Guidelines mit insgesamt 78 Erfolgskriterien. Diese Erfolgskriterien sind je einer der drei Konformitätsstufen zugeordnet, die

von niedrig A bis hoch AAA reichen. Damit lässt sich bei einer Evaluation oder Anforderung definieren, welcher Konformitätsstufe eine Applikation erfüllen soll. Daraus ergibt sich dann ein Set an Kriterien, die erfüllt werden müssen.

ARIA Authoring Practices Guide (APG)

Die ARIA Authoring Practices Guidelines sind eine Sammlung von Komponenten Patterns, Anleitungen und Beispielen, die Entwickler unterstützen sollen, Web-Komponenten und Widgets barrierefrei zu entwickeln.

Es geht dabei vor allem um die Anwendung ARIA-Attributen innerhalb von HTML Tags. Beispielsweise kann einem `<div>`-Element die Rolle *Button* zugewiesen werden:

`<div role="button">`. Das `<div>`-Element wird dann von Screenreadern als Button angesagt.

Interessant sind im Rahmen dieser Arbeit vor allem die Patterns. Davon gibt es 30 Stück, und sie beschreiben erwartetes Verhalten (im Bezug auf ARIA-Tags) und benötigte ARIA-Attribute. Für die Implementation dieser Tags und das dynamische Anpassen ist der implementierende Entwickler zuständig.

2.3.2 Herausforderungen rund um Accessibility

Warum ist eine gute Accessibility nicht der Standard und weit verbreitetes Wissen unter Entwicklern? Warum sind so viele Webseiten nicht Accessible? Was kann dagegen gemacht werden? Das sind schwierig zu beantwortende Fragen. Das Thema ist vielschichtig und von vielen Parteien abhängig. Es involviert die HTML Standards, die Browser, die Assistive Technology Entwickler, die Accessibility Standards, die Entwickler an der Front und weitere.

Abuaddous et al. [11] haben verschiedene Ursachen in diesem Bereich identifiziert. Der WCAG 2.0 Standard sei schwierig zu navigieren und unvollständig, da die Anwendung des Standards an sich noch nicht ausreichend ist, um Accessibility garantieren zu können. Entwickler seien nicht auf Accessibility sensibilisiert oder nicht motiviert diese umzusetzen, da Accessibility Testing zeitraubend ist.

Puzis et al. [13] argumentieren, dass hohe Kosten für die Implementation von ARIA für Webseiten der Grund sind für die schwache Adaption des Standards.

2.4 UI Libraries und Komponenten

Für die Neuentwicklung von Webanwendungen braucht es häufig UI-Komponenten, die besser konfigurierbar oder stylebar sein müssen, als die nativ verfügbaren HTML-Komponenten. Beispiele dafür sind Dialoge, Kontextmenüs, Switches oder Tooltips. Auch Checkboxes, Selects oder Tooltips werden gerne neu implementiert. Eine Eigenentwicklung ist dennoch aufwändig, vor allem wenn Komponenten robust, accessible, flexibel konfigurierbar und dazu noch ansehnlich sein sollen.

UI Libraries sind daher eine gute Wahl für viele Projekte, um funktional und visuell gut aufgestellt zu sein. Sie bringen aber eine Vielzahl an Herausforderungen mit sich:

Menge Es gibt eine unübersichtlich grosse Anzahl an UI und Component Libraries und Einzelkomponenten. Allein die Suche auf npmjs.com (Stand 29.09.2022) zeigt für *ui framework* 4580, für *datepicker* 3097 und für *ui library* 11810 Ergebnisse. Daneben gibt es noch viele Pakete, die Teilbereiche wie Formulare oder einzelne Komponenten der UI anbieten.

Auffindbarkeit Wenn man nach UI Libraries sucht, sind Suchresultate entweder Libraries oder Blogbeiträge, die Libraries vorstellen. Es gibt aber keine Plattform, die Vergleiche anstellt oder die es einem ermöglicht, sich eine richtige Übersicht zu verschaffen.

Einheitlichkeit HTML5 bietet einen guten Grundstock an Controls und Inputelementen. Aber selbst diese werden von manchen von Grund auf neu implementiert, wie aus einer Umfrage von Greg Whitworth [14] hervorgeht. Die wichtigsten Gründe dafür sind inkonsistentes Verhalten über verschiedene Browser, mangelnde Anpassungsmöglichkeit des Aussehens und um Funktionalität hinzuzufügen.

Für komplexere Komponenten wie beispielsweise einem Multiselect gibt es keine echten Standards. Es gibt die erwähnten ARIA-APG Patterns, die sich jedoch auf die Anwendung der ARIA-Tags fokussieren und nicht als Standard eines Patterns zu verstehen sind.

Dazu gibt es bei UI Libraries verschiedene Variationen einer Komponente und verschiedene Bezeichnungen für gleiche oder sich leicht unterscheidende Komponenten. Daneben sind die Funktionsumfänge einzelner Komponenten unterschiedlich. Ein Beispiel: Ein Dialog wird je nach Library Modal, Dialog, Modal dialog oder Prompt genannt.

Vergleichbarkeit Verschiedene UI Libraries sind schwer miteinander zu vergleichen. Sie unterscheiden sich durch Funktionsumfang: Es gibt Bibliotheken die ungestylte aber funktionale Komponenten anbieten, andere sind reine Styling Libraries, manche bieten Styling und Funktionen. Dann gibt es Libraries, die sich auf ein ****Frontend-Framework**** spezialisiert haben, solche die mit allem Frameworks oder auch ohne verwendet werden können und solche die für jedes Framework eine eigene Implementation haben.

Fehlende Accessibility Qualifikation Für die Einstufung von UI Libraries zu Accessibility gibt es ebenfalls keine direkt anwendbaren Standards. So können UI Libraries selber auch nicht problemlos angeben, dass sie accessible wären. Dokumentationen Libraries behaupten dann gerne, dass sie einfach accessible sind. Andere geben an, dass die WCAG und/oder die ARIA-APG Kriterien erfüllt seien. Wieder andere erwähnen das Thema nicht einmal.

WCAG und APG zielen darauf ab, für die Umsetzung im Endprodukt verwendet zu werden, und nicht in einer Sammlung von Komponenten-Vorlagen. Der WCAG eignet sich darum nicht, weil die Kriterien nicht komponentenspezifisch sind und nicht beschreiben, wie diese sich zu verhalten haben, sondern sich auf die allgemeine Funktionsweise einer Webseite konzentrieren und eher generell gehalten sind. Ein Grossteil der WCAG Kriterien lassen sich auf UI-Libraries gar nicht anwenden. Die APG-Patterns enthalten dagegen andere Kriterien nicht, die gemäss WCAG wichtig wären, wie zum Beispiel die Sichtbarkeit des Fokus.

Eine wichtige Rolle für die Umsetzung der Accessibility im Endprodukt spielt auch die Dokumentation. Wenn die Dokumentation anhand von Beispielen mit korrekt gesetzten ARIA-Tags die Komponenten präsentiert, ist die Wahrscheinlichkeit höher, dass sich Entwickler mit den Tags auseinandersetzen. Wenn dann noch kurz erklärt ist, was diese Tags machen und wozu sie sind, ist das vermutlich die bestmögliche Ausgangssituation.

Somit kann gesagt werden, dass es aus all diesen Gründen schwierig ist, eine passende Library für ein Projekt mit Anforderungen im Bereich Accessibility zu finden und auszuwählen.

2.4.1 OpenUI

OpenUI ist eine Initiative, die danach strebt, einen Standard zu schaffen für UI-Elemente und Controls, die Entwicklern die Gestaltung und Erweiterung erlauben. Damit soll vermieden werden, dass Standardcontrols neu implementiert werden, um das gewollte Aussehen oder die gewollte Funktion zu erreichen. Dafür sollen Komponenten vollständig spezifiziert werden, von Zuständen über Verhalten und einzelnen Bestandteilen bis hin zu den notwendigen Accessibility-Anforderungen. Das Projekt befindet sich zurzeit in einer frühen Phase, ist daher spannend, aber eine baldige Lösung ist nicht in sicht.

2.5 Lösungsansatz

Die geforderte Applikation soll also trotz oder gerade wegen der erläuterten Schwierigkeiten und Hindernisse eine Bewertung und Einschätzung der UI Libraries ermöglichen, damit bessere Entscheidungen getroffen werden können. Dabei spielen die fehlenden Qualifikationen für Accessibility eine zentrale Rolle. Es ist weder das Ziel noch wäre es realistisch, im Rahmen dieser Arbeit einen Standard zu entwerfen, der die Lücke tatsächlich schliessen würde. Es soll aber eine Lösung sein, die einen Beitrag zur höheren Akzeptanz und Adoption von Accessibility leistet.

Um realistischerweise die Menge an UI Libraries bewältigen zu können, bietet sich ein Crowdsourcing an. Die Tests sollen also in erster Linie von einer Community aus Personen durchgeführt werden, denen das Thema Accessibility ebenfalls wichtig ist. Da die Zeit von Freiwilligen wertvoll ist und auch respektiert werden soll, braucht es einen Prozess, der es erlaubt, die Tests auf eine einfache und schnelle Art durchzuführen.

Da es kein zentrales Verzeichnis für UI Libraries gibt, braucht es ein eigenes Verzeichnis. Damit sind die Testresultate an einer zentraler Stelle verfügbar. Dies würde es auch ermöglichen, verschiedene Libraries direkt mit Unterstützung des User Interface miteinander zu vergleichen.

Um die unterschiedlichen Funktionsumfänge der verschiedenen Libraries zu berücksichtigen, soll komponentenbasiert getestet werden. Das heisst, jede Komponente wird separat getestet und gewertet, und aus allen getesteten Komponenten wird dann ein Score für die Library berechnet. Wenn also jemand bestimmte Komponenten benötigt, kann auch nur diese betrachtet werden. Komponentenbasiertes testen ermöglicht daher die Vergleichbarkeit auch zwischen UI-Libraries mit stark unterschiedlichem Funktionsumfang.

Um UI-Komponenten bewerten zu können, müssen sie in Typen eingeteilt werden. Eine absolut homogene Gruppierung in Komponentenpatterns wird nicht in allen Fällen gelingen. Manche Typen haben Subtypen, die sich leicht unterscheiden. Viele Patterns sind aber ähnlich und können in einen Topf geworfen werden. Wenn die Kriterien flexibel sind und von einem Grund-Funktionsumfang ausgehen, lassen sich vermutlich auch solche Unterschiede entschärfen.

Damit eine solche Anwendung, wie sie hier vorschwebt, akzeptiert wird, sind verschiedene Voraussetzungen zu erfüllen. Es braucht eine gute Übersichtlichkeit und Bedienbarkeit, damit Be-

suchen nicht gleich wieder gehen. Die gesuchten Daten müssen schnell gefunden werden. Es kann auch nicht die Anforderung an den gängigen Benutzer sein, dass er Accessibility-Experte ist. Dies soll sich in allen Bereichen widerspiegeln. Im Gegenteil, hier gibt es eine Chance: Werden Besucher der Anwendung sanft an das Thema herangeführt, schwinden unter Umständen Berührungängste. Dazu kommt, dass die Kosten für die Implementation von Accessibility für normale Projekte durch die einfachere Zugänglichkeit von Daten zur Accessibility einer Library gesenkt werden können.

2.6 Vision

Das vorherige Kapitel zusammengefasst ergibt die wichtigsten Eckpunkte für die Applikation und stellt damit die Vision dar:

- Die Applikation soll Tests crowdsourced durchführen.
- Das Testen soll schnell und einfach von sich gehen.
- Die Tests sollen komponentenbasiert sein.
- Die Resultate sollen öffentlich zur Verfügung stehen und verständlich sein.
- Die Plattform soll auch für Entwickler ohne Accessibility-Kenntnisse zugänglich sein.
- Die Komponententypen sollen nach bestmöglich gruppiert werden, ohne den Anspruch einer Standardisierung zu haben.
- Die Applikation braucht eine gute Usability, um akzeptiert zu werden.
- Die Tests sollen bestehende Accessibility-Standards achten.

2.6.1 Konkurrenz

Soweit bekannt ist, gibt es keine ähnliche Plattform. Es gibt einen Blogartikel von Darek Kay [12], der ein ähnliches Konzept hat wie diese Arbeit. Das liegt daran, dass dieser Blogartikel eine der Inspirationen für diese Arbeit war.

Es gibt eine Plattform mit einigen Parallelen, aber anderem Fokus, nämlich A11ySupport.io [2]. Diese Plattform testet, in welchen Browser und Screenreader-Kombinationen welche ARIA-Tags unterstützt werden. Sie ist ebenfalls crowdsourced, basiert aber stärker auf einem Git-basierten Prozess.

2.6.2 Limitierungen

Es ist auf jeden Fall so, dass ein gutes Resultat einer Library nicht garantiert, dass das Endresultat der Entwicklung accessible ist. Ebenfalls ist es möglich, mit einer Library mit schlechten Resultaten eine Webseite zu gestalten, die accessible ist. Es ist aber einfacher, mit einer besseren Ausgangslage ein besseres Ergebnis zu erzielen.

3 Design

In diesem Kapitel wird erläutert, wie die Arbeit konzeptuell gestaltet ist, was die Anforderungen sind und wie der Kriterienkatalog und das Scoring entwickelt wurde und wie das User Interface der Anwendung gestaltet ist.

3.1 Gesamtkonzept

Die Webapplikation soll es ermöglichen, Tests crowdsourced durchzuführen und deren Auswertung darzustellen. Die Webanwendung braucht zwei zusätzliche konzeptuelle Bestandteile: Einmal einen Kriterienkatalog für drei ausgewählte Komponenten und einmal ein System zur Auswertung der Testresultate in Form eines Scoring Konzepts.

3.1.1 Designentscheidungen

Testmethodik

Die Komponenten werden manuell von einem Tester direkt anhand der offiziellen Dokumentation des Herausgebers der Library getestet. Dies darum, weil es eine einfache Möglichkeit ist, die Komponenten ohne Eigenimplementation zu testen. Die Applikation stellt die Kriterien dar und gibt Anleitung und Hilfestellung für die Durchführung des Komponententests.

Die Entscheidung fiel gegen ein automatisiertes Testen, da Accessibility im Allgemeinen schwierig automatisiert zu testen ist [10]. Dies könnte anhand einer standardisierten Implementation der Komponenten vermutlich verbessert werden, wäre aber mit erhöhtem Aufwand für die Tester verbunden. Die Testfälle müssten gegebenenfalls auch noch individuell angepasst werden. Es wurde daher zugunsten des manuellen Testens nicht weiterverfolgt.

