


# National College of Ireland

Computing BSHC4

Software Development

2020/2021

Matthew Byrne

X17138744

X17138744@student.ncirl.ie

# Aria-Able

# Technical Report

# Contents

# Executive Summary

Detailed inside this report are the findings, research and culmination of a project dedicated to building a software accessibility tool using web components-based architecture. This document has extensive research in official specifications to web accessibility, along with details to single page application (SPA) framework that expands on complexity. The main areas inside the report are the inner-workings of traditional web components and their features; a high-level analysis of a framework that uses SPA architectures, and everything finalised for the project entitled "*Aria-Able*".

Along with the detailed textual information that is contained in this document, there are visual plots, illustrations and diagrams that will provide readers clarity into the aims of the project and its intentions to deliver finalised software as a product. There are dedicated sections to the functional requirements of the application, insights into why this area of development was chosen as a background, along with emphasis on referencing material that was researched and created. The benefits for a viewer to the pages of this report is gaining a better insight into the areas of web accessibility, and component driven architecture in client-side applications. From this conducted research, and continued practical ability in creating applications, new technology discoveries are a constant in this written document, along with the benefits and drawbacks in the areas of website accessibility. The purpose to these findings is to gain higher knowledge while also implementing professional practices to software development over the given time span to complete the project.

The results in this report are now finalised and the sections have been updated from the previous submission of the project proposal template. This technical document is now the main accompaniment for the practical delivery of the project.

# 1.0 Introduction

## 1.1. Background

Over the past year I have given research into the field of pure frontend development and client-side applications to find that this area of the industry is creative. The aim is to move forward and proceed with this well-established area of the industry for my career. Because of this and the time given to research, I intended to undertake a project that represents what I am passionate about with software development; along with showcasing my abilities at the highest level.

Web accessibility and how people access the internet is not just of moral values but of ethical values also. With the creation of the internet infrastructure, it was fundamentally designed to be interacted with for all walks of life, from all around the globe irrespective of advantages or disadvantages. Accessibility and web accessibility are of utmost importance as it was originally when the first website arrived with full accessibility features in 1991 by Tim Berners Lee the father of the internet [18]. The areas of accessibility are connected to many areas with cognitive, auditory, neurological, physical, and visual elements [19].

With the nineties and later decades, there have been specifications, standards, practices, and systems dedicated to increasing the level of accessibility in web pages.

Over time and due to the increased complexity of information stored in websites and with other factors such as browser-based optimisations, there has been a continued drop in web pages not containing full accessible content. With this there have been legal ramifications with website accessibility for companies and investors over the years that provide an important background for the project.

In the year of 2018 website lawsuits tripled to 2,258 cases, over the 814 that were documented in the United States in 2017 with accessibility issues on websites [20]. Moving more closer to home, the EU Web Accessibility Directive of 2016 took action to tackle a continued downturn of websites with a law that states all government sites created after September 2018 will have to be fully accessible, along with mobile applications at a later period [21]. With these facts discovered, the wheels were set in motion to find could there be potential in making these areas better when it comes to web accessibility.

My background for this project was to give the research to build an accessibility tool that offers value for non-public bodies and websites currently being developed. I chose this area to contribute and create an open-source piece of functionality that can be beneficial to not just a user, but to also developers who code websites who may leave accessibility as an afterthought in the software development lifecycle. With the project being of ethical and moral principles, along with an area that I continue to have common interest in more and more, I felt that the choice to proceed with a project that highlights a huge area was the best natural approach. This project intends not only to be beneficial to myself in its creativity and development, but to anyone who interacts in the open-source community.

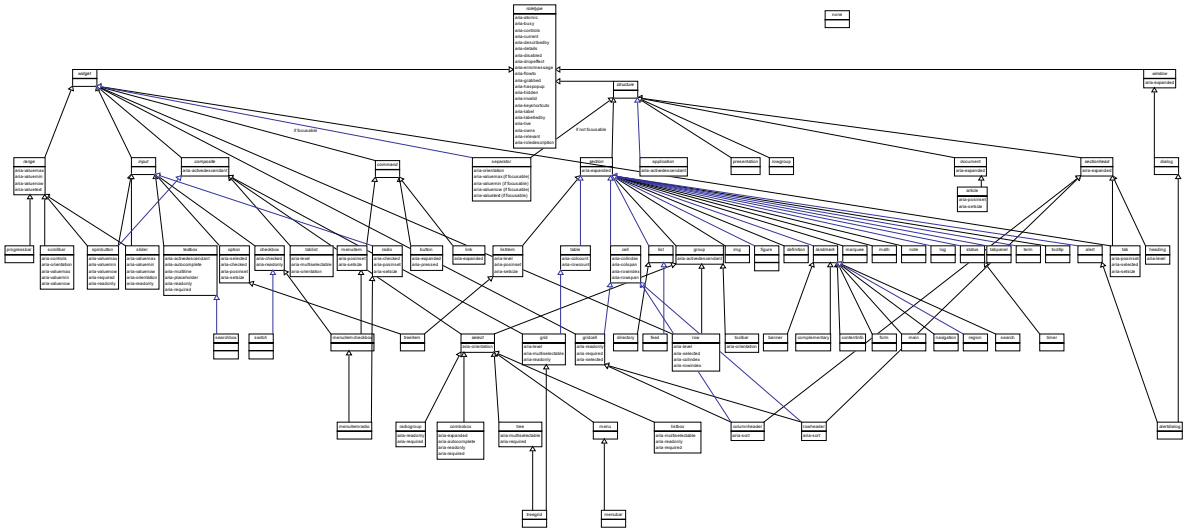
## 1.2. Aims

"Aria-Able" is a design based library "package" [1] that lets users attain accessibility elements to websites for an improved user experience. How this functionality is achieved is by taking advantage of specifications and semantics that are already available through HTML5 and open source. To activate accessibility standards in websites, the World Wide Web Consortium (W3C) [2] has designed a HTML standard (currently version 5) to be semantic; an element on screen can visually describe meaning to both the browser and the document object model based in the browser.

To accompany these HTML5 style tags that are being iteratively updated by the W3C, there was another specification created that compliments this area. This is called Accessible Rich Internet Applications or ARIA for short [3]. Like HTML5 and their pre-defined semantic tags, ARIA is another set of attributes that compliments HTML5 bringing assistive technology to a browser window. This can take many forms for example, screen readers for input fields, check boxes, sliders and more. How ARIA specification is managed is by breaking up areas of interaction on priority to the interacting user into sections. The main activating semantic elements are called "roles" [2]. With these roles for ARIA to be activated, there is a process of adding an attribute to a HTML5 tag. The aim of project is to encapsulate and abstract this ARIA logic away to where you can simply add the relevant custom component tag forming a library of created accessibility tags.

This ability is achieved by creating custom Web Components [4]. The main implementation or driving factors of these components are the current popularity in JavaScript and component-based logic for more modular over monolithic application building. Along with these component creations, which are technically detailed in this document, the continued complexity of the projects aims is to then introduce these components into a library of JavaScript entitled React and Storybook JS [5]. React renders components and custom components with code to a user and is a client-side based framework developed by Facebook. Storybook is a design system that works by the creation of "stories" as elements in a library. A bigger example of this would be Google's Material Design system.

By utilising the constructs of a framework such as React and Storybook JS; the aim is to create a modular component system where there is never any need to declare additional ARIA semantics and achieve encapsulation benefits.



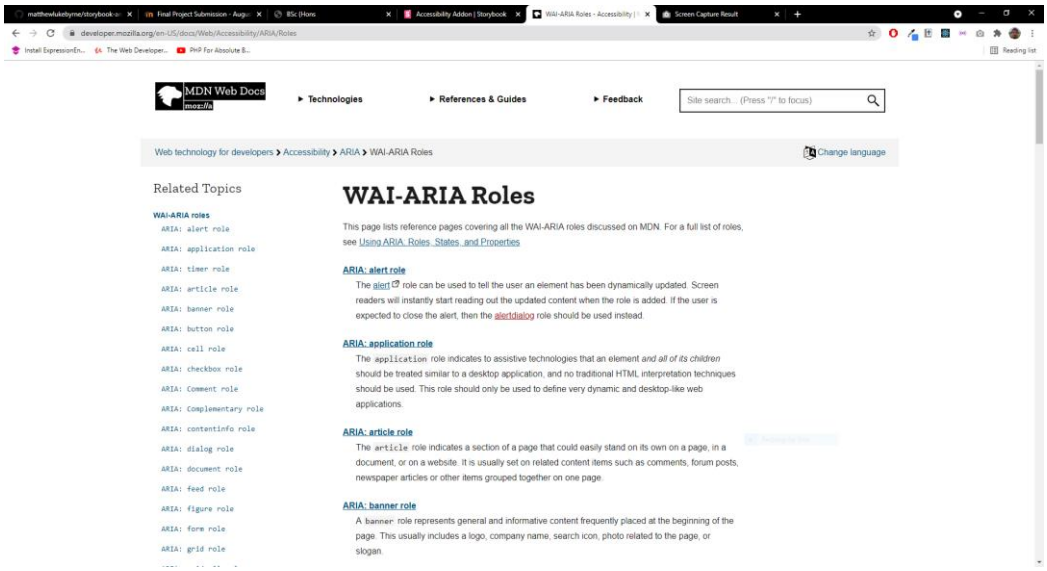



Fig.1. Full WAI-ARIA Accessibility Roles Mappings (W3C, WGAC)

## 1.3. Technology

The primary codebase language, JavaScript was developed across two repositories; one that facilitates the React client-side application and the other as a component building area that utilised Storybook. There was the possibility to increase the complexity of the codebase language choice with TypeScript when developing accessible web components, as well as a multitude of tools that utilise the language encapsulated into open-source modules [6]. In addition to this there is HTML, SCSS, Docker scripts, MDX files and more in the main repositories also. In the area to the tools and processes that make up client-side development (bundler tools, linting, compilers) there was a huge array of tools implemented with the likes of Parcel, Webpack and Babel JS that have all created a great production-based feel Parceljs.org. (2021).

Web components utilise the Document Object Model (DOM) [14] which is a native object that manipulates information on a web browser window. This practice is synonymous and well known with a client-side based language like JavaScript; so, it was logical to proceed with this language heavily in the areas for development. The fundamentals of components was explored, before moving onto compilers and frameworks that can produce components faster such as Ionics's Stencil for advanced complexity [7]. Stencil can extend the ability to create web components by having syntax and structure that shares similarity to JavaScript style framework code. Plain web components share similarities to Java syntactically to which functionality is contained inside a class.

These classes have methods which are being called on when loaded in the browser window. Once these methods are called, there is a syntax to create and instantiate your own custom HTML tag which then becomes your component call. With this technology the ARIA attributes inside JavaScript code the codebase was encapsulated and running the associated logic as HTML tags in the browser. This was the core functionality of the project and its component creating strategy. Components required several steps to ensure they are running correctly with a bigger scale of needed logic; this is where the stages of complexity increased over time by adding additional technologies. Unit and Snapshot testing was conducted with a tool called Jest a native framework to the React library and Enzyme which is a behaviour driven testing tool. These testing areas are very popular industry standards in relation to JavaScript development, and research into these tools is further explained in the testing section.

With the biggest stage of the project, React was a framework that required a learning curve to understand its abilities and how it converts a traditional loading website into a single page application. Along with this, there were other tools thrown into the frame such as React Router that allows declarative routing across a multitude of pages. React is a library itself that allows JavaScript to be called over component calls and this is core area of the library. Because components are broken up on a web page and their logic is encapsulated this allows for faster loading of dynamic websites as it only renders certain sections to a browser window. It does not need the several attempts to fetch information and do a refresh from a server. The client-side aspects of the project are where my completed components were inserted into the React framework. These are only just the primary technologies that were used in the stages of the project and in the repositories on GitHub there is documentation listing a multitude of tools. The list of technologies and tools are listed in full in the appendix of this report.

### 1.4. Structure

Leading on from the revision of the proposal and after addressing introductory sections of this report, these sections ahead contain the requirements & engineering aspects of the project. There is also an emphasis to highlight the benefits of the different types of individuals aimed at utilising this idea of software development in the future. Throughout the requirements sections the intention was to create a systematic approach to gathering requirements in two areas; the user who can interact with the product called "Aria-Able"; and the developer who will be able to install and implement these features for their own future applications. The next section highlights on system compatibility and requirements for future support of the Aria-Able platform.

## 2.0 System

### 2.1. System Requirements

In the areas regarding the system compatibility, there are now few issues when developing custom web components specifications in JavaScript. Full support across all major browsers that provide support is now implemented, along with OS compatibility for Windows, Mac and Linux. Web components are rendered with Vanilla JavaScript (which is just simply JavaScript code without the use of a platform or framework), is supported by all browsers including the use of React and Storybook. There used to be a period where due to browser incompatibility there was need for such an item called "polyfills". Now thanks to increased adoption rates to Chromium browsers across platforms, this concern is of less priority now. Detailed below is the specification standard for web components and its current percentages across specifications. This information was provided by "*Can I Use*" which keeps a detailed description of past and future browser support for DOM APIs and the HTML elements.