Tester

Für das Testing kommen in erster Linie Entwickler im Bereich Frontend in Frage. Das Ziel ist, dass nur geringe Vorkenntnisse im Bereich Accessibility vorhanden sein müssen. Dies soll mit verschiedenen Hilfestellungen und guter Usability erreicht werden.

Die Tester können mit und ohne Erfahrung im Bereich Accessibility, mit oder ohne Behinderung sein. Das heisst auch, die Plattform selber soll accessible sein. Es kann jedoch nicht gewährleistet werden, dass alle Kriterien von allen Personen getestet werden können.

3.2 Konzept Kriterienkatalog

Um zu bestimmen, ob eine Komponente einer UI-Library accessible ist, braucht es Kriterien. Für die verschiedenen Komponenten braucht es demnach einen Katalog an Kriterien, anhand derer die Komponenten geprüft werden. Die Kriterien werden dann mithilfe der Applikation präsentiert und ausgewertet.

Kriterien haben drei Aspekte: Einerseits das zu beurteilende Objekt (die Komponente), dann die Perspektive aus welcher beurteilt wird (der Testmodus), und die Erwartung, was das Objekt zu erfüllen hat.

Auswahl der UI Libraries

Für die Entwicklung der Kriterien und das beispielhafte Testen wurden drei UI Libraries ausgewählt. Voraussetzend ist, dass alle Libraries Komponenten mit eingebauter Funktionalität haben, aktiv unterhalten werden und eine ausführliche Dokumentation haben. Die Auswahl soll zudem bestehend aus einer bekannte Library, einer Library mit Fokus auf Accessibility und einer ohne Fokus auf Accessibility sein.

Ausgewählt wurden:

Material UI Beliebte UI-Library von Google

Radix UI Unstyled und accessible UI-Komponenten-Library

Semantic UI Beliebte Library ohne Fokus auf Accessibility

Auswahl der Komponententypen

In einem ersten Schritt wurden interaktive Komponententypen gesammelt und in vier Gruppen aufgeteilt, wie in Abbildung 3.1 sichtbar ist. Quellen dafür waren die APG-Patterns, die ausgewählten UI Libraries und die Komponentenliste von Open-UI. Die Gruppen sollen Komponenten mit ähnlichen Eigenschaften zusammenfassen. Dann wurden drei Komponenten ausgewählt, die auch in den zuvor ausgewählten Libraries vorkommen und dabei aus unterschiedlichen Gruppen kommen. Die Entscheidung fiel zugunsten der folgenden Komponenten aus:

Dialog Dialog, auch Modal genannt, öffnet ein kleines Dialogfenster und deaktiviert dabei den Rest der Seite.

Accordion Ein Accordion ist ein Widget mit mehreren ein- und ausklappbaren Abschnitten.

Tooltip Ein Tooltip zeigt eine kleine Popup-Info zum aktuell fokussierten oder überflogenen Element an.

Ziele für den Kriterienkatalog

Jedes Kriterium soll dazu beitragen, dass die getestete Komponente gut für Accessibility aufgestellt ist.

Grundsätzlich sollen die Kriterien objektiv sein, damit die Resultate der gleichen Kriterien von verschiedenen Testern eine hohe Übereinstimmung zeigt. Zudem sollen die Kriterien allgemein-

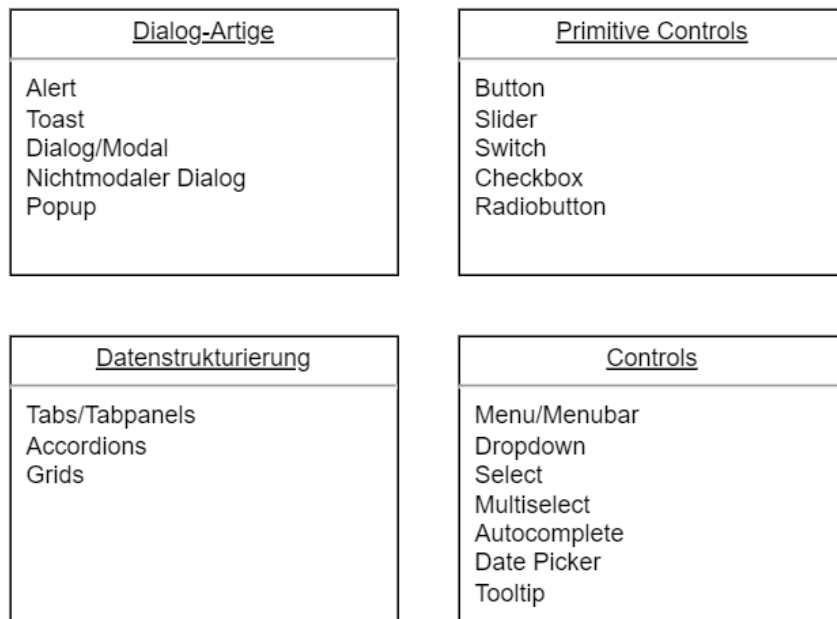


Abbildung 3.1: Komponententypen nach Art gruppiert

gültig sein, das heisst sie sollen auf möglichst viele Varianten der Umsetzung und Präsentation der jeweiligen Komponenten passen.

Die Kriterien sollen die folgenden Bedingungen möglichst gut erfüllen:

- Sie müssen einfach verständlich sein, damit alle Tester das gleiche darunter verstehen.
- Sie müssen mit zumutbarem Aufwand testbar sein, damit sich der Aufwand für Tester im Rahmen hält.
- Sie müssen eine benötigte Accessibility-Funktion oder -Eigenschaft der Komponente prüfen.
- Sie müssen objektiv beurteilbar sein, damit die Testresultate vergleichbar sind.
- Sie müssen allgemeingültig und flexibel sein, damit unterschiedliche Umsetzungen der Komponente von verschiedenen Libraries trotzdem geprüft werden können und vergleichbar sind.
- Sie müssen die Richtlinien des WCAG einhalten und sich am APG orientieren.
- Die Kriterien für eine Komponente insgesamt sollen möglichst den gesamten Funktionsumfang einer Komponente abdecken.

Anatomie des Kriteriums

Das Kriterium besteht aus einem prägnanten Satz, der das Kriterium darstellt und die wichtigsten Informationen enthält. Dazu kommt eine Beschreibung mit Hinweisen für den Tester, was genau gemeint ist und wie es getestet werden kann.

Ein Kriterium soll mit *Ja* (zutreffend), *Nein* (nicht zutreffend) und *Nicht Beurteilbar* beant-

wortet werden können. Die Antwort *Ja* soll zutreffend sein, wenn die Accessibility nicht eingeschränkt ist. Die Antwort *Nicht Beurteilbar* soll gewählt werden, wenn ein Kriterium nicht geprüft werden kann, in etwa weil die zu prüfende Funktion mit der Implementation in der Dokumentation.

Die Kriterien zielen mehrheitlich auf die Funktion der Komponente ab. Deshalb kommen reine CSS UI Libraries etwas kurz, da die meisten Kriterien nicht auf sie angewendet werden können. Das gibt den UI Libraries aufgrund der Auswertungsmethodik jedoch keinen Nachteil.

Testmodi

Die Kriterien werden in verschiedene Gruppen aufgeteilt, die die Perspektive der Nutzung spiegeln. Da ist einerseits Bedienung mit Screen und Tastatur, und andererseits die Nutzung des Screenreaders in Kombination mit Tastaturnutzung. Daher werden die beiden Gruppen *Keyboard* und *Screenreader* benannt. Sie stellen die beiden Testmodi dar.

Quelle der Kriterien

Der APG liefert empfohlene Tastaturbedienungs und Inputs, wie sich ein Element zu verhalten hat, als auch welche Informationen weitergegeben werden müssen.

Der WCAG liefert allgemeinere Patterns und Kriterien, vorallem mit der Tastaturbedienung. Dies sind beispielsweise das Kriterium 2.1.1, das die Tastaturbedienung regelt, oder das Kriterium 1.4.13, das das Verhalten von Inhalten regelt, die bei Hover oder Fokus eingeblendet werden.

Die Kriterien

Im Anhang Abschnitt 9.5 finden sich die Kriterien inklusive Auswertung zu jeder Komponente und jeder Library.

3.3 Scoring-Konzept

Die Auswertung wird anhand der erfüllten und nicht erfüllten Kriterien erstellt. Die nicht entschiedenen Kriterien sollen dabei nicht beachtet werden. Die verschiedenen Tests pro Nutzungsgruppe und Komponente werden aggregiert und zusammen ausgerechnet. Dabei werden unterschiedliche Verfahren und Merkmale verwendet. Der gesamte Ablauf ist im Abbildung 3.2 dargestellt und wird nachfolgend im Detail erklärt.

Eine Klassifizierung analog des WCAG ist nicht vorgesehen, da dies suggerieren würde, dass mit einer Library automatisch eine Webseite erstellt werden kann, die den WCAG Katalog nach einer bestimmten Klasse erfüllt wird.

Für die Auswertung der Tests ist wichtig, dass die Werte vergleichbar sind, und das über verschiedene Ebenen hinweg. Die verschiedenen Ebenen, auf denen Ausgewertet wird, sind:

- Testdurchführung

3.3. Scoring-Konzept

- Testmode
- Komponente
- Version der Library

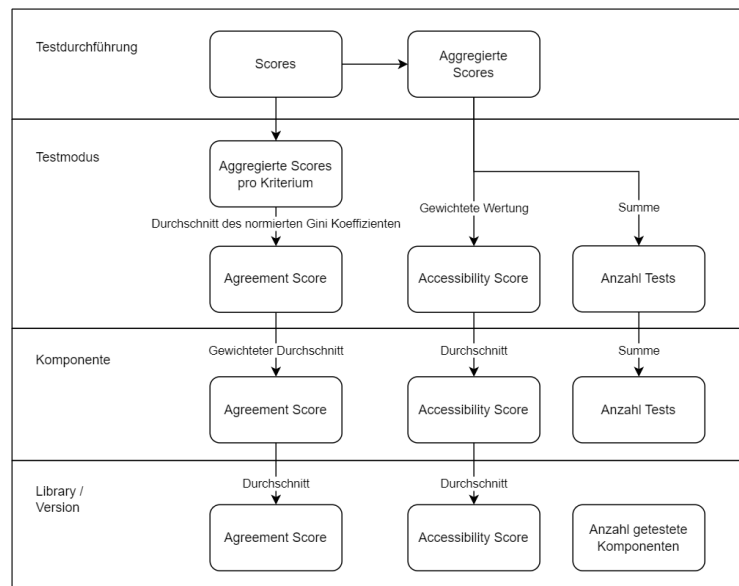


Abbildung 3.2: Modell, wie das Scoring ausgewertet wird.

Testdurchführung

Jede Testdurchführung wird gespeichert und abgelegt. Dabei gibt es ein Resultat (= Score) (Positiv / Negativ / Nicht bewertbar, Kurzform: +/-/0) pro Kriterium. Die Resultate werden zusätzlich in Form eines aggregierten Scores abgelegt. Dabei wird die totale Anzahl von jeder Resultatsvariante berechnet. Das könnte dann bei einem Test mit sechs Kriterien zum Beispiel vier positive, ein negatives und ein nicht bewertbares Kriterien ergeben.

Testmodus

Hier werden die verschiedenen Testdurchführungen aggregiert und pro Modus bewertet. Es gibt folgende Größen:

- Aggregierte Scores (+/-/0) pro Kriterium für weitere Berechnung
- Scores (+/-/0) Total für weitere Berechnung
- Durchschnittlicher Accessibility Score
- Anzahl Tests
- Agreement Score pro Kriterium
- Agreement Score im Durchschnitt

Accessibility Score Jeder Testmodus bekommt einen Accessibility Score. Ein Test hat eine bestimmte Anzahl Kriterien. Um einen Score zu erhalten, werden die Anzahl positiv und negativ bewerteten Kriterien zusammengezählt und davon der gerundete Prozentsatz der Positiven berechnet. Die nicht bewerteten Kriterien werden nicht berücksichtigt.

Beispiel: 4 Positive, 2 Negative, 1 Neutral -> Der Score ist 67%.

Agreement Score Für die Berechnung der Übereinstimmung der Antworten eines Kriteriums wird ein normierter Gini-Koeffizient als Konzentrationsmass verwendet. Der Wert sollte dabei möglichst nahe bei 1 sein, was zeigen würde, dass sich die verschiedenen Tester auf eine Option geeinigt haben. Der Gini-Koeffizient muss normiert werden, weil die Resultate sonst nicht repräsentativ wären. Beim nicht normierten Gini-Koeffizient wäre, bei drei Optionen und Einigkeit auf eine Option, der Wert 0.67 statt 1.

Der Agreement Score wird aus allen Antworten eines jeden Kriteriums berechnet. Um eine Aussage über alle Kriterien eines Testmodes zu machen, wird der Durchschnitt der Gini-Koeffizienten berechnet.

Ein tiefer Agreement Score für ein Kriterium könnte bedeuten, dass es unklar formuliert ist, ein tiefer Agreement Score über mehrere Kriterien könnte zum Beispiel auf Missbrauch hindeuten.

Komponente

Für jede Komponente einer Library werden folgende Kennzahlen berechnet:

- Durchschnittlicher Accessibility Score über die Testmodi
- Anzahl Tests total
- Agreement Score: Gewichteter Durchschnitt aus allen Kriterien beider Testmodi

Library / Version

Für jede Version einer Library werden folgende Kennzahlen berechnet:

- Durchschnittlicher Accessibility Score: berechnet aus dem Durchschnitt des Accessibility Scores der Komponenten
- Anzahl getestete Komponenten
- Agreement Score berechnet aus dem Durchschnitt der Agreement Scores der Komponenten

Anzahl getestete Komponenten Um eine Komponente als *getestet* einzustufen, gibt es verschiedene Möglichkeiten. Man könnte beispielsweise eine bestimmte Anzahl Tests fordern, oder auch einen bestimmten Agreement Score. Einfachheitshalber wird die Komponente als getestet eingestuft, sobald beide Testmodi eine Testdurchführung haben.

3.4 Applikation

Für das Testen der Kriterien und Präsentieren der Resultate braucht es eine Webapplikation. Diese wurde anhand der Resultate der Analyse, der Form des Kriterienkataloges und den Details des Scoring gestaltet.

3.4.1 Anforderungen

Im folgenden Abschnitt sind die Anforderungen für die Applikation spezifiziert. Zuerst werden die Akteure vorgestellt, dann die funktionalen und nicht-funktionalen Anforderungen.

Akteure

Es gibt für Besucher der Webanwendung zwei Rollen. Das ist einmal der Besucher und einmal der Tester:

Besucher Ein Besucher kommt auf die Webseite und möchte sich über Accessibility von einer oder verschiedenen UI Libraries erkunden, oder sich einfach umsehen. Der Zugang kann über die Homepage geschehen oder direkt auf eine detaillierte Ansicht einer Library. Der Besucher kann am Inhalt der Webseite nichts verändern.

Tester Ein Tester kann sich aktiv an der Webapplikation beteiligen, das heisst Libraries hinzufügen und die Tests der Komponenten durchführen. Es wird davon ausgegangen, dass die Person als Besucher auf die Webseite kommt und sich dann entschliesst, sich zu registrieren. Ein Tester kann auch alles machen, was ein Besucher kann.

Funktionale Anforderungen

Die funktionalen Anforderungen wurden in Form von User Stories (US) erfasst. Sie wurden aufgrund der Aufgabenstellung und der Analyse erstellt. Es wurde bewusst auf ein Use Case Diagramm verzichtet, da es keine zusätzlichen Informationen zeigen würde.

- **US 1** Webapplikation Übersicht
 - Ein Besucher kann durch die Webapplikation navigieren und sich über den Zweck der Webseite informieren.
- **US 2** Testresultate ansehen:
 - Ein Besucher/Tester kann eine Übersicht über die Testresultate von allen UI Libraries ansehen. Die Übersicht wird mit dem Scoring der Libraries dargestellt.
 - Ein Besucher/Tester kann sich die Testresultate einer Library im Detail ansehen. Es werden die einzelnen Komponenten mit Scores und bewerteten Kriterien aufgelistet.
 - In der Detailansicht kann zwischen verschiedenen Versionen von Libraries gewechselt werden.
 - Die Testresultate werden anhand des **Scoring-Konzepts** ausgewertet und dargestellt.

- **US 3** Registration und Anmeldung:
 - Ein Besucher kann sich über die Möglichkeit, mitzumachen informieren.
 - Ein Besucher kann sich registrieren und erhält so die Möglichkeit, Inhalte beizusteuern, wird also zum Tester.
 - Ein Tester kann sich an- und abmelden. Es ist erkennbar, welcher Tester aktuell angemeldet ist.
- **US 4** Komponenten testen:
 - Ein Tester kann zu jeder Library neue Komponenten-Tests durchführen.
 - Der Tester kann selber entscheiden, welche Komponente und welchen Modus er testet. Es wird jederzeit angezeigt, was genau getestet wird (Library, Version, Komponente und Testmodus).
 - Es kann und soll grundsätzlich alles mehrfach getestet werden. Der Tester wird informiert, welche Testmodi und Komponenten wie häufig bereits getestet wurden.
 - Der Tester wird informiert und instruiert, wie die Tests ablaufen und was er tun muss.
 - Der Test kann nicht abgeschlossen werden, wenn nicht alle Kriterien ausgewählt wurden.
 - Die Testkriterien werden anhand des **Kriterienkataloges** dargestellt.
 - Die Browserversion und das Betriebssystem werden automatisch erfasst und mit den Testresultaten gesichert.
 - Die Kriterien können mit »Yes«, »No« und »Not decidable« beantwortet werden und bei »Not decidable« mit einem Kommentar ergänzt werden.
 - Nach Beendigung eines Tests kann der Nutzer zurück auf die Homepage navigieren.
- **US 5** UI-Library-Katalog erweitern:
 - Ein Tester kann eine neue Library hinzufügen.
 - Dazu werden die folgenden Metadaten erfasst: Titel, aktuelle Version, Links zu Dokumentation und Homepage.
- **US 6** UI-Library Versionsverwaltung:
 - Ein Tester kann für eine Library zusätzliche neue Versionen erfassen.
 - Für jede Version einer Library werden Komponenten separat getestet und ausgewertet.
 - Die neu hinzugefügte Version wird automatisch als aktuelle Version gesetzt und in der Hauptübersicht angezeigt.

Optionale Anforderungen

- **US 7** Erweiterung des Kriterienkatalogs durch Tester
 - Ein Tester kann für weitere Typen von UI-Komponenten Kriterien erfassen.

- **US 8** Kommentarfunktion für Tester
 - Ein Tester kann zu vorhanden Testresultaten Kommentare hinzufügen.
- **US 9** Vergleichsfunktion für UI Libraries
 - Besucher können Libraries mit einem ausgewählten Set Komponenten-Typen direkt nebeneinander vergleichen.

Nichtfunktionale Anforderungen

Allgemein

- Besucher verstehen, dass die Webseite ein Community-Projekt ist und dass jeder mitmachen kann.
- Besucher können sich ein Bild davon machen, wie die Testresultate entstanden sind und was die Bedeutung des Scores ist.
- Die Plattform ist mit Hinblick auf das internationale Zielpublikum in Englisch gehalten.
- Die Bereiche Performance und Security sind kein Fokus dieser Arbeit, da es ein Proof-of-Concept ist.

Usability

- Formulare geben adäquat Rückmeldung bei Fehlern oder Problemen.
- Die Webapplikation wird, wo sinnvoll, mit Hinweisen in Textform und visuellen Hilfen wie Farben oder Icons unterstützt.
- Buttonbeschriftungen, Titel, Hilfstexte und Fehlermeldungen sind einfach zu verstehen.
- Die Webapplikation ist übersichtlich gestaltet und einfach zu navigieren. Details werden durch Interaktion offenbart.
- Entdeckbarkeit: Es wird stellenweise auf Aktionen hingewiesen, die ein Besucher machen kann, wenn er sich registriert.
- Die Webapplikation soll mit minimalem Vorwissen bedienbar sein.
- Die Webapplikation wird durch User-Tests auf die vorangehenden Anforderungen geprüft.

Accessibility

- Die Webapplikation ist mit Tastatur und mit Screenreader navigier- und bedienbar.
- Entspricht dem WCAG 2.1 auf Stufe AA, Prüfung anhand der *A11y Project Checklist* [1].
- Die Webseite wird mit dem Deque Axe-Core Browserplugin geprüft.

Constraints

- Es soll eine Webanwendung sein, also dementsprechend Web-Technologien einsetzen.

3.4.2 Benutzerführung

Nachfolgend wird das Konzept der Benutzeroberfläche vorgestellt und erklärt, wie damit interagiert wird. Im Anhang Abschnitt 9.6 sind alle, auch die hier nicht gezeigten Wireframes abgelegt.

Öffentlicher Bereich

Auf der Homepage wird, wie auf Abbildung 3.3 sichtbar, ein Besucher mit einer grafischen Repräsentation des Codenamen (*Project*) *Cactus* über den Sinn und Zweck der Seite aufgeklärt. Zudem gibt es kurze Testabschnitte, die das gleiche Ziel haben. Mit einem auffallenden Call-to-Action Button wird der Nutzer zu den Resultaten der UI-Library tests geleitet.



Abbildung 3.3: Wireframe der Homepage von Project Cactus

In der Libraries Übersichtsseite (Abbildung 3.4) sind die verschiedenen Bibliotheken aufgeführt. Man erkennt anhand der eingeblendeten Kennzahlen direkt die wichtigsten Werte zu jeder Library. Ebenfalls wird angezeigt, wenn eine Library noch keine Wertung beziehungsweise Tests hat. Es wird darauf hingewiesen, dass man per Registration auch weitere Libraries hinzufügen kann. Wenn der Nutzer angemeldet ist, kann dieser stattdessen eine neue Library hinzufügen. Im Disclaimer wird erklärt, dass die Scores in erster Linie ein Referenzwert für die Ausgangslage bei Verwendung einer der Libraries sind.

Die Detailansicht dient der genauen Betrachtung der Scores einer Library, wie in Abbildung 3.5 sichtbar ist. Die Library kann um weitere Versionen ergänzt werden, und man kann zwischen den verschiedenen Versionen wechseln. Standardmässig wird die aktuellste Version angezeigt. In der blauen Box sind die Werte, die auf den verschiedenen Stufen (Library, Komponente, Testmodus und Kriterium) verwendet werden, genauer erklärt.

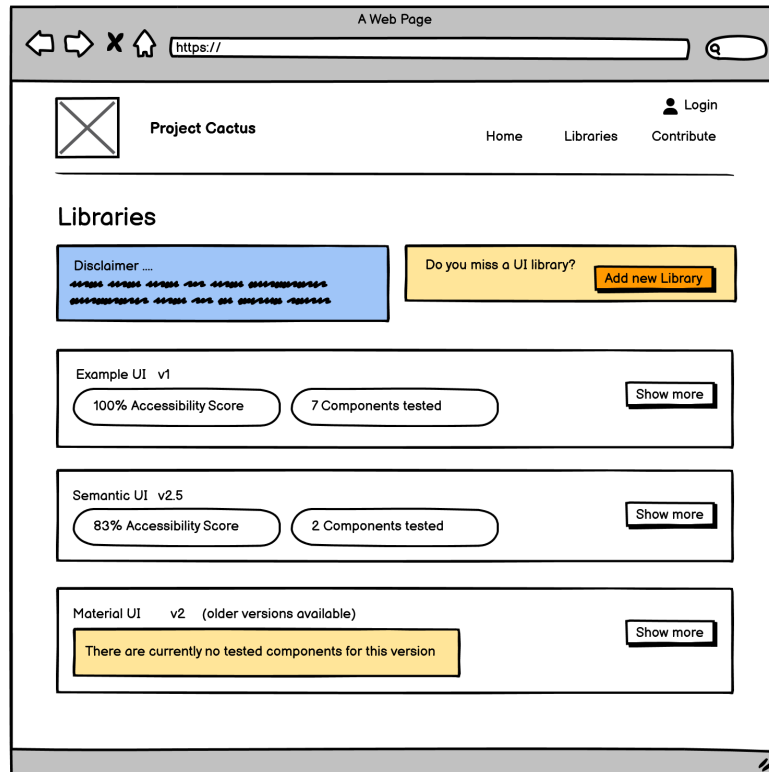


Abbildung 3.4: Wireframe der Übersichtsseite der Libraries

Die Ansicht unterscheidet sich leicht für Besucher und Tester. Besucher sehen einen Hinweis auf die Möglichkeit, UI Libraries hinzuzufügen oder Tests beizusteuern, und werden via einem Link auf die Contibute-Seite gewiesen, wo sie sich registrieren können. Ein Tester kann neue Versionen der Library und Komponenten-Tests hinzufügen. Neue Versionen können direkt auf der gleichen Seite hinzugefügt werden. Für die Tests werden die Tester ins *Testlab* weitergeleitet.

Registration und Anmeldung

Die Registration ist auf der Seite Contribute möglich. Hier wird ein interessierter Nutzer über die Möglichkeit und Zweck der Registration informiert. Nach der Registration ist der Nutzer automatisch angemeldet und kann selbständig weiternavigieren.

Das Login wird mit einem einfachen Formular gelöst, das Usernamen und Passwort verlangt. Der Nutzer wird nach der Anmeldung auf die Library-Übersicht weitergeleitet, damit er gleich loslegen kann. Erreichbar ist das Login als separate Zeile im Menü. An dieser Stelle wird auch der angemeldete Nutzer angezeigt, und man hat die Möglichkeit, sich abzumelden.

Testlab

Das Testlab ist ein separater Bereich der Webseite, der nur als angemeldeter Nutzer zugänglich ist. Es ist in vier Schritte unterteilt. Im ersten Schritt *Testmode* (Abbildung 3.6) gibt es eine Einführung und es wird ausgewählt, welche Komponente und welcher Testmodus verwendet wird. Im zweiten Schritt *Preparation* (Abbildung 3.7) bereitet sich der Tester vor, indem

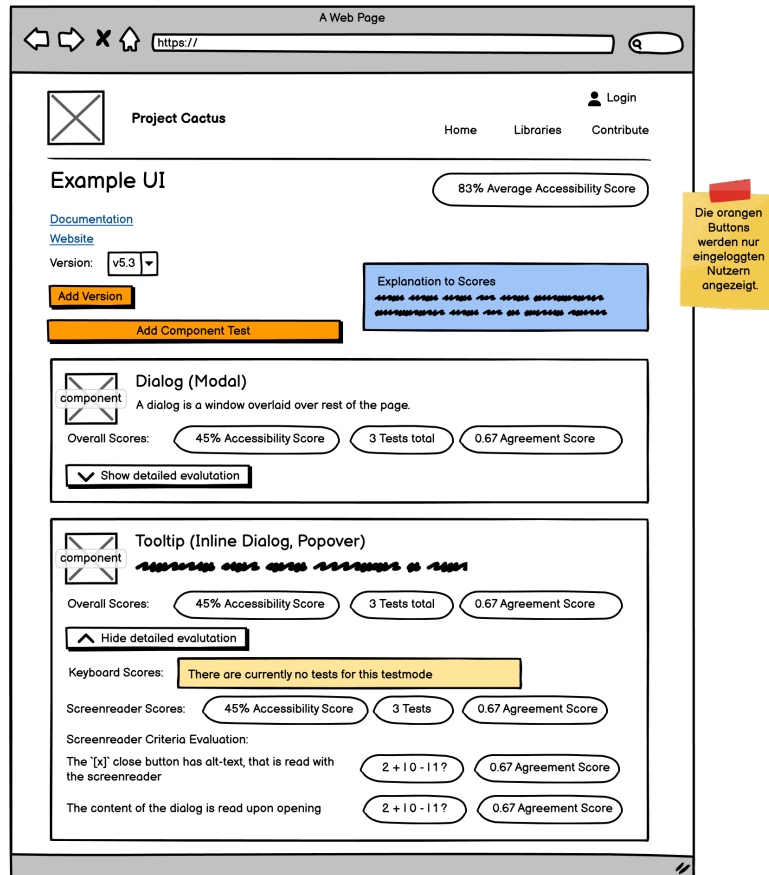


Abbildung 3.5: Detailansicht einer Library

er die Dokumentation öffnet und zur geforderten Komponente navigiert. Im dritten Schritt (Abbildung 3.8) werden die Kriterien aufgeführt, dort muss der Tester dann die Bewertung vornehmen. Mit Abschluss des Tests kommt der Tester zur *Confirmation*, (Abbildung 3.9) von wo aus er zurück zur Detailansicht navigieren kann.