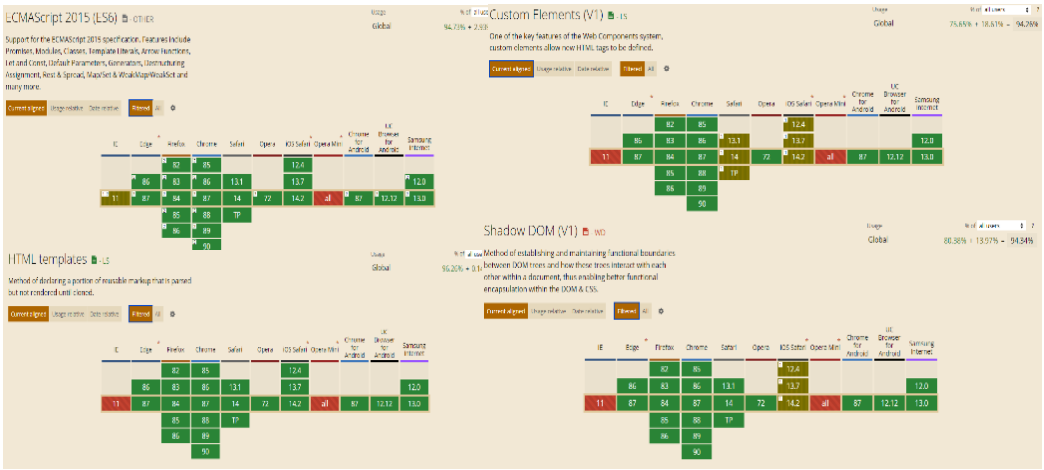



Fig.2 System Requirements Browser Overview generated with Can I Use.

### 2.1.1. Functional Requirements

Below is the list of original functional requirements in the application from the Mid-Point Presentation, unchanged to see how the project has developed and delivered on its statements for component functionality and the software development lifecycle. The list is ordered, the highest priority at the top ranging to the lowest priority at the bottom. Comments are made to highlight what areas were affected and unaffected in the process. Due to the lack of encapsulating logic into a NPM style package as a requirement there is a fair amount of crossover with requirements. NPM potential was halted due to the scope and range with the project and also with the logical capabilities of the accessible component library.

| Requirement Statement | Must / Want | Comments |
|---|---|---|
| The **user** can access accessibility functionality that is coming from the packaged logic. | Must | This was delivered and deployed via Storybook JS. |
| The **user** can access accessibility functionality when an assistive technology is enabled in the browser. | Must | This was delivered and deployed via Storybook JS. |
| The **developer** can install the functionality with a preferred terminal command. | Must | This was delivered with Storybook via Storybook JS |
| The **user** can become the developer and utilise this tool when building their own websites. | Must | This was not delivered on yet. |
| The **developer** can install the packaged logic on any platform, and it can still run in any browser | Must | This was not delivered only with Storybook JS deployed. |
| The **user** can access additional information on the project with the React document site. | Must | This was delivered with the React application. |
| The **user** can test the created custom logic on the React site in the “Documentation” section. | Want | This was not delivered on yet. |
| The **developer** can install the application on NPM and be able to access the features in a non-JavaScript codebase. | Want | This was not delivered on yet. |
| The **user** can access and take advantage of the accessibility features and logic on mobile platforms. | Want | This was not delivered on yet. |
| The **user** in the future can sign up through email for a newsletter that informs people of recent updates and additions the software. | Want | Database installed via Google Firebase. |

Fig.3 Final Updates Functional Requirements (July 2021)

### 2.1.1.1. System Use Case Diagram

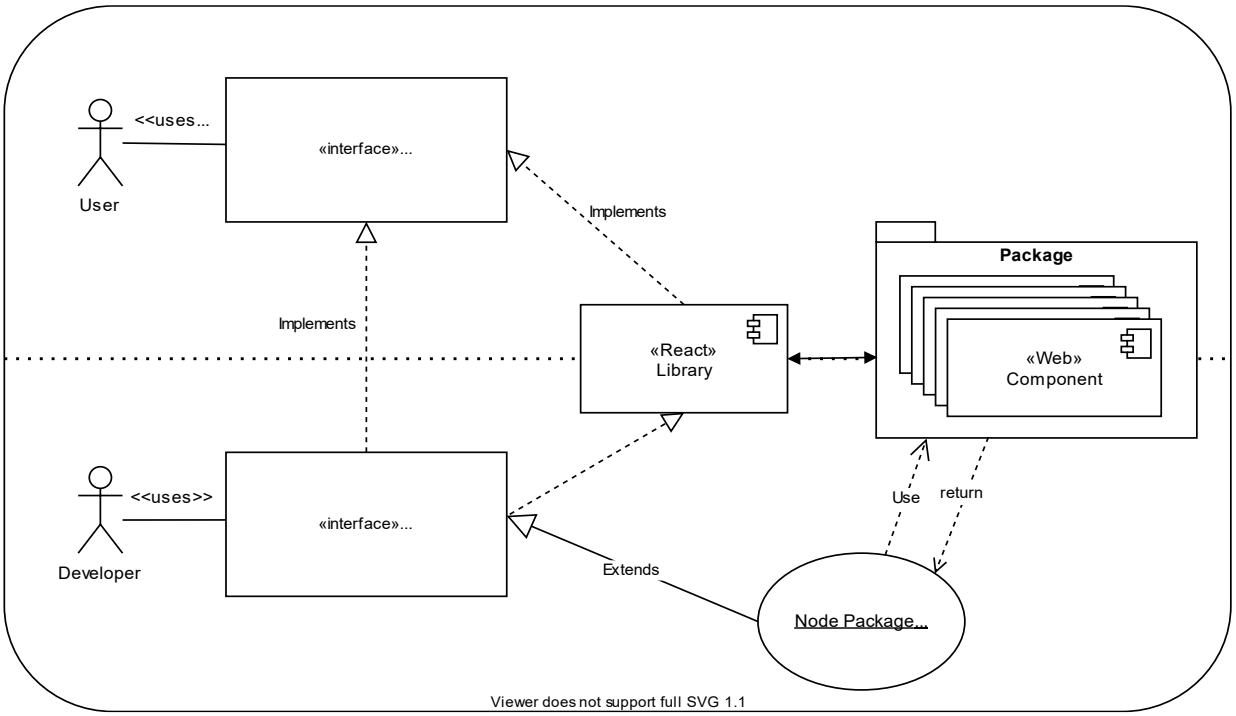



Fig.4 Final Updates Functional Requirements - Midpoint (December 2020).

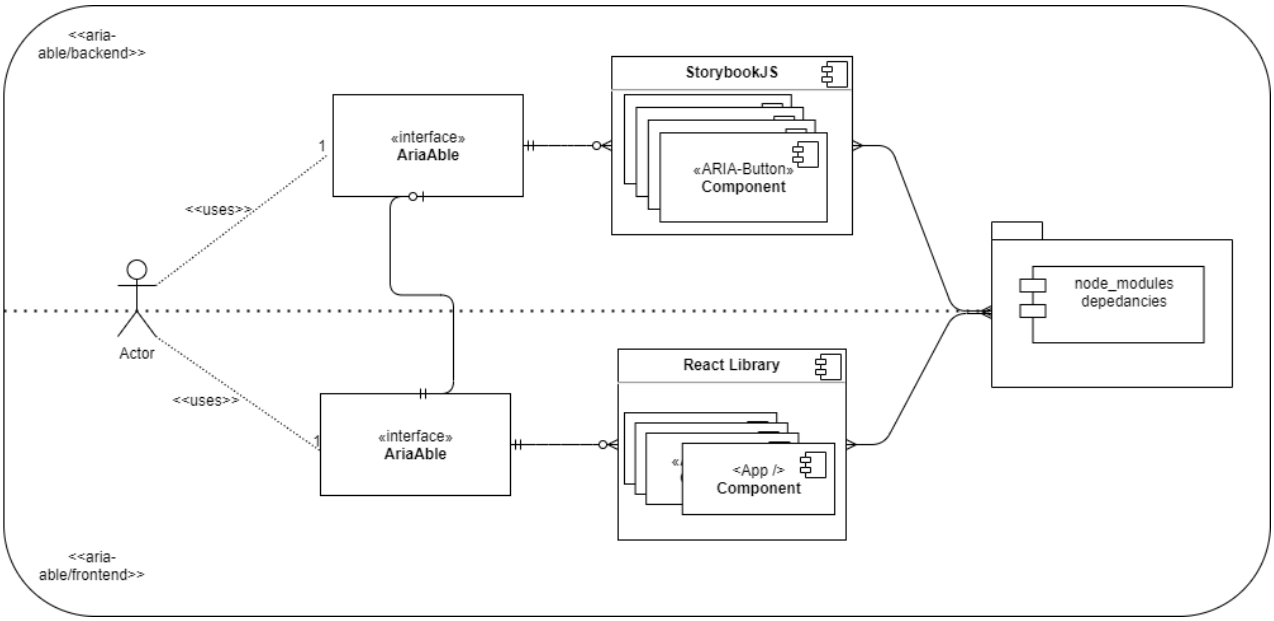



Fig.5 Final Updates Functional Requirements - Final (July 2021).

#### 2.1.1.2. Requirement 1 < **User Visits Aria-Able Frontend React** >

#### 2.1.1.3. Description & Priority

Where the user visits www.aria-able.com/frontend and sees the landing page with CTA.

#### 2.1.1.4. Use Case

**Scope**

The scope of this use case is to detail the process of the user access the website to find information for accessible web components and where to find them.

**Description**

This use case describes whether the user can click the Call to Action (CTA) to find out more and then discover elements.

**Flow Description**

**Precondition**

User has an operational machine with access to a browser and the internet, the ability to click a mouse OR to press and cycle through the TAB button multiple times with a keyboard.

**Activation**

This use case starts when the user clicks the CTA button OR scrolls down the page to see more content.

**Main flow**

1. The user log onto www.aria-able.com/frontend
2. The user sees accessible web components font in <h2> displayed with Bezier animation triggered. (Home Path (/)
3. The user clicks on the button. (target=" _blank" rel=norefferer)
4. The user sees www.aria-able.com/backend link and Storybook JS

**Alternate flow**

A1: <alt_flow>

1. The user scrolls instead of clicking onto the CTA.
2. The user presses the keyboard TAB too many times and has to loop back.
3. The user reads the Features.js section AND FAQSection.js components
4. The user clicks on "Components" on Navbar.js components.
5. The user clicks on "Contact" on Navbar.js components.

**Exceptional flow**

E1: <excep_flow>

6. The browser window fails to load the components to the screen.
7. The user exits the browser into the process of the DOM loading to the screen.
8. The website fails to store and show it's React Components.

**Termination**

The browser window has closed.

**Post condition**

The system goes into a wait state. The required URL path is waiting to be requested again.

#### 2.1.1.5. Requirement 1 < **User Visits Aria-Able Backend StorybookJS** >

#### 2.1.1.6. Description & Priority

Where the user visits www.aria-able.com/backend and sees the landing page with testing display.

#### 2.1.1.7. Use Case

**Scope**

The scope of this use case is to detail the process of the user access the Aria-Able Storybook testing library and to click on the accessibility tab to get information on the stored ARIA components.

**Description**

This use case describes whether the user can click the Call to Action (CTA) to on Accessibility Add-On and view information.

**Flow Description**

**Precondition**

User has an operational machine with access to a browser and the internet, the ability to click a mouse OR to press and cycle through the TAB button multiple times with a keyboard.

**Activation**

This use case starts when the user clicks the Accessibility Add-Ons tab OR clicks on Add-Ons

**Main flow**

5. The user log onto www.aria-able.com/backend
6. The user sees accessible web components Add-On tab displayed.
7. The user clicks on the button.
8. The user sees a modal display "pass" "warning", and "violation"
9. The user clicks on "pass "warning, and "violation"
10. The user views information accessibility with "pass" "warning" and "violation"

**Alternate flow**

A1: <alt_flow>

9. The user clicks on anything else BAR accessibility TAB or Add-Ons TAB
10. The user presses on Accessible Components tabs on left dropdown.
11. The user clicks TABS on the single components.
12. The user sees a modal display "pass" "warning" and "violation"
13. The user clicks on "pass" "warning" and "violation"
14. The user views information accessibility with "pass" "warning" and "violation"

**Exceptional flow**

E1: <excep_flow>

15. The browser window fails to load the www.aria-able.com/backend to the screen.
16. Storybook JS fails to have their servers running and nothing renders to the screen.
17. The user exits the browser into the process of the DOM loading to the screen.
18. The website fails to store and show it's React Components.

**Termination**

The browser window has closed.

**Post condition**

The system goes into a wait state. The required URL path is waiting to be requested again.

### 2.1.2. Data Requirements

With data retrieval, the component perspective of the project will involve using openly sourced specifications from the W3C and WAI-ARIA. On the React application there was an original intention to have a connected database that stored basic information and allowed newsletters info of added accessibility features suggestions. This would mean that personal email addresses will be stored and saved on secure database. Under GDPR regulations introduced by the European Union, consent must be given that gives data protections to users and developers who hold sensitive data. There was and still is potential when developing the React application to personalise the experience with a greeting of information tied to the associated account. Logically and because the tool will be an open source in the ethical areas to related to accessibility, functionality to the user was offered free of charge. No credit card information was stored in the database on the final version.