3.4. Applikation

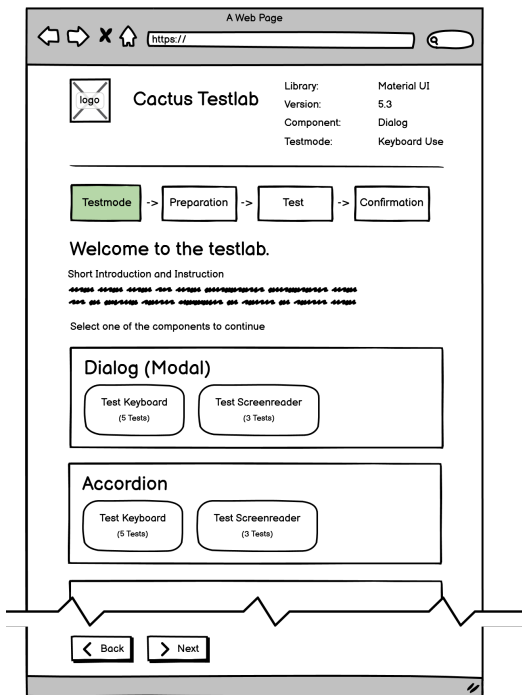


Abbildung 3.6: Auswahl des Testmodus im Testlab

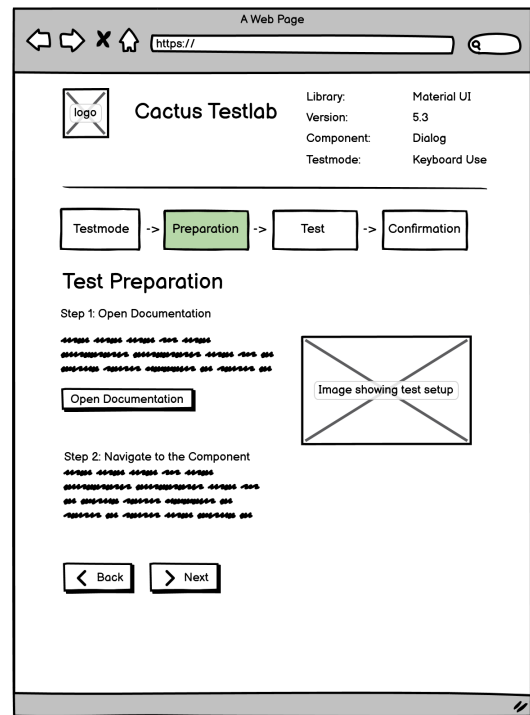


Abbildung 3.7: Instruktionen zum Testen

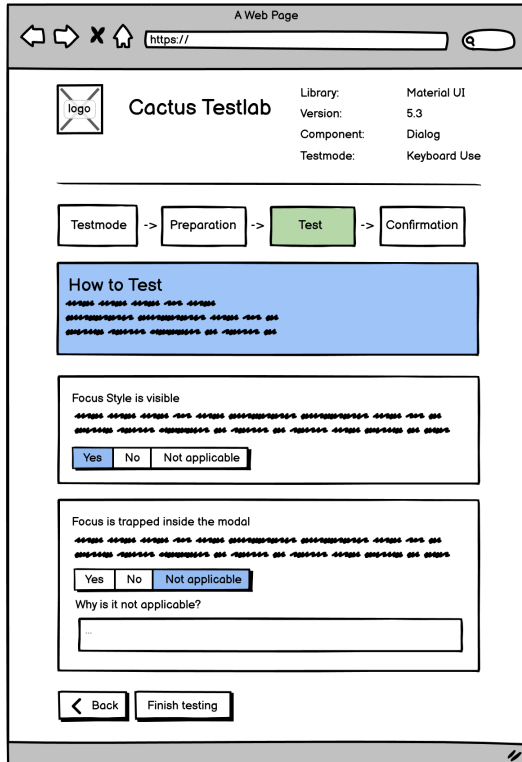


Abbildung 3.8: Das Testformular zum Prüfen der Kriterien

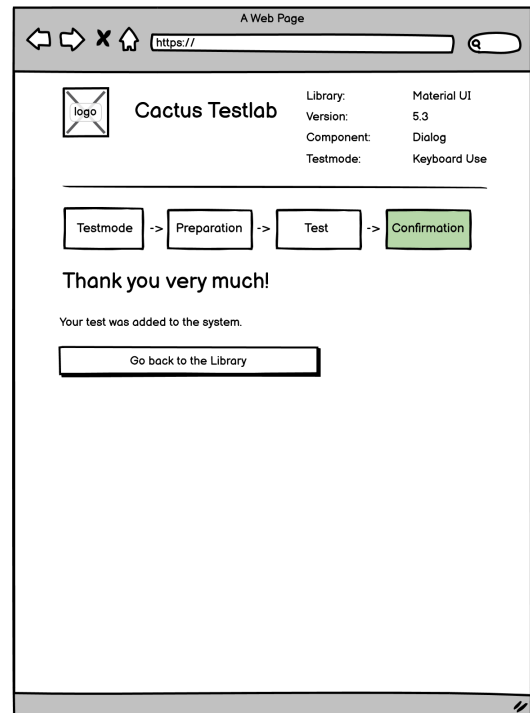


Abbildung 3.9: Bestätigung für den Nutzer

4 Umsetzung

In diesem Kapitel wird die Architektur vorgestellt, die verwendeten Technologien vorgestellt und die beiden Software-Teile im Detail vorgestellt.

Die Bedienung der Software ist im Rahmen eines Video-Walktrough erklärt, das mit dem Bericht abgegeben wurde. Die Installation ist in den README.md Dateien jeweils im Front- und Backend erklärt.

4.1 Applikationsname und Logo

Die Anwendung wurde »Project Cactus« benannt. Cactus besteht aus dem Akronym *CACT*, was für *Collaborative Accessibility Component Testing* steht, und aus dem Suffix *-us*, um daraus ein echtes Wort zu machen. Die Bezeichnung Project kommt daher, dass es ein Projekt ist, das fortlaufend weiterentwickelt und betreut werden muss und dabei keine Kommerziellen Absichten hat. Cactus war dann die Inspiration für das eigens erstellte Logo (Abbildung 4.1), das einen Kaktus mit Sonnenbrille (oder Blindenbrille) im Sand mit einem Strandball darstellt, damit der Auftritt sympathisch wirkt.



Abbildung 4.1: Logo der entwickelten Webanwendung »Project Cactus«

4.2 Architektur

Die Webapplikation ist ein Client-Server-System, wobei der Client ein Web-Client ist. Auf Abbildung 4.2 ist die Architektur visualisiert. Das Backend als Server ist für die Prozessierung der Testergebnisse und die Bereitstellung der Daten zuständig. In der Datenbank werden die Testresultate persistent hinterlegt. Zusätzlich stellt das Backend die Infrastruktur für die Registration und Authentifizierung bereit. Im Frontend, das als Client auf dem Browser des Nutzers fungiert, werden die Testresultate dargestellt, zudem können über das Frontend Tests durchgeführt werden.

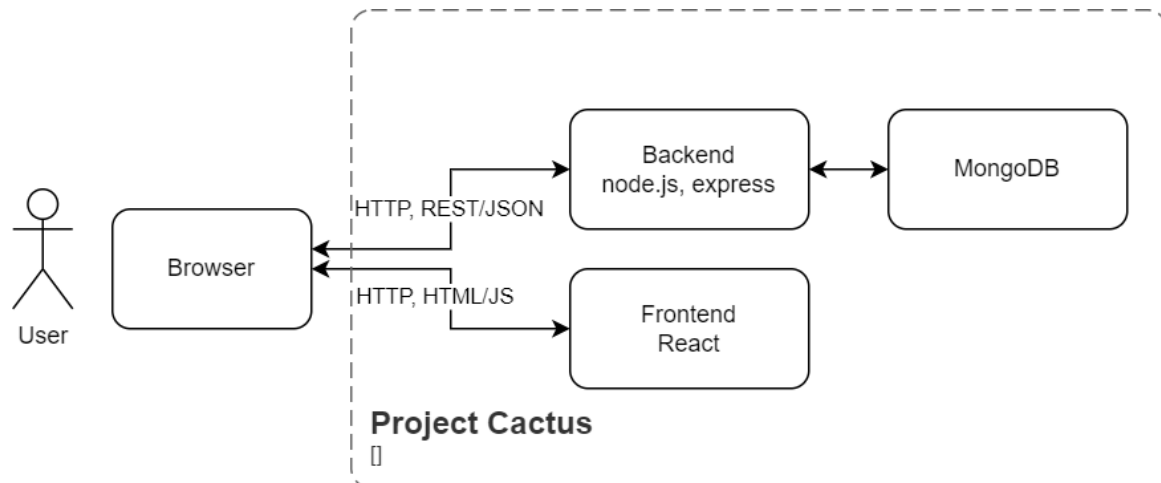


Abbildung 4.2: Architektur der Webapplikation

4.3 Technologien

Die Technologien wurden in erster Linie aufgrund von Vorkenntnissen ausgewählt, da es sich bei der Webapplikation einerseits um ein Proof-of-Concept handelt, und andererseits weil der Projektumfang es nicht erlaubt hat, viel Zeit in Technologieevaluationen zu investieren.

Für das Frontend wird React mit Typescript auf Basis von Node.js verwendet. Dies ist eine bekannte und beliebte Frontend-Library, wo bereits Vorkenntnisse vorhanden sind. React bietet eine hohe Flexibilität in der Implementation.

Für das Backend wird Node.js mit Express und Typescript verwendet. Als Datenbank wird MongoDB und für die Datenbankanbindung wird Mongoose verwendet. Für die Authentifizierung wird ein JWT Token eingesetzt. Bei diesem Setup ist ein wenig Erfahrung vorhanden und es bietet eine relativ flache Lernkurve.

4.3.1 Coding Guidelines

Um eine saubere Code-Struktur zu gewährleisten, wurden auf Front- und Backend die Tools ESLint und Prettier verwendet. Für zusätzliche Unterstützung zur Auffindung von Accessibility-Problemen wurde der axe Accessibility Linter verwendet.

4.4 Frontend

Das Frontend wurde gemäss den erhobenen Requirements und den Wireframes umgesetzt. Für die Umsetzung wurde keine UI-Library verwendet, um einerseits in der Libraryfrage neutral zu sein und andererseits, um die verschiedenen Komponenten flexibel und nach Bedarf konfigurieren implementieren zu können. Das Routing wurde gemäss Abbildung 4.3 eingerichtet. Es erlaubt kurze Wege zwischen allen Funktionen und Informationen.

Die Dateien im Frontend sind wie in Abbildung 4.4 ersichtlich in zwei Hauptordner aufgeteilt. Im Ordner »Shared« sind alle Programmdateien, die von verschiedenen Pages oder direkt von der

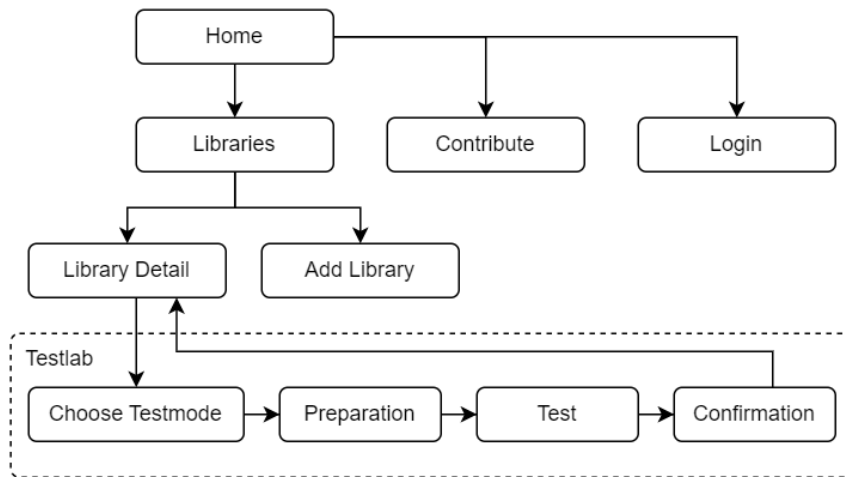


Abbildung 4.3: Routing in React

App.tsx verwendet werden. Im Ordner »Pages« sind alle Seiten modularisiert abgelegt, zusammen mit Unterseiten, spezifischen Subkomponenten und Stylingdateien.

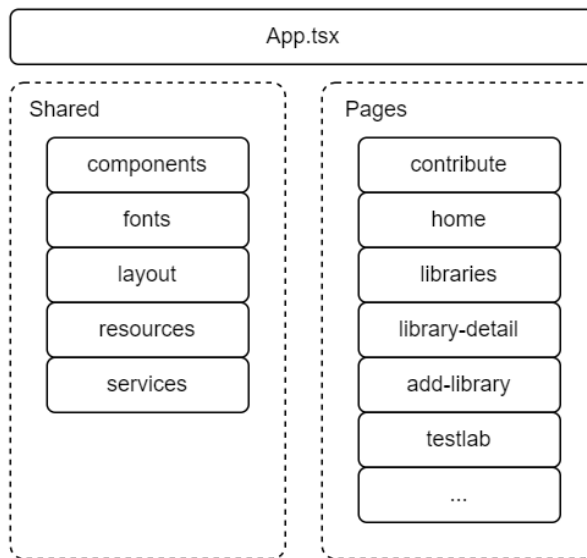


Abbildung 4.4: Die Dateistruktur im Frontend

4.4.1 Verwendete Libraries

Im Frontend wurden wenige Libraries über die standardmässig von Create-React-App eingerichteten hinaus eingesetzt. Es sind:

react-icons ist eine Library, die eine Vielzahl an Icons bereitstellt. Die Icons wurden für Buttons und Alerts eingesetzt.

react-device-detect ist eine Library, die den User Agent des Besuchers ausliest. Mithilfe dieser Library wurde die automatische Erkennung des Browsers und des Betriebssystems gelöst.

@axe-core/react ist eine Library von Deque, die die gerenderte Seite im Blick behält und bei Problemen direkt in die Browser-Konsole Meldungen ausgibt.

4.4.2 Evaluation AssisitivLabs

Assisitiv Labs ist einfach zu bedienen und ist spezifisch für Entwickler gedacht, die sich nicht NVDA installieren können oder wollen. Es ist daher eine gute Sache, um auch von anderen Plattformen aus Accessibility Tests mit allen Screenreadern zu machen. Das Angebot ist 14 Tage kostenlos, danach für einzelne Personen mit 20\$ im Monat im bezahlbaren Rahmen, insofern man es regelmässig benutzt. Für die aktuelle Arbeit wird AssistivLabs als Empfehlung für Tester im Rahmen der Instruktion im Testlab verlinkt.

4.4.3 Visuelle Gestaltung

Die Seite verwendet für eine gute Lesbarkeit schwarze Schrift auf weissem Hintergrund. Um die Lesbarkeit zusätzlich zu erhöhen, wird die Schrift Atkinson Hyperlegible [4] verwendet. Die Schrift wurde vom Braille Institute of America entwickelt und lässt sich auch bei verschwommeners Sicht noch lesen. Für Scores werden stark gerundete Elemente verwendet. Buttons sind durch die leicht kontrastierenden Rahmen, leicht gerundeten Ecken und die Hover-Effekte gut erkennbar. Zusammengehörende Informationen werden durch einen Rahmen visuell gruppiert. Alle Grafiken (ausser den kleinen Icons) wurden selbst gestaltet.