### 2.1.3. User Requirements

User requirements was an area which has direct associations with accessibility, so this was dominant focus throughout the project. User requirements met the physical and cognitive needs of users along with being able to have comfortable use of the user interface. User requirements share the same abilities as the web component process with support for assistive devices, tab focus fields to illustrate where the user is on the page, and contrast and theme settings that were explored inside the project proposal document. One method which can be used to support user requirements can be the use of Contextual Task Analysis. This is to gain insights by observing users of the application witnessing how they navigate a page. This was conducted on a face-to-face smaller aspect by members of my family in relation to the development of the landing page.

### 1.1.1. Environmental Requirements

Tackling the area of environmental environments was the reference to the interface abilities and limitations to the user interface in certain surroundings. In these environmental areas, the use of mobile technologies and touch screen devices for compatibility must be mentioned. The development of the documentation crossed over to mobile applications as well as desktop with multiple *media queries* added to ensure responsive viewing across all platforms and machines.

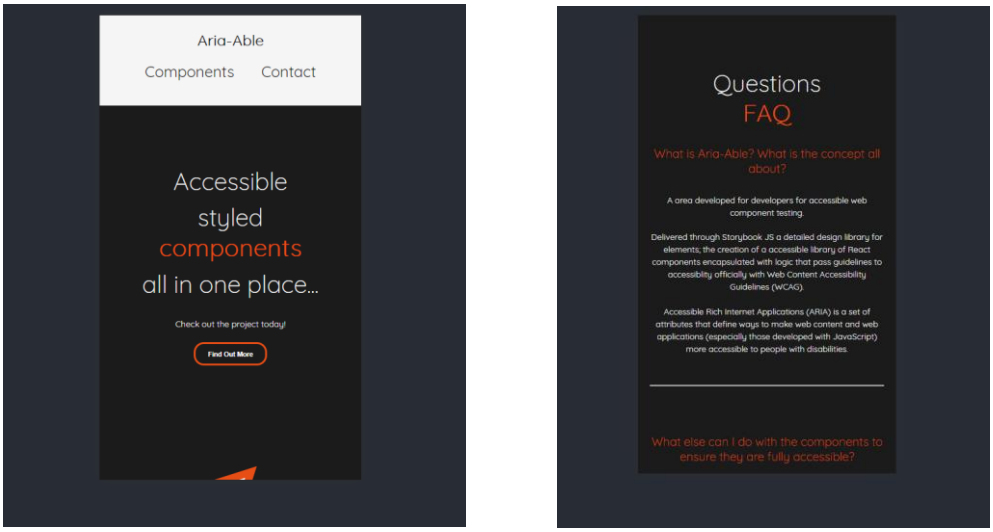



Fig.6 Responsive Media Queries Optimisations for User and Usability Requirements (React App).

### 1.1.2. Usability Requirements

For usability, the interfaces intention was to be intuitive and easy to use when accessing and exploring the application with discoverable Call To Action (CTA Button) displayed on the landing with larger text facilitating accessibility standards. In the context of the application, the intention on the UI was to focus on simplicity and structure a perspective, that can allow all individuals to access the page and have a painless experience. Usability is essentially a practicability type of requirement in relation to front-end styled development; where the application should be efficient and easy to use, along with a level of intuitiveness every time something appears on screen (e.g., such as error a warning tone for an assistive reader or animation effect). Lastly the sense of having "low perceived workload" can come into effect. This means that the interface is not cluttered or intimidating and can be friendly visiting for the first-time user. The structure of my application built in React had a de-cluttered landing screen for easy navigation. The busy areas were the documentation sections, and this was a challenge to ensure those visual areas were made enticing and interesting to the user.

## 1.2. Design & Architecture

In this section this is breakdown of the two primary areas of development with web components in Storybook and single page application (SPA) building in React. More importantly I will look at the more challenging areas of the software with React Router Dom, React Hooks, and Storybook's Add-Ons. One important note to explain is that the two areas of technologies are vast, and so a high-level analysis with visual accompaniment will be given to what I created highlighting challenging aspects of the application. Here we start with Storybook and component building, the challenges with logic and then the areas that connect more closely to design based systems being the logical option over Stencil JS for the project. After this we will explore an overview of the React framework and discover its primary concept of "state" this also occurs in Storybook; and to illustrate how both technologies share consistent similarities that also helped my learning progress throughout the stages of the project.

## 1.3. Implementation (React & Storybook)

- **Storybook**

Originally there and very heavily mentioned in the proposal there was interaction with Stencil JS which is a Vanilla JavaScript component building library. After some time working with this tool and getting to grips with its functionality it was decided that the logical aspects in achieving complex functionality was too much and a revision was needed to find an area that would help me achieve my original aims. This is when the discovery of Storybook was made as a design system library that can mimic logical behaviour. It also had a fantastic platform that allowed me to use the areas of the React library which I was more familiar with.

Once initialising a project, a "stories" file is developed that allows the user to write JavaScript and HTML combined and this is called JSX. JSX is a constant through the React library and due to its connection and similarities with codebase it only felt right to take a redirection into the project to achieve accessible component building for the final project. In a Storybook project, you can write a "story" which then gets rendered to a testing platform. This is achieved with a unique markdown file with the extension of MDX. Frontendmasters.com. (2020). Also, any configurations needed for Storybook would be in a storybook folder based in the root folder of the application.

On Storybook there can be a multitude of Add-Ons Js.org. (2021). that can activate certain functionality within the building process of building code in the Storybook MDX files. Below are the main tools I used on the platform of Storybook to facilitate and help my accessible components. These are listed and described in this table below. Additionally, the final project presentation will highlight these areas accordingly with live examples.

| | |
|---|---|
| Storybook Actions | The actions addon is used to display data received by event handler (callback) arguments in your Storybook stories' components. |
| Storybook Knobs | Knobs allow you to edit props (properties) dynamically using the Storybook UI. You can use these knobs as a dynamic variable that in turn updates the 'state'. |
| Storybook Context | Contexts creates the components to be 'polymorphic' meaning they adapt to real work contextual environments (e.g., dark theme or light theme, or multiple languages) |
| Storybook A11y | This makes the building of components become more accessible by activating conditions or requirements that are approved by the WCAG and the W3C. |

Fig.7 Full List Storybook Add-On Functionalities that are initialised in storybook folder of Aria Able backend.

Storybook's main project building is that you create a Styled Component element that had CSS classes and elements. This then gets interpolated into a functional component that can be read in React and in JavaScript. With this functional component you can apply different types of "state". State and the idea of state is essentiality the observable current properties of an active element. Is the page open or closed? Is the modal active or not? Is the button clicked? Using functions and functional programming, state is then accessed as a variable that can be assigned modified and updated by the developer. In Storybook I assigned and updated the HTML elements to that were in the syntax of MDX to be interpolated into something that is natively read by the browser window. I had a root folder consisting of utilities for my colours, themes, and font layouts of my components that plugged directly into Storybook which created added flexibility. With this file and with a command in the terminal you can then run your Storybook project from the command terminal to view your rendered components.

There was a lot of connections and associations to export and import of files declared at the top of the page and then the majority of component building was processed underneath. In total fifteen components were built and developed; once these were developed they were visually redesigned to suit the colours and theming of the Aria-Able product. Along with these styled components and the visual show, below I would like to point out some particular highlights. This was a challenging area to tackling but the idea of creating modularity and instantaneous components after setting the building blocks for the creation of the first Styled Components was the real achievement here.

- The "utils" for utilities declaration is where all my custom themes, fonts and colours are coming from. Then they are passed through down to the created elements.
- The backticks `` represent the interpolation with the Styled Components. I declared it to be used at the top of the page. "*styled.button*" represents that you want to create a button element.
- I am also using style modifiers. By passing in a string of "small" or "large" I can change the visual attributes of an element that is passed down through "props" or properties from the storybook MDX files.

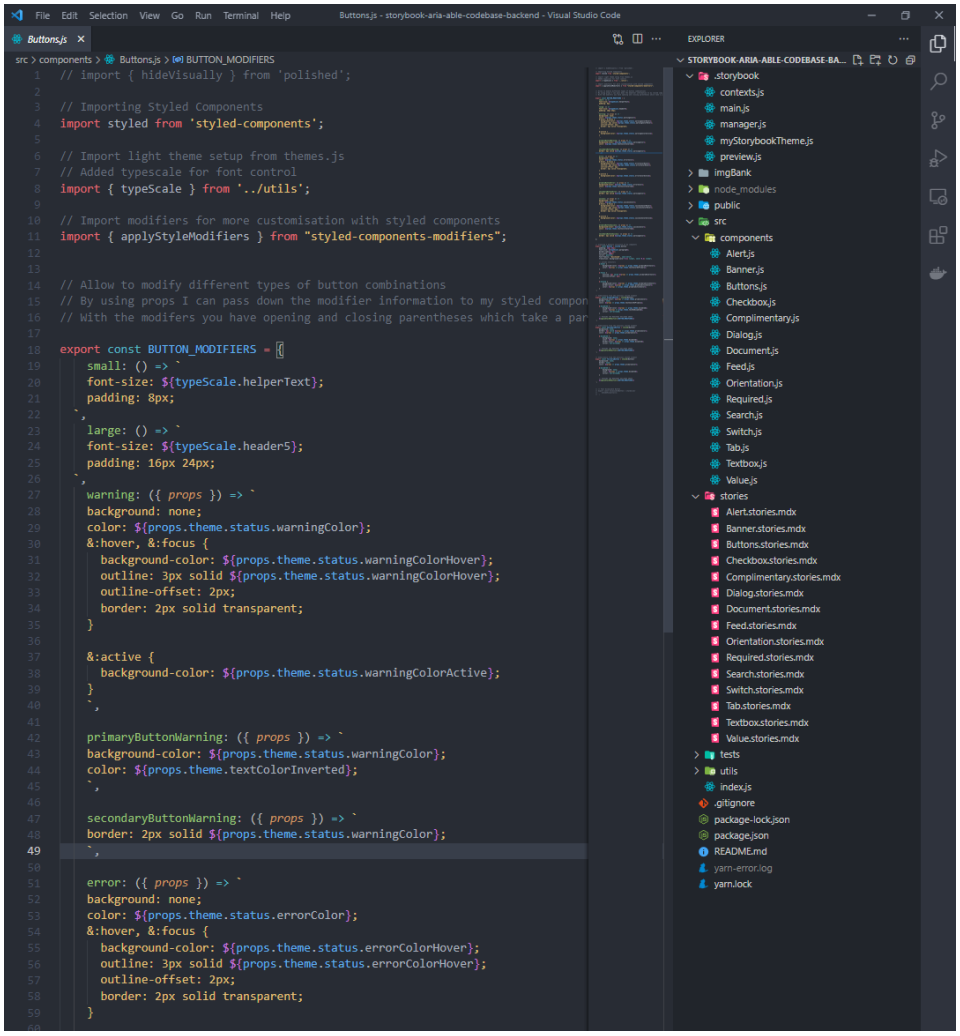



Fig.9 Storybook Styled Web Component Demonstration (Modular Approach to generating).

- **React**

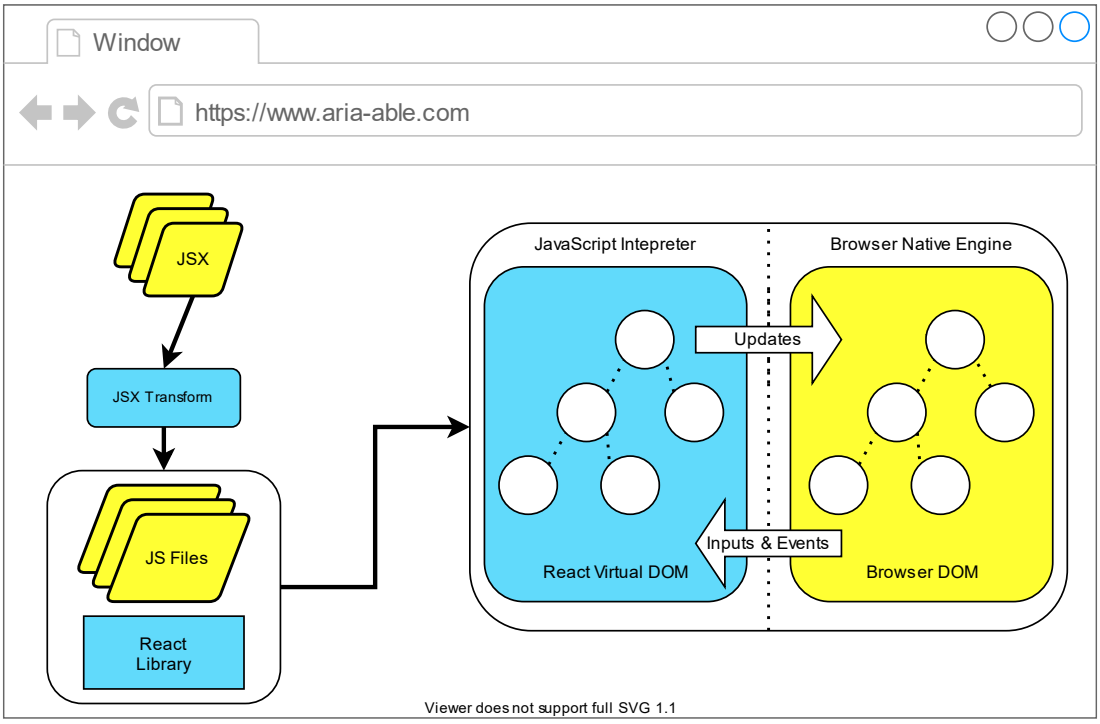



Fig.10 ReactDOM overview with diffing and state management in the browser window.