4.5 Backend

Das Backend dient als Datenquelle und Datenverwaltung für das Frontend. Die wichtigsten Aufgaben sind die Benutzerauthentifizierung, das Bereitstellen der UI Libraries mit Wertungsdaten und die Verarbeitung der ankommenden neuen Testresultate.

Im Architekturdiagramm auf Abbildung 4.5 ist ersichtlich, wie das Backend aufgebaut ist. Der Einstiegspunkt ist das Server.js File, das den Server bereitstellt und alles zusammenhält.

Die Unterteilung in Routes, Controllers und Models vereinfacht den Code und trennt unterschiedliche Aufgaben in unterschiedliche Dateien. Routes legen die URLs für die verschiedenen API-Calls fest. Die Controller verarbeiten die Anfragen und fordern Daten bei der Datenbank an. Die Models sind für die Kommunikation mit der Datenbank zuständig.

Es gibt drei Hauptaufgaben im Backend. Der user"der für Login und Registration verantwortlich ist, die Library, die für die Erstellung der Libraries und Versionen zuständig ist und das Testing, das für die Auswertung der Testresultate zuständig ist, zusammen mit den Scoring-Services.

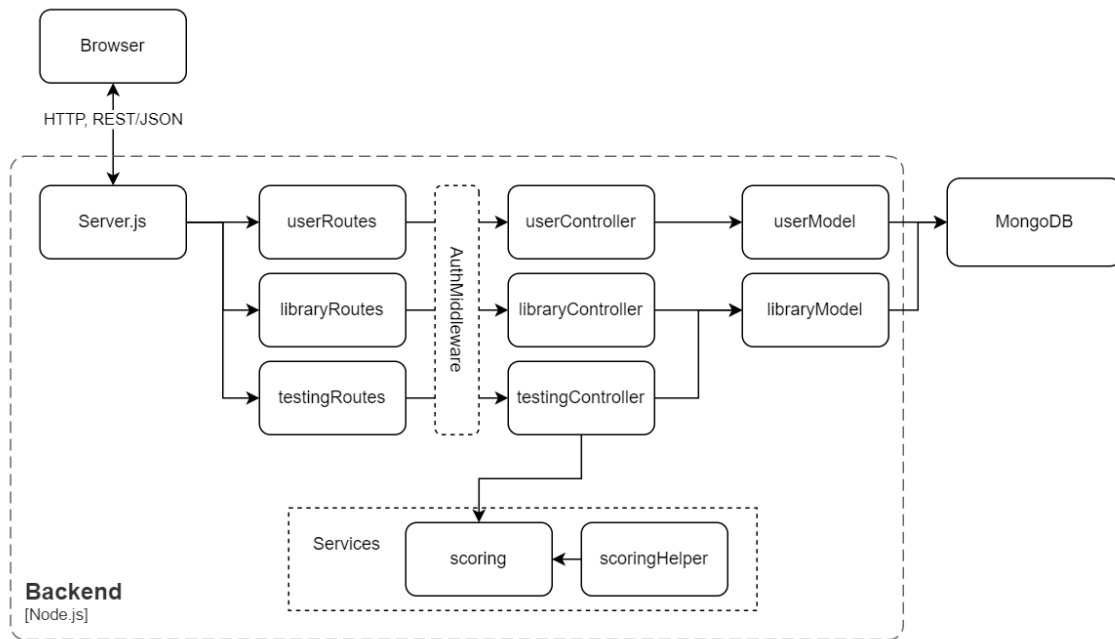


Abbildung 4.5: Architektur Backend

4.5.1 API

Die im Backend implementierte API dient als Schnittstelle für das Frontend, um Daten abzuholen und Änderungen vorzunehmen. Um ungewollte Zugriffe und Änderungen zu verhindern, wurde eine Authentifizierung mittels Java Web Token implementiert. Die API wurde mit Swagger nach dem OpenAPI Standard dokumentiert und ist im Backend-Code abgelegt.

4.6 Datenbank

Das Datenmodell wurde so gestaltet, dass es simpel zu implementieren ist und für die Anwendung im Proof-of-Concept ausreichend Performant ist. Durch die geringe Kohäsion zwischen einzelnen UI Libraries aber sehr hoher Kohäsion innerhalb einer Library eignet sich der NoSQL Ansatz von MongoDB sehr gut.

Da eine MongoDB verwendet wird, ist die Art der Datenspeicherung deutlich anders als bei relationalen Datenbanken. Die Daten werden pro Collection verschachtelt abgelegt. Dies zeigt sich beim Datenmodell in Abbildung 4.6.

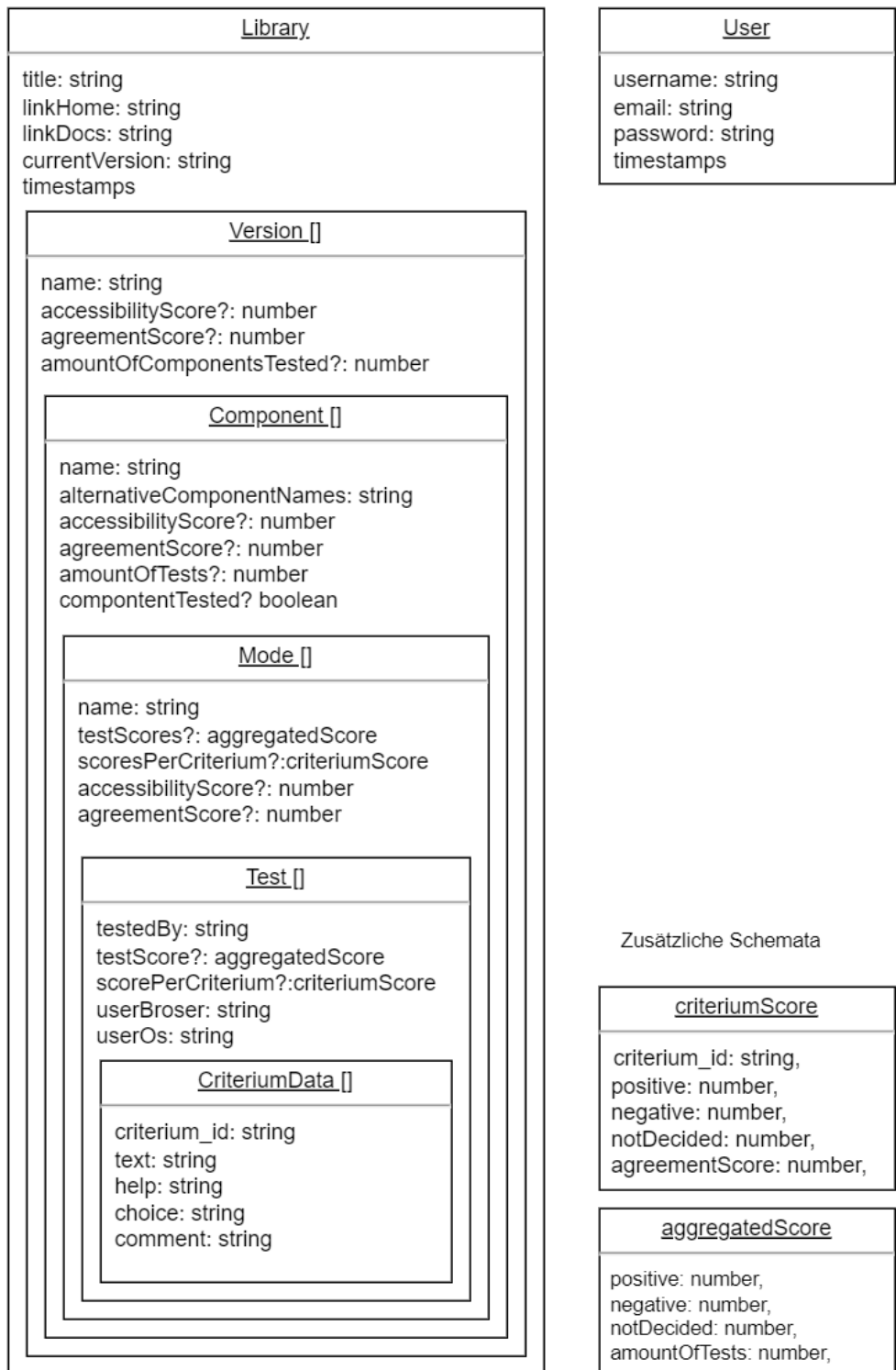


Abbildung 4.6: Datenmodell der MongoDB Datenbank

5 Qualitätssicherung

Die Qualitätssicherung wurde anhand des Projektplanes durchgeführt. Anschliessend sind die verschiedenen Bereiche im Detail ausgewertet.

5.1 Auswertung Code-Reviews

Mithilfe der zwei Code-Reviews der gesamten Codebasis von Michael Gfeller konnten diverse Verbesserungen an Front- und Backend gemacht werden. Der detaillierte Bericht ist im Anhang unter Abschnitt 9.3 zu finden.

5.2 Auswertung Systemtests

Mittels der Systemtests konnte in regelmässigem Abstand der Entwicklungsstand geprüft und die Prioritäten für das Projekt gesteuert werden. Mit dem abschliessenden Systemtest konnten alle Testfälle erfüllt werden. Anhand der Systemtests kann somit ein Teil der Requirements als erfüllt nachgewiesen werden. Die Systemtest Spezifikationen und Protokolle sind im Anhang Abschnitt 9.1 zu finden.

5.3 Auswertung Accessibility Evaluation

Die Webapplikation wurde auf drei verschiedene Arten auf Accessibility geprüft und nach jedem Schritt wurden die gefundenen Probleme ausgemerzt. In einem ersten Schritt wurde die Anwendung anhand des Deque-Axe-Core Plugins auf Probleme untersucht. Dabei kamen nur kleine Probleme wie beispielsweise ein mangelnder Kontrast bei Buttons zum vorschein.

Als nächstes wurde die Anwendung analog des Systemtests getestet, jedoch einmal mit Tastaturbedienung und einmal mit Screenreader. Während die Tastaturbedienung keine Probleme verursacht hat, gab es mit dem Screenreader einige Schwierigkeiten: Da die Anwendung eine Single-Page-Application ist, gibt es keine echten Page-Reloads. Darum wird dem Benutzer ein Wechsel des Seiteninhalts nach einer Navigation nicht bekanntgegeben. Um dies zu beheben, wurde eine Komponente implementiert, die einerseits den Titel der Seite anpasst und andererseits den Fokus nach der Navigation steuert, sodass man als Benutzer weiss, dass man sich nun auf einer neuen Seite befindet. Daneben wurde noch ein Skiplink implementiert und wo nötig ARIA-Live Regions.

Zum Abschluss wurde die WCAG Konformität näherungsweise mittels einer extensiven Checkliste des A11yprojects getestet. Hier wurden keine nennenswerten Probleme mehr gefunden.

Das Vorgehen und das Protokoll finden sich im Anhang Abschnitt 9.2.

5.4 Auswertung Usability Tests

Obwohl nur zwei Usability Tests durchgeführt werden konnten, gab es eine hohe Übereinstimmung / grosse Überschneidungen der Erkenntnisse. Die wichtigsten davon sind fehlende Visualisierungen, zu stark versteckte Hilfetexte und Beschreibungen und nicht ganz deutliche Kommunikation des kollaborativen Gedankens hinter der Arbeit. Die Probleme wurden im Anschluss an die Usability Tests behoben, und drei Feature-Ideen wurden in den Empfehlungskatalog aufgenommen. Hier muss angemerkt werden, dass die Usability Tests nicht den gesamten Funktionsumfang der Anwendung getestet haben.

Die kompletten Testspezifikationen und Protokolle sind im Anhang ?? zu finden.

6 Projektauswertung

Zu Beginn des Projekts wurde ein Projektplan ausgearbeitet, der die wichtigsten Rahmenbedingungen definiert. Dieser wurde im Verlauf des Projekts auf aktuellem Stand gehalten. Der Projektplan und die weiteren Projektunterlagen als auch der Timereport sind im Anhang ?? zu finden.

Ein ein-Personen-Projekt braucht in sich kein Koordinationsaufwand, dafür aber ähnlich viel Planung wie ein Mehrpersonen-Projekt. Dafür wurde eine reduzierte Version von Scrum+ eingesetzt. Dies hat sich bewährt. Die Aufgaben konnten fortlaufend im Wochenabstand geplant und dann umgesetzt werden.

Im Projekt lief gemäss den Meilensteinen bis Mitte Dezember nach Plan. Der Meilenstein 5 wurde mit einer Woche Verspätung erreicht, weil die verfügbare Zeit von einem anderen Modul gebraucht wurde. Der Meilenstein 6 wurde ebenfalls mit einer Woche Verspätung erreicht, dies aufgrund krankheitsbedingtem Ausfall. Die totale Soll-Zeit konnte jedoch erreicht werden und der Meilenstein 7 wurde Plangemäss erreicht.

7 Ergebnisse

7.1 Evaluation der Requirements

Die Ergebnisse des abschliessenden Systemtests zeigen, dass die funktionalen Anforderungen (US 1 bis 6) erfüllt sind. Die optionalen funktionalen Anforderungen (US 7 bis 9) konnten nicht erfüllt beziehungsweise umgesetzt werden. Die nichtfunktionalen Anforderungen wurden anhand der Usability Tests und der Accessibility Evaluation geprüft und sind damit ebenfalls erfüllt.

7.2 Bezug zur Aufgabenstellung

Mit Erfüllung der Anforderungen kann nun der Vergleich zur Aufgabenstellung gezogen werden. Die geforderte Anwendung wurde prototypisch umgesetzt und besitzt die benötigten Funktionen.

Die Aufgabenstellung spricht implizit die möglichen Unterschiede zwischen verschiedenen Browsern und Betriebssystemen an. Diese Thematik, also wie stark sich die Unterschiede im Rahmen der erstellten Kriterien zeigen, wurde nicht untersucht. Durch die automatische Erfassung können jedoch in Zukunft Testresultate dahingehend untersucht werden.

7.3 Empfehlungen zur Weiterführung

Für den Schritt vom Proof-of-Concept zur vollwertigen Applikation die live geschaltet werden kann, sind noch einige Arbeiten vonnöten. Dies bedeutet einerseits ein Ausbau der nichtfunktionalen Anforderungen, aber auch der Funktionen, des Kriterienkatalogs und der Wertung.

7.3.1 Zusätzlicher Scope

Der Aufwand wird für die untenstehenden Massnahmen um die 120 - 160 Stunden geschätzt.

Sicherheit Die Applikation soll gegen übliche Angriffsvektoren im Web abgesichert werden. Daneben soll auch möglicher Missbrauch der Plattform selber möglichst eingedämmt werden.

Performance Um eine erhöhte Nutzer- und Datenlast zu ermöglichen, müssen die aktuellen Performance-Schwachstellen der Applikation eruiert werden. Ein wichtiger Punkt ist sicher eine Optimierung des Datenmodells. Hier könnte man beispielsweise durch eine Aufspaltung des Library-Objekts auf Ebene der Versionen zu den Komponenten kleinere Da-

tenpakete versenden, damit nicht bei jeder Abfrage der Library-Übersicht die gesamten Testdaten aller Libraries übertragen werden.

Deployment Natürlich braucht es eine geeignete Umgebung, die Applikation für Nutzer bereitzustellen.

SEO Natürlich soll die Anwendung bei Live-Gang dann auch möglichst gefunden werden. Dafür bräuchte es vermutlich noch die eine oder andere Optimierung des Inhalts.

7.3.2 Erweiterter Kriterienkatalog

Der Aufwand wird für die untenstehenden Massnahmen um die 30 - 40 Stunden geschätzt.

Kriterienkatalog Beim Kriterienkatalog sind bisher drei Komponenten erfasst. Um UI Libraries aber möglichst umfassend zu testen, bräuchte es jedoch Kriterien für alle gängigen Komponentenpatterns.

Dynamischer Kriterienkatalog : Der Kriterienkatalog wird aktuell statisch mit dem Frontend ausgeliefert. Man könnte ihn aber auch ins Backend integrieren. Dies würde es erlauben, mit einem geeigneten UI, den Kriterienkatalog zu erweitern. Es ist aber die Frage, ob man das möchte, oder ob man Änderungen am Kriterienkatalog eher mit einem besser kontrollierbaren und koordinierbaren Werkzeug wie beispielsweise Issues und Merge-Request löst.

Weitere Testmethoden : Zurzeit werden einige Aspekte der Library nicht getestet, wie beispielsweise Kontrastwerte. Hier wäre es möglich, ein zusätzliches Testverfahren zu machen, dass einige Punkte der gesamten Library testet. Oder man nimmt bei einzelnen Komponenten das Kriterium Kontrast zusätzlich auf.

Unterscheidung Plattformen Die aktuelle Umsetzung konzentriert sich auf Windows, Chrome und NVDA. Wie sieht es mit anderen Browsern und Screenreadern aus? Gibt es Unterschiede, sodass man die Bewertung auftrennen müsste?

7.3.3 Erweiterte Funktionen

Der Aufwand wird für die untenstehenden Massnahmen um die 80 - 100 Stunden geschätzt.

Moderation und Verwaltung Um Anpassungen an den Daten vorzunehmen, fehlerhafte Daten zu korrigieren oder Missbräuche rückgängig zu machen, wäre eine zusätzliches Interface für die Administration hilfreich. Dafür sind eigens Anforderungen zu erheben.

- Zusätzlich sollen alle Änderungen der Daten nachvollziehbar sein, also mit einem Nutzer verknüpft werden.
- In diesem Zuge ist auch zu überlegen, wie mit ungültigen oder unpassenden hinzugefügten Libraries umgegangen wird und wie diese Fälle präventiv, etwa durch Information und Validation verhindert werden können.

Ausbau der Benutzer Das aktuelle Registrations- und Loginprozedere sind rudimentär. Für eine Weiterentwicklung bräuchte es mindestens noch eine Verifikation der Mailadresse, Passwortrichtlinien und eine Möglichkeit, das Passwort zurückzusetzen. Eine andere be-

ziehungsweise zusätzliche Möglichkeit wäre es, das Login via Oauth mit anderen Authentifizierungsstellen zu erlauben, wie beispielsweise GitHub.

Accessibility & Usability Die Anwendung hat zum Ende der Arbeit eine solide Ausgangslage. Da die gemachten Usability Tests nicht den gesamten Funktionsumfang abbildeten, könnte sich eine zusätzliche Serie Usability Tests lohnen. Mit fortschreitender Entwicklung ist sicher ein Monitoring der Accessibility nötig. Auch ein Accessibility-Statement auf der Homepage würde sicher begrüsst werden.

Testing

- Die Option *Nicht entscheidbar* könnte sich als unpopulär herausstellen, da man eine Begründung erfassen muss. Um dem entgegenzuwirken, könnte man eine Auswahl der häufigsten Gründe geben, warum ein Kriterium nicht entscheidbar ist.
- Wenn eine Antwort von der bisherigen Mehrheit der Antworten abweicht, könnte man direkt während des Tests noch eine Begründung verlangen.

Verbesserte Vergleichsmöglichkeiten

- **Direktvergleich:** Zwei oder mehrere Libraries können gegenübergestellt werden, um zu sehen wie die einzelnen Komponenten im Vergleich abschneiden.
- **Filter und Suche:** Für ein einfacheres Zurechtfinden bei einer grossen Anzahl an Libraries braucht es Such- und Filterfunktionen. Die Filter sollen dabei mit Hauptfokus auf Komponenten entwickelt werden und es ermöglichen, nur Libraries mit den ausgewählten Komponenten anzuzeigen.

Erweiterte benutzerbezogene Funktionen

- Tester können in einer eigenen Übersicht sehen, welche Libraries und Komponenten Sie bereits getestet haben.
- Ein einzelner Tester soll nicht mehr als einmal einen Testdurchlauf pro Version und Modus durchführen können. Dies soll auch im Frontend so angezeigt werden. Allenfalls könnte man eine Wiederholung eines Testdurchlaufs erlauben.

7.3.4 Ausbau: Optionale Requirements

Die nachfolgenden Punkte sind als Diskussionsstoff gedacht und werden deshalb nicht mit einer Aufwandschätzung versehen.

- **Knowledge Base:** Eine Knowledge Base stellt Wissen bereit. Im Fall dieser Anwendung würde es darum gehen, Benutzern nahezubringen, wie mit Tastatur oder verschiedenen Screenreadern getestet werden kann. Während dem Projekt war immer wieder die Frage offen, ob es Videoanleitungen für die Tests braucht. Während sich dies so nicht herauskristallisiert hat, könnte es sich für eine Knowledge Base durchaus anbieten, dies in Videoformat bereitzustellen.
- **Community:** Mit der Registration werden Nutzer nicht nur zu Testern oder Teil der Crowd, sondern man könnte auch eine Community aufbauen. Dafür bräuchte es die Möglichkeit, dass sich ausgetauscht werden kann, auf die eine oder andere Art. Eine naheliegende Variante wäre es, dafür Github oder eine ähnliche Plattform zu verwenden, wo Issues von externen Personen erfasst und diskutiert werden können.
- Möglichkeiten für den Kontakt zur Community
 - Blog: Ein Blog könnte dafür verwendet werden, um über laufende Entwicklungen

und Veränderungen zu informieren. Hier könnte eine Kommentarfunktion auch einen Austausch in der Community ermöglichen.

- **Kontaktmöglichkeit:** Bei Problemen, Fragen oder Rückmeldungen soll es die Möglichkeit geben, direkt mit den Betreibern Kontakt aufzunehmen.
- **Feedback:** Die Bestätigungsseite des Testlabs könnte ausgebaut werden, um eine Rückmeldung vom Tester einzuholen. Entweder Freitext oder konkrete Fragen zum Reflektieren der Kriterien oder des Testablaufs.

7.3.5 Weitere Gedanken

- **Scoring:** Je nach Entwicklung der Plattform kann es sein, dass sich die aktuell ausgewählten Werte als unpraktisch erweisen, irritierend sind oder die Resultate nicht repräsentativ abbilden. Deshalb könnte eine qualitative Wertung ergänzt werden. Eine mögliche Abstufung wäre von »very bad« bis »very good«. Dies würde eine Dynamisch Auswertung erlauben. Eine Library mit wenigen Tests hat bei wenigen negativen Testresultaten schon eine grosse Einbusse im Score, bei Vielen fällt dies weniger ins Gewicht. Eine dynamische qualitative Auswertung würde es ermöglichen, auch die Anzahl Tests oder den Agreement Score in das Resultat miteinzubeziehen.
- **Alternative Verwendung der Applikation:**Die vorliegende Applikation könnte mit relativ wenig Aufwand durch Austauschen des Kriterienkataloges und des Scorings und Anpassung der Benutzeroberfläche für komplett andere Zwecke verwendet werden. Die Kernkompetenz ist es ja, dass etwas kollaborativ, anhand vorbestimmter Kriterien, mehrfach bewertet wird und davon eine Auswertung gemacht werden soll.

8 Schlusswort

Die entwickelte Webanwendung »Project Cactus« ist ein Proof-of-Concept, das das Testen aus Sicht der Accessibility von UI Libraries erlaubt und die Resultate allen Besuchern bereitstellt. Sie leistet hoffentlich einmal einen Beitrag dazu, dass das Web für alle mehr zugänglich ist, indem sie den Entwicklern die Suche nach einer geeigneten UI-Library vereinfacht und Ihnen gleichzeitig ermöglicht, in die Welt des Accessibility Testings hineinzuschnuppern. Falls die Plattform erfolgreich ist, könnte sie vielleicht auch unaccessible Libraries dazu umstimmen, mehr in Accessibility zu investieren.

Die Anwendung wurde mit Usability Tests, Systemtests und einer Accessibility Evaluation auf Herz und Nieren geprüft und hat dies erfolgreich überstanden. Es bestätigt die Erfüllung des geforderten und benötigten Funktionsumfangs und bestärkt die Qualitäten der Anwendung.

Es steht damit nichts im Weg, die Entwicklung fortzusetzen. Dafür wurde ein umfangreicher Katalog an Empfehlungen erstellt. Der Kriterienkatalog muss ausgebaut werden, um alle wichtigen Komponenten-Typen abzubilden. Die Anwendung muss Performance- und Sicherheitstechnisch auf Vordermann gebracht werden und deployed werden. Daneben gibt es diverse grössere und kleinere Funktionen, die für eine vollwertige Anwendung noch benötigt werden oder mindestens das Benutzererlebniss verbessern.

9 Anhang

Abbildungsverzeichnis

1	UI Libraries Diagramm	3
2	Homepage der entwickelten Webanwendung	4
3	Detailansicht einer UI-Library	5
3.1	Komponententypen nach Art gruppiert	21
3.2	Scoring-Modell	23
3.3	Wireframe der Homepage von Project Cactus	28
3.4	Wireframe der Übersichtsseite der Libraries	29
3.5	Detailansicht einer Library	30
3.6	Testlab Auswahl Testmodus	31
3.7	Testlab Instruktion	31
3.8	Testlab Testformular	31
3.9	Testlab Bestätigung	31
4.1	Project Cactus Logo	32
4.2	Architektur der Webapplikation	33
4.3	Routing in React	34
4.4	Die Datiestruktur im Frontend	34
4.5	Architektur Backend	36
4.6	Datenmodell der MongoDB Datenbank	37

Tabellenverzeichnis

2.1	Gruppierung von Behinderungen	14
-----	---	----

Literaturverzeichnis

- [1] A11y project checklist. <https://www.a11yproject.com/checklist/>. Accessed: 2022-01-13.
- [2] Accessibility support. <https://a11ysupport.io/>. Accessed: 2022-01-13.
- [3] Aria authoring practices guide (apg). <https://www.w3.org/WAI/ARIA/apg/>. Accessed: 2022-01-13.
- [4] Atkinson hyperlegible font. <https://brailleinstitute.org/freefont>. Accessed: 2022-01-13.
- [5] ech-0059 accessibility standard v3.0. <https://ech.ch/de/ech/ech-0059/3.0>. Accessed: 2022-01-13.
- [6] W3C WAI web accessibility laws and policies. <https://www.w3.org/WAI/policies/>. Accessed: 2022-01-13.
- [7] Web content accessibility guidelines. <https://www.w3.org/WAI/standards-guidelines/wcag/>. Accessed: 2022-01-13.
- [8] The WebAIM million accessibility report. <https://webaim.org/projects/million/#conclusion>. Accessed: 2022-01-13.
- [9] *Gleichstellung von Menschen mit Behinderungen*. Number 15003394. Bundesamt für Statistik (BFS), Neuchâtel, Dec 2020.
- [10] Craig Abbott. Axe-core vs pa11y. <https://github.com/abbott567/axe-core-vs-pa11y>. Accessed: 2022-01-13.
- [11] Hayfa Y Abuaddous, Mohd Zalisham Jali, and Nurlida Basir. Web accessibility challenges. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2016.
- [12] Darek Kay. The state of accessible web ui frameworks. <https://darekkay.com/blog/accessible-ui-frameworks/>. Accessed: 2022-01-13.
- [13] Yury Puzis, Yevgen Borodin, Andrii Soviak, Valentyn Melnyk, and IV Ramakrishnan. Affordable web accessibility: A case for cheaper aria. In *Proceedings of the 12th International Web for All Conference*, pages 1–4, 2015.
- [14] Greg Whitworth. Initial thoughts on standardizing form controls. <https://www.gwhitworth.com/posts/2019/form-controls-components/>. Accessed: 2022-01-13.

9.1 Systemtests

Auf den nachfolgenden Seiten sind die folgenden Dokumente zu finden:

- Systemtest Spezifikation 1.0
- Systemtest Protokolle nach Spezifikation 1.0
- Systemtest Spezifikation 1.1
- Systemtest Protokolle nach Spezifikation 1.1

Systemtest Spezifikation

Version 1.0

Zweck

Die Systemtests dienen dem Test des gesamten Systems. Sie werden anhand der User Stories konstruiert und sollen deren Umsetzung testen.

Testfälle

ID	Bezeichnung	Erwartetes Resultat
1	Übersicht über Libraries mit Testresultaten	Die Übersicht und Detailansicht ist vollständig und nützlich.
2	Login für Testlab und Registration	Es gibt eine Loginfunktion.
3	Library hinzufügen	Tester können Libraries hinzufügen.
4	Library Testing	Tester können Komponenten testen.
5	Versionsunterscheidung	Libraries können verschiedene Versionen haben.
6	Mehrfachtesting	Komponententests können mehrfach durchgeführt werden.

Systemtest-Protokoll

Dieses Dokument dient zur Protokollierung der Systemtests anhand der Systemtest-Spezifikation.