React is a component style framework where every element on a page can be broken up into sections. It was my ambition to bring my own custom web components into React. The concept of "*state*" however is important inside React, because like web components and abilities with the shadow DOM, React has its ways to manage the concept of state. To explain this concept of state with React on a high-level is to say a component created is a JavaScript object representing the ability to dynamically change or update on a screen. These components are set in one state and rendered on a screen when the code is initialised or ran. When there is a change to this component, a re-render is triggered inside the React framework. Above is a visual architectural breakdown of how React manages to share what is essentially two DOM's, the original and React own virtual DOM.

How state of a component gets discovered in React from concept is quite simple. When there is a change in the state the DOMs get compared. This practice is called "diffing". When there is change and DOM's are not identical then the virtual DOM sends a batch update to update the UI. Because the page is broken up into sections with components this means that only certain areas of the page need to be re-rendered thus resulting in a type of dynamic application over a traditional website. This is the main core concept and a feature of many features that lays in the React ecosystem. Everything in React starts out with a root element which is then rendered down from parent to child. This is a very similar process to the architecture of JavaScript and the Document Object Model where the analogy of a tree is mentioned starting out as a parent root component and then expanded out to a multitude of branches. In my React project I started with the bare bones basics of React building and by not using the new and latest function call features until I got used to the structure.

My application starts with a root element div that then gets inserted into the <App/> component via a index.js file. From there the components get rendered and pass down to the children elements representing a Pages Component with {Components, Contact, FrontPage, NotFound and PonentDetails} and then inside this is {FAQSection, FeatureSection, GlobalStyle, LandingSection, Navbar}. These pages needed a place to go and be redirected to the page, so the package of the popular React Router DOM was installed that wrapped my App component around all other components down the tree to ensure declarative routing. The elements used for React Router DOM are {BrowserRouter, Route, Switch, Link}. Developedbyed.com. (2021) BrowserRouter must be wrapped around the entire parent of components to ensure every element can become declarative. From there it then has to require Switch to ensure that each route is going to be rendered exclusively to whatever route is wrapped in Route itself. And lastly is Link which tells exactly what the string of URL you intend to be directed to. ReactRouterWebsite. (2021). A visual example taken from the code about this is presented below.

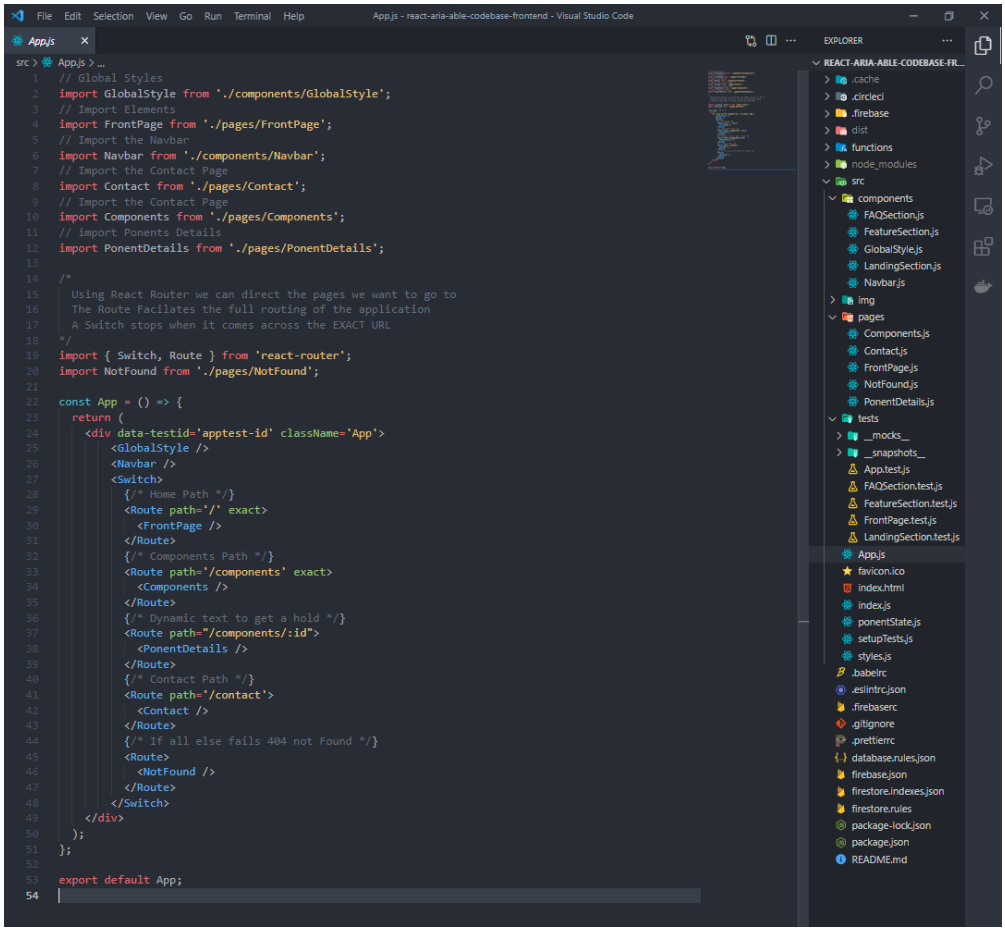



Fig.11 React Router DOM example of declarative routing in the App.js file.

When it came to the client-side application accompanying the components it was all about finessing and make that side of the project better and better. In the *package.json* file there is a massive amount of dependencies built up from scratch and I can proudly say that I built up every area of the folder along the way. In the root folder there is an array of .RC files which are configuration files for packages dedicated to linting and formatting such as Prettier and ES Lint in my project. These elements were all connecting and integrating off the React components inside the root folder making for a steady workflow.