Angaben zur Durchführung

Datum:	24.11.22
Tester:	Mathias Lenz
Spezifikations-Version:	1.0

Test-Protokoll

ID	Implementiert	Bemerkungen	Status
1	Teilweise	Das Scoring funktioniert nur teilweise. Die Links in der Detailansicht nutzen zu viel Platz.	40%
2	Teilweise	Der Loginstatus wird nicht angezeigt. Die Loginfunktion an sich ist nur ein Platzhalter	10%
3	Ja	Funktioniert, gibt aber keine Rückmeldung und man wird nicht weitergeleitet.	30%
4	Ja	Testing funktioniert grundsätzlich. Es gibt aber keinen Zurück-Button oder eine Möglichkeit, abzubrechen. Die Instruktionen und Hilfen sind nicht besonders hilfreich bis anhin. Es hat keine Überschrift oder Navigation.	60%
5	Nein		0%
6	Teilweise	Es wird aktuell nicht verhindert, und ist daher möglich. Das aktuelle Scoring-System	10%

Empfehlungen

Ich empfehle, das Scoring-System als erstes in Angriff zu nehmen, damit dies überall funktioniert. Dann muss schrittweise die UX verbessert und Funktionen ergänzt werden.

Systemtest-Protokoll

Dieses Dokument dient zur Protokollierung der Systemtests anhand der Systemtest-Spezifikation.

Angaben zur Durchführung

Datum:	19.12.22
Tester:	Mathias Lenz
Spezifikations-Version:	1.0

Test-Protokoll

ID	Implementiert	Bemerkungen	Status
1	Ja	Die Übersicht ist vorhanden und funktioniert. Es fehlen allerdings noch verschiedene Angaben, wie zum Beispiel die Version(en) oder eine Angabe zur Qualität der Tests. Ebenfalls fehlen Angaben, wenn keine Tests existieren. Ein Filter wäre unter Umständen auch noch eine Implementation wert.	80%
2	Ja	Der Login und die Registration funktionieren. Jedoch ist die Benutzerführung nicht optimal. Der Nutzer wird ohne Rückmeldung eingeloggt und registriert.	80%
3	Ja	Das Hinzufügen von Libraries funktioniert ebenfalls. Hier fehlt ebenfalls noch Benutzerführung.	90%
4	Ja	Die Libraries können getestet werden. Eine Nummerierung der Steps wäre hilfreich. Der Text bei Specify braucht mehr Stuktur. Die Hilfetexte sollten deskriptiver sein. Der Help Button sollte besser beschriftet sein. Das Commentfeld sollte required sein.	90%
5	Teilweise	Ja, sie können mehrere Versionen haben, es ist allerdings nicht möglich, neue Versionen hinzuzufügen. oder sie zu wechseln	50%
6	Ja	Komponenten können mehrfach getestet werden. Diese werden gezählt und dargestellt.	100%

Empfehlungen

Generell braucht es mehr Rückmeldung an den Nutzer. Dies sollte nächster Fokus der Umsetzung sein.

Systemtest Spezifikation

Version 1.1

Zweck

Die Systemtests dienen dem Test des gesamten Systems. Sie werden anhand der User Stories konstruiert und sollen deren Umsetzung testen. In der Spalte US sind die zugehörigen User Stories referenziert.

Testfälle

ID	US	Bezeichnung	Erwartetes Resultat
1	1	Übersicht	Navigation durch die Webseite möglich und Zweck erkennbar.
2	2	Übersicht über Libraries mit Testresultaten	Die Übersicht und Detailansicht ist vollständig und nützlich.
3	3	Login und Registration	Registration und An-/Abmeldung möglich. Anmeldestand sichtbar.
4	5	Library hinzufügen	Tester können Libraries hinzufügen.
5	4	Library Testing	Tester können Komponenten testen. Instruktionen sind vorhanden und hilfreich. Details gemäss US 4.
6	6	Versionsunterscheidung	Libraries können mehrere Versionen haben. Es können weitere hinzugefügt werden.
7	4	Mehrfachtesting	Komponententests können mehrfach durchgeführt werden. Resultate werden entsprechend dargestellt und aktualisiert.

Systemtest-Protokoll

Dieses Dokument dient zur Protokollierung der Systemtests anhand der Systemtest-Spezifikation.

Angaben zur Durchführung

Datum:	01.01.23
Tester:	Mathias Lenz
Spezifikations-Version:	1.1

Test-Protokoll

ID	Implementiert	Bemerkungen	Status
1	Ja	Grundsätzlich erfüllt. About Seite noch recht leer. Könnte von visueller Unterstützung profitieren.	90%
2	Ja	Erfüllt. Hinweise zu Scores um Kriterienauswertung erweitern. Icons verwenden.	95%
3	Ja	Erfüllt. Fehlende Errormessage bei bereits existierendem Nutzer.	95%
4	Ja	Erfüllt. Leere Library wird korrekt dargestellt	100%
5	Ja	Teilweise erfüllt. Instruktionen fehlen teilweise noch.	90%
6	Ja	Verschiedene Versionen können erstellt und jeweils getestet werden. Neue Version kann nicht mit Enter bestätigt werden. Automatischer Wechsel der Version funktioniert nicht richtig.	90%
7	Ja	Erfüllt	100%

Empfehlungen

Behebung der oben beschriebenen kleineren Probleme.

Systemtest-Protokoll

Dieses Dokument dient zur Protokollierung der Systemtests anhand der Systemtest-Spezifikation.

Angaben zur Durchführung

Datum:	10.01.23
Tester:	Mathias Lenz
Spezifikations-Version:	1.1

Test-Protokoll

ID	Implementiert	Bemerkungen	Status
1	Ja	Erfüllt	100%
2	Ja	Erfüllt	100%
3	Ja	Erfüllt	100%
4	Ja	Erfüllt	100%
5	Ja	Erfüllt	100%
6	Ja	Erfüllt	100%
7	Ja	Erfüllt	100%

Empfehlungen

Alles erfüllt.

9.2 Accessibility Evaluation

Auf den nachfolgenden Seiten sind die folgenden Dokumente zu finden:

- Accessibility Evaluation Konzept
- Accessibility Evaluation Protokoll

Accessibility Evaluation

Die Webapplikation wird nach folgenden Kriterien evaluiert:

- Keyboardbedienbarkeit: Die zentralen Funktionen und alle Seitenbereiche lassen sich mit reiner Keyboardbedienung erreichen und bedienen.
- Screenreaderbedienbarkeit: Alle Seitenbereiche werden vom Screenreader sinnvoll durchgearbeitet und die notwendigen Funktionen sind verständlich zugänglich. Dies wird nach bestem Wissen und Gewissen des Testers beurteilt.
- Deque Axe Core Dev Tools Plugin: Das Axe Core Plugin wird aktiviert und verwendet, um Schwachstellen in der Seite zu finden.
- WCAG: Für eine annähernde Evaluation gemäss WCAG Kriterien wird die Checkliste des A11yprojects verwendet, die auf den Kriterien vom WCAG Level A und AA basiert: <https://www.a11yproject.com/checklist/>

Vorgehen:

Die auftretenden Probleme werden dokumentiert und vor der Weiterführung der Evaluation mit dem nächsten Abschnitt behoben.

- Deque Axe Core Dev Tools Plugin: Das Plugin wird auf den wesentlichen Seiten der Webseite verwendet, um potentielle Probleme zu finden.
- Keyboard- und Screenreaderbedienbarkeit :
 - Navigation durch die Webseite mittels der Kriterien des Systemtests, mit Screenreader und reiner Tastaturbedienung.
- WCAG-Konformität:
 - Durchgehen der Checkliste anhand der wesentlichen Seiten.

Anmerkung: Die wesentlichen Seiten sind: Homepage, Libraries Übersicht, Libraries Detail, gesamtes Testlab, Add Library, Contribute und Login.

Accessibility Evaluation Protokoll

Datum:	03.01.2023
Tester:	Mathias Lenz

Die auftretenden Probleme werden dokumentiert und vor der Weiterführung der Evaluation mit dem nächsten Abschnitt behoben.

Deque Axe Core Dev Tools Plugin

Das Plugin wird auf den wesentlichen Seiten der Webseite verwendet, um potentielle Probleme zu finden.

Bereich	Aufgetretene Probleme
Homepage	.button-primary hat nicht genügend Kontrast
Libraries Übersicht	-
Libraries Detail	Mehr als ein Main-Element vorhanden Add-Version Input hat das Label nicht korrekt zugewiesen.
gesamtes Testlab	ID von input-Element hat Bug und wird nicht korrekt generiert, ist deshalb nicht unique
Add Library	-
Contribute und Login	-

Anmerkungen:

Die Probleme wurden anschliessend an den Test behoben.

Keyboard- und Screenreaderbedienbarkeit

Navigation durch die Webseite mittels der Kriterien des Systemtests, mit Screenreader (NVDA) und reiner Tastaturbedienung.

Bezeichnung	Erwartetes Resultat	Aufgetretene Probleme
Übersicht	Navigation durch die Webseite möglich und Zweck erkennbar.	Kein Skiplink vorhanden- Navigation mühsam

Bezeichnung	Erwartetes Resultat	Aufgetretene Probleme
Übersicht über Libraries mit Testresultaten	Die Übersicht und Detailansicht ist vollständig und nützlich.	Show more buttons sind unspezifisch - sollten mehr infos bieten für screenreader, Result-Bubble wird mühsam vorgelesen
Login und Registration	Registration und An-/Abmeldung möglich. Anmeldestand sichtbar.	Login und Registrations-Feedback wird nicht vorgelesen
Library hinzufügen	Tester können Libraries hinzufügen.	Kein Screenreaderfeedback bei Erstellung
Library Testing	Tester können Komponenten testen. Instruktionen vorhanden.	Kein Screenreaderfeedback bei Seitenwechsel oder
Versionsunterscheidung	Libraries können verschiedene Versionen haben.	Kein Screenreaderfeedback bei Erstellung
Mehrfachtesting	Komponententests können mehrfach durchgeführt werden. Resultate werden entsprechend dargestellt	-

Anmerkungen:

Alerts werden mühsam vorgelesen -> Code angepasst, nun wird die ganze Message inklusive Typ vorgelesen.

Result-Bubble -> Angepasst, wird nun in einem Zug vorgelesen.

Fehlend ist vor allem Rückmeldung bei Problemen. -> aria-live regions wo möglich ergänzt

Kein Skiplink -> Skiplink erstellt

Accessible Navigation:

<https://a11y-guidelines.orange.com/en/articles/single-page-app/>

<https://www.gatsbyjs.com/blog/2019-07-11-user-testing-accessible-client-routing/>

Unspezifische Buttonbezeichnungen -> Buttons mit aria-label und dynamischen Werten ergänzt.

Seitenwechsel -> Autofocus und Pagetitle-Management auf neues Heading-Element, eigene Komponente dafür implementiert

WCAG-Konformität

Durchgehen der Checkliste anhand der wesentlichen Seiten: <https://www.a11yproject.com/checklist/>

Bereich	Aufgetretene Probleme
Homepage	-
Libraries Übersicht	-
Libraries Detail	-
gesamtes Testlab	-
Add Library	-
Contribute und Login	Labels sind nicht nahe an den Inputfeldern -> geändert

Anmerkungen:

Die Kriterien sind zum Teil recht subjektiv zu beurteilen. Es wurde nach bestem Wissen und mit den verfügbaren verlinkten Informationen geurteilt. Da viele Probleme mit den vorhergehenden Tests bereits behoben wurde, sind die Resultate hier sehr gut.

9.3 Code Reviews

Auf den nachfolgenden Seiten ist das Dokument zu den Code Reviews zu finden.

Code Reviews

Folgendes wurde nach dem Code Review 1 vom 24.11.22 behandelt:

- Server
 - Index-Seite stellt etwas dar
 - Umsetzung ESM und Typescript ist im Moment keine Priorität
- Typescript
 - Unused Code entfernt
 - Überflüssige console-logs entfernt
 - Generell Template-Strings eingesetzt statt String Concat
 - Object destructuring wurde umgesetzt
- Frontend/React
 - Context-API für Token verwendet
 - Page und Components aufgetrennt und besser geordnet
 - Prop-Drilling möglichst eingedämmt
 - unique key Prop Fehler wurde behoben
- CSS
 - CSS Files aufgetrennt
 - Eine Umstellung auf SCSS hat im Moment keine Priorität
 - Neue Komponente für Alerts erstellt und für verschiedene Meldungen eingesetzt
- HTML
 - Falsch verwendete Tags entfernt
 - Layout neu strukturiert
- Fragen
 - Extra kein Mono-Repo? Ja, Deployment ist keine Priorität. Getrennte Repos machen das Handling für mich einfacher.
 - testRoutes: unglückliche Bezeichnung -> wurde auf testingRoutes geändert.

Folgendes wurde nach dem Code Review 2 vom 16.12.22 behandelt:

- Allgemein
 - Kein README/Default-README -> Es wurden nun READMEs mit kurzen Anleitungen wo nötig erstellt.
- Server
 - Der Server setzt jetzt ESM und Typescript ein.
 - Die Dokumentation der API wurde mit Swagger erstellt.
 - server.js in app.js und index.js aufgeteilt
 - Ein Logging Framework hat aktuell keine Priorität, aber es wurden wo sinnvoll console.logs eingefügt um Aktionen auf dem Server nachvollziehen zu können.
 - Res.send wurde durch res.json abgelöst
 - Eine Auslagerung der Logik in den Controllern war leider nicht mehr umzusetzen
- Client
 - Client funktioniert nun auch mit leerem Startpunkt.

- UI für Login und Registration wurde geändert.
- `useState<any>()` wurde entfernt
- Eine ProtectedRoute Component wurde erstellt
- Die gesamte Dateistruktur wurde für eine bessere Übersicht umgestellt.
- Alle Programm-Files wurden von PascalCase in kebab-case umgestellt, um Namenskonflikten im Zusammenahng mit git vorzubeugen. Dies aufgrund des Inputs vor dem ersten Codereview.
- Testing
 - Es wurden Unit Tests für scoring und scoringHelpers erstellt
- HTML
 - Sections setzen die Heading für Screenreader nicht korrekt. Es bleibt alles bei H1 -> mit NVDA getestet.
- Code Qualität
 - Die Code-Qualität wurde wo möglich verbessert
 - Es gibt nun einen Fetch Interceptor/HTTPService

9.4 Code Reports

Auf den nachfolgenden Seiten ist der Code-Report zu finden.

Code Statistiken

Frontend - Lines of Code

59 text files.
59 unique files.
4 files ignored.

github.com/AlDania1/cloc v 1.96 T=0.51 s (115.0 files/s, 6829.5 lines/s)

Language	files	blank	comment	code
TypeScript	40	212	33	2398
CSS	18	113	19	635
Text	1	18	0	75
SUM:	59	343	52	3108

Backend - Lines of Code

18 text files.
18 unique files.
0 files ignored.

github.com/AlDania1/cloc v 1.96 T=0.09 s (191.6 files/s, 12411.5 lines/s)

Language	files	blank	comment	code
TypeScript	18	113	23	1030
SUM:	18	113	23	1030

9.5 UI Library und Komponenten Tests

Auf den nachfolgenden Seiten sind die Resultate der Tests mit der Applikation zu finden.

- Material UI
- Radix UI
- Semantic UI



Dialog (or Modal, Prompt)

A dialog is a window overlaid over rest of the page. Users cannot interact with content outside of the dialog.

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score 1 Tests 1 Agreement Score

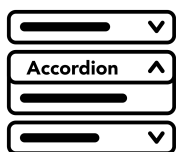
Keyboard Criteria Evaluation:

- The focus (style) of interactive elements is visible 1 | 1 | 0 | 0 | 0 | 1 Agreement Score
- Buttons are focusable 1 | 1 | 0 | 0 | 0 | 1 Agreement Score
- The focus does not leave the dialog 1 | 1 | 0 | 0 | 0 | 1 Agreement Score
- The dialog closes with the `esc` key 1 | 1 | 0 | 0 | 0 | 1 Agreement Score
- The focus returns to the calling button after closing or canceling the dialog with `esc` 1 | 1 | 0 | 0 | 0 | 1 Agreement Score

Screenreader Scores: 100% Accessibility Score 1 Tests 1 Agreement Score

Screenreader Criteria Evaluation:

- The [x] button of the dialog is announced as the close button (or similar). 1 | 1 | 0 | 0 | 0 | 1 Agreement Score
- The content of the dialog is read upon opening 1 | 1 | 0 | 0 | 0 | 1 Agreement Score



Accordion (or Disclosure)

An accordion is a vertically stacked set of headings each with a collapsible panel

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The header elements are focusable

1 | 0 | 0

1 Agreement Score

The focus (style) of the header elements is visible

1 | 0 | 0

1 Agreement Score

The panel opens/closes with `space` or `enter` keys upon focus of a header element

1 | 0 | 0

1 Agreement Score

Navigation between panel headers is possible with `tab`

1 | 0 | 0

1 Agreement Score

Screenreader Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Screenreader Criteria Evaluation:

Any hidden text is not read

1 | 0 | 0

1 Agreement Score

Tooltip



Tooltip (or Inline Dialog, Popover)

A tooltip is a small popup element that shows additional information when hovering or focusing the tooltip triggering element.

Overall Scores: 100% Accessibility Score

2 Tests total

1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The element with the tooltip is focusable

1 | 0 | 0

1 Agreement Score

The focus (style) of the triggering element is visible

1 | 0 | 0

1 Agreement Score

The tooltip is shown upon focus

1 | 0 | 0

1 Agreement Score

The tooltip closes with the `esc` key

1 | 0 | 0

1 Agreement Score

Screenreader Scores: 100% Accessibility Score

1 Tests

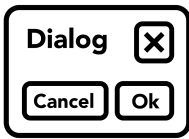
1 Agreement Score

Screenreader Criteria Evaluation:

The tooltip is read upon focus of the element

1 | 0 | 0

1 Agreement Score



Dialog (or Modal, Prompt)

A dialog is a window overlaid over rest of the page. Users cannot interact with content outside of the dialog.

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score 1 Tests 1 Agreement Score

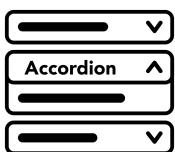
Keyboard Criteria Evaluation:

The focus (style) of interactive elements is visible	1 0 0	1 Agreement Score
Buttons are focusable	1 0 0	1 Agreement Score
The focus does not leave the dialog	1 0 0	1 Agreement Score
The dialog closes with the `esc` key	1 0 0	1 Agreement Score
The focus returns to the calling button after closing or canceling the dialog with `esc`	1 0 0	1 Agreement Score

Screenreader Scores: 100% Accessibility Score 1 Tests 1 Agreement Score

Screenreader Criteria Evaluation:

The [x] button of the dialog is announced as the close button (or similar).	1 0 0	1 Agreement Score
The content of the dialog is read upon opening	1 0 0	1 Agreement Score



Accordion (or Disclosure)

An accordion is a vertically stacked set of headings each with a collapsible panel

Overall Scores: 100% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The header elements are focusable

1 | 0 | 0

1 Agreement Score

The focus (style) of the header elements is visible

1 | 0 | 0

1 Agreement Score

The panel opens/closes with `space` or `enter` keys upon focus of a header element

1 | 0 | 0

1 Agreement Score

Navigation between panel headers is possible with `tab`

1 | 0 | 0

1 Agreement Score

Screenreader Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Screenreader Criteria Evaluation:

Any hidden text is not read

1 | 0 | 0

1 Agreement Score

Tooltip



Tooltip (or Inline Dialog, Popover)

A tooltip is a small popup element that shows additional information when hovering or focusing the tooltip triggering element.

Overall Scores: 100% Accessibility Score

2 Tests total

1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 100% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The element with the tooltip is focusable

1 | 0 | 0

1 Agreement Score

The focus (style) of the triggering element is visible

1 | 0 | 0

1 Agreement Score

The tooltip is shown upon focus

1 | 0 | 0

1 Agreement Score

The tooltip closes with the `esc` key

1 | 0 | 0

1 Agreement Score

Screenreader Scores: 100% Accessibility Score

1 Tests

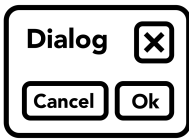
1 Agreement Score

Screenreader Criteria Evaluation:

The tooltip is read upon focus of the element

1 | 0 | 0

1 Agreement Score



Dialog (or Modal, Prompt)

A dialog is a window overlaid over rest of the page. Users cannot interact with content outside of the dialog.

Overall Scores: 12% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores: 25% Accessibility Score 1 Tests 1 Agreement Score

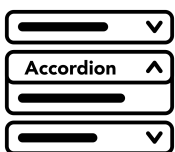
Keyboard Criteria Evaluation:

- The focus (style) of interactive elements is visible 0 thumbs up | 0 thumbs down | 1 question mark 1 Agreement Score
- Buttons are focusable 0 thumbs up | 1 thumbs down | 0 question mark 1 Agreement Score
- The focus does not leave the dialog 0 thumbs up | 1 thumbs down | 0 question mark 1 Agreement Score
- The dialog closes with the `esc` key 0 thumbs up | 1 thumbs down | 0 question mark 1 Agreement Score
- The focus returns to the calling button after closing or canceling the dialog with `esc` 1 thumbs up | 0 thumbs down | 0 question mark 1 Agreement Score

Screenreader Scores: 0% Accessibility Score 1 Tests 1 Agreement Score

Screenreader Criteria Evaluation:

- The [x] button of the dialog is announced as the close button (or similar). 0 thumbs up | 0 thumbs down | 1 question mark 1 Agreement Score
- The content of the dialog is read upon opening 0 thumbs up | 1 thumbs down | 0 question mark 1 Agreement Score



Accordion (or Disclosure)

An accordion is a vertically stacked set of headings each with a collapsible panel

Overall Scores: 50% Accessibility Score 2 Tests total 1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores:

0% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The header elements are focusable

0 | 1 | 0 | 0

1 Agreement Score

The focus (style) of the header elements is visible

0 | 1 | 0 | 0

1 Agreement Score

The panel opens/closes with `space` or `enter` keys upon focus of a header element

0 | 1 | 0 | 0

1 Agreement Score

Navigation between panel headers is possible with `tab`

0 | 1 | 0 | 0

1 Agreement Score

Screenreader Scores:

100% Accessibility Score

1 Tests

1 Agreement Score

Screenreader Criteria Evaluation:

Any hidden text is not read

1 | 0 | 0 | 0

1 Agreement Score

Tooltip**Tooltip (or Inline Dialog, Popover)**

A tooltip is a small popup element that shows additional information when hovering or focusing the tooltip triggering element.

Overall Scores:

0% Accessibility Score

2 Tests total

1 Agreement Score

^ Hide detailed evaluation

Keyboard Scores:

0% Accessibility Score

1 Tests

1 Agreement Score

Keyboard Criteria Evaluation:

The element with the tooltip is focusable

0 | 1 | 0 | 0

1 Agreement Score

The focus (style) of the triggering element is visible

0 | 1 | 0 | 0

1 Agreement Score

The tooltip is shown upon focus

0 | 1 | 0 | 0

1 Agreement Score

The tooltip closes with the `esc` key

0 | 1 | 0 | 0

1 Agreement Score

Screenreader Scores:

0% Accessibility Score

1 Tests

1 Agreement Score

Screenreader Criteria Evaluation:

The tooltip is read upon focus of the element

0 | 1 | 0 | 0

1 Agreement Score

9.6 Wireframes

Auf den nachfolgenden Seiten sind die Wireframes zu finden.



https://



Project Cactus

Login

Home

Libraries

Contribute

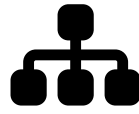
Welcome to Project Cactus



Collaborative



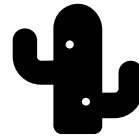
Accessibility



Components



Testing



Cactus

The Mission

Placeholder text for The Mission section.

Check out the Libraries

Contribute!

Placeholder text for Contribute! section.

Contribute





https://



Project Cactus

Login

Home

Libraries

Contribute

Libraries

Disclaimer ...

[Placeholder text for disclaimer]

Do you miss a UI library?

Add new Library

Example UI v1

100% Accessibility Score

7 Components tested

Show more

Semantic UI v2.5

83% Accessibility Score

2 Components tested

Show more

Material UI v2 (older versions available)

There are currently no tested components for this version

Show more






https://



Project Cactus

 Login

[Home](#)

[Libraries](#)

[Contribute](#)

New Library

Handwritten-style decorative text

Name of the Library

Current Version

Link to Homepage

Link to Documentation

Cancel and return

Add new Library





https://



Project Cactus

Login

Home

Libraries

Contribute

Example UI

83% Average Accessibility Score

[Documentation](#)

[Website](#)

Version: v5.3

Add Version

Add Component Test

Explanation to Scores

[Blurred text]

Die orangen Buttons werden nur eingeloggten Nutzern angezeigt.



Dialog (Modal)

component

A dialog is a window overlaid over rest of the page.

Overall Scores:

45% Accessibility Score

3 Tests total

0.67 Agreement Score

Show detailed evaluation



Tooltip (Inline Dialog, Popover)

component

[Blurred text]

Overall Scores:

45% Accessibility Score

3 Tests total

0.67 Agreement Score

Hide detailed evaluation

Keyboard Scores:

There are currently no tests for this testmode

Screenreader Scores:

45% Accessibility Score

3 Tests

0.67 Agreement Score

Screenreader Criteria Evaluation:

The `[x]` close button has alt-text, that is read with the screenreader

2 + 10 - 11?

0.67 Agreement Score

The content of the dialog is read upon opening

2 + 10 - 11?

0.67 Agreement Score

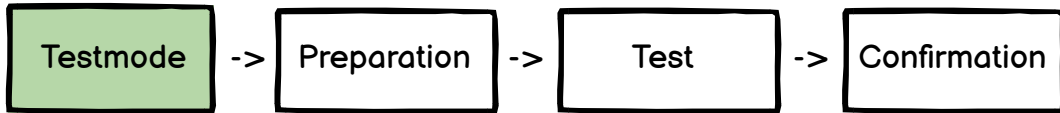


https://



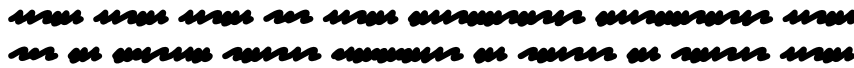
Cactus Testlab

Library: Material UI
Version: 5.3
Component: Dialog
Testmode: Keyboard Use



Welcome to the testlab.

Short Introduction and Instruction



Select one of the components to continue

Dialog (Modal)

Test Keyboard
(5 Tests)

Test Screenreader
(3 Tests)

Accordion

Test Keyboard
(5 Tests)

Test Screenreader
(3 Tests)

< Back

> Next



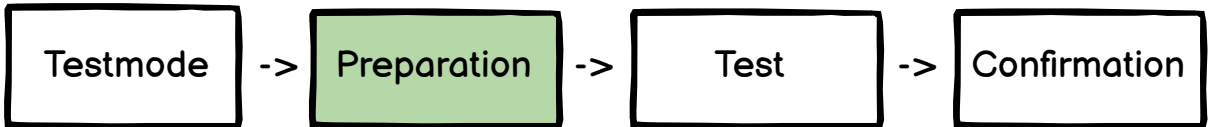


https://



Cactus Testlab

Library: Material UI
Version: 5.3
Component: Dialog
Testmode: Keyboard Use

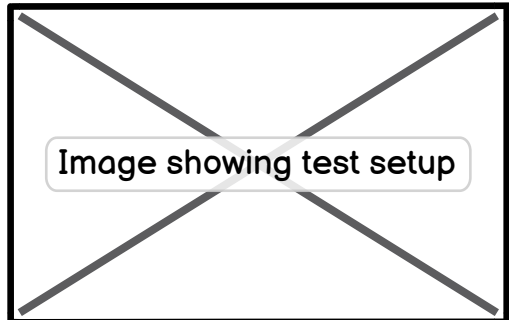


Test Preparation

Step 1: Open Documentation

Placeholder text for Step 1 instructions

Open Documentation



Step 2: Navigate to the Component

Placeholder text for Step 2 instructions

< Back

> Next



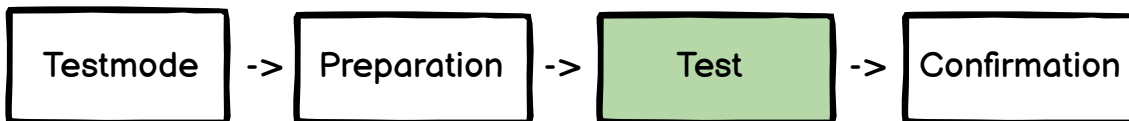


https://



Cactus Testlab

Library: Material UI
Version: 5.3
Component: Dialog
Testmode: Keyboard Use



How to Test

.....
.....
.....

Focus Style is visible

.....
.....

Yes No Not applicable

Focus is trapped inside the modal

.....
.....

Yes No Not applicable

Why is it not applicable?

.....



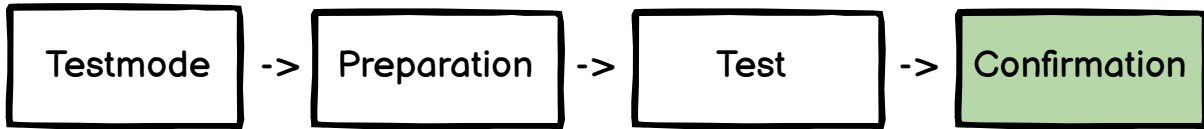


https://



Cactus Testlab

Library: Material UI
Version: 5.3
Component: Dialog
Testmode: Keyboard Use



Thank you very much!

Your test was added to the system.

[Go back to the Library](#)