Once the groundwork was made for the landing section which comprises of three components attached together, the next steps was to look at the aligning repository on the component side with Storybook to see how to get the data into the React application using the concept of state. There was a challenge to go about this as React had recent updates to how it manages state and dynamic content in its applications thanks to the arrival of a new concept entitled "hooks". Hooks allow you to write a functional component rather than a traditional class component without the need for a constructor where you can pass through a two-element array variable that allows you to declare and "set" something. Setting meaning that the state will be updated in the browser. Below is a visual example of this and a challenging area of the application using the useState React hook to gain access to the component objects I have placed in the React application. In fact, there are actually not one or two, but three hook examples in this piece being used Reactjs.org. (2020). *Introducing Hooks*.

- useHistory() – allowed me to get the URL of the current React Router path by at first logging simply to the console and then storing it in a variable. (This was captured in Google Chrome Dev Tools below)
- useState() – manages to store all of my components inside an array element capturing the current state of the application.
- useEffect() – ran when the component mounts to the page (as in when the user leaves and comes back to the page), so I am saying to filter and catch the current history and then mount that saved useHistory URL only ONCE to the DOM when it mounts.

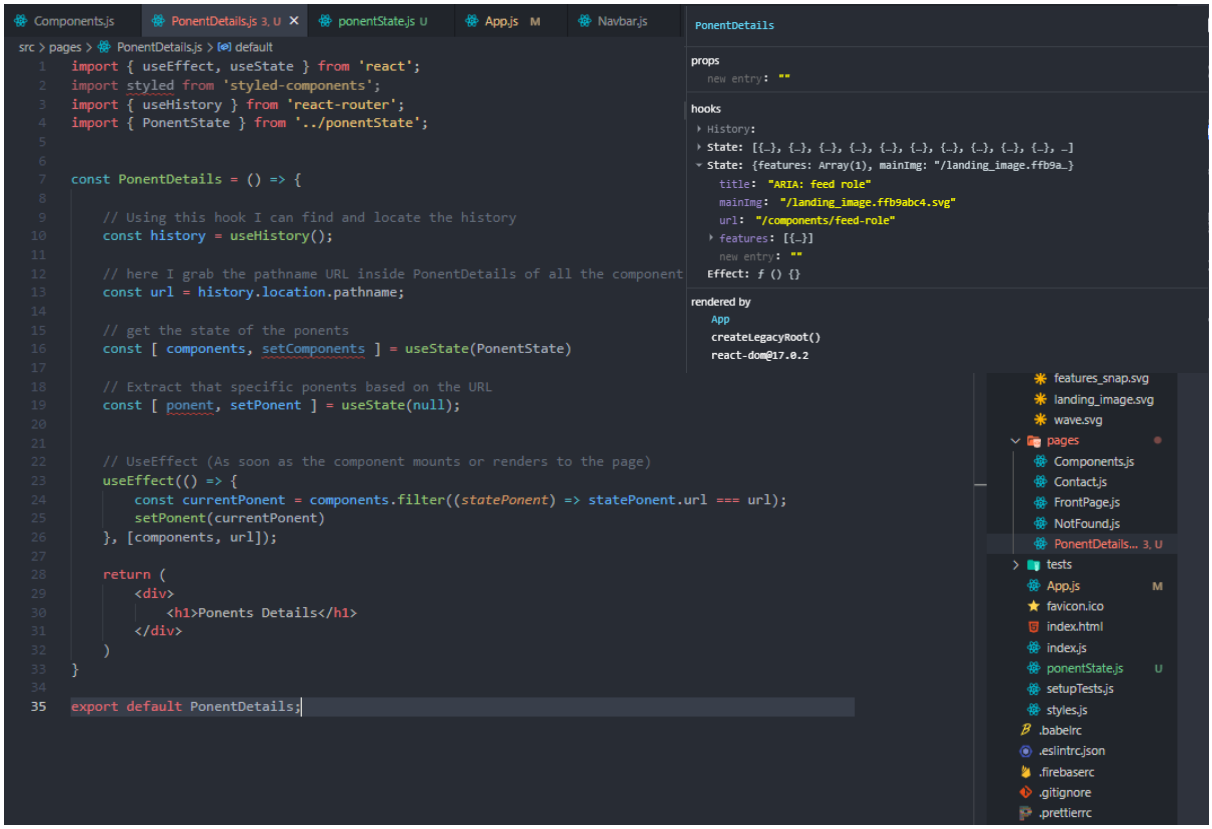



Fig.12 React Hooks with useState(), useEffect() and the useHistory() hook to keep track of the URL path above.

## 1.4. Final Graphical User Interface (GUI)


Fig.13 AdobeXD original concept artwork for landing page section of Aria-Able.

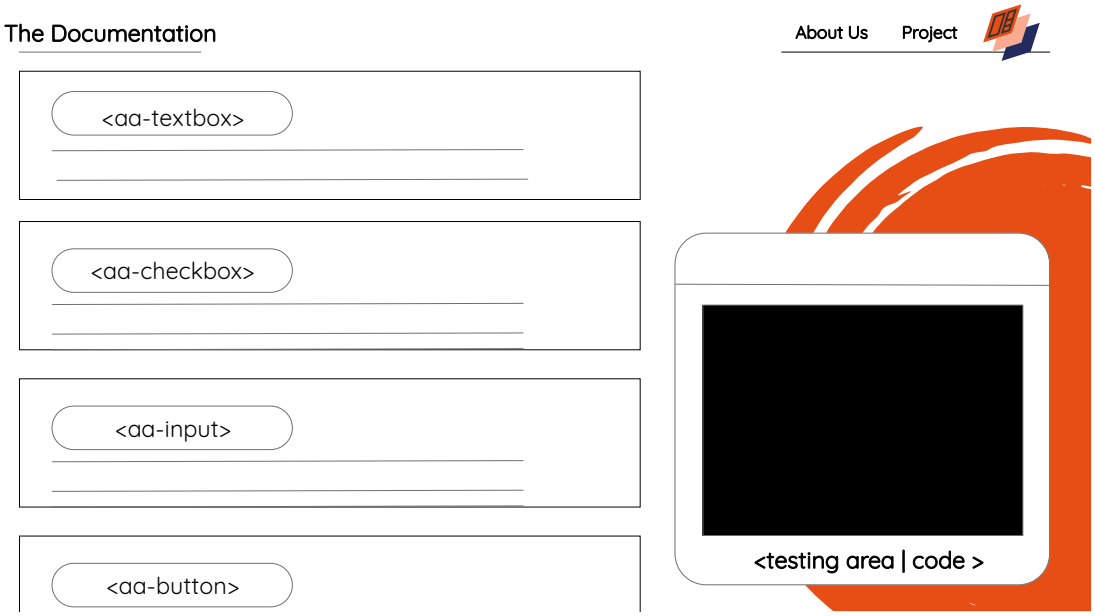



Fig.14 AdobeXD original concept artwork for “components” page section of Aria-Able.

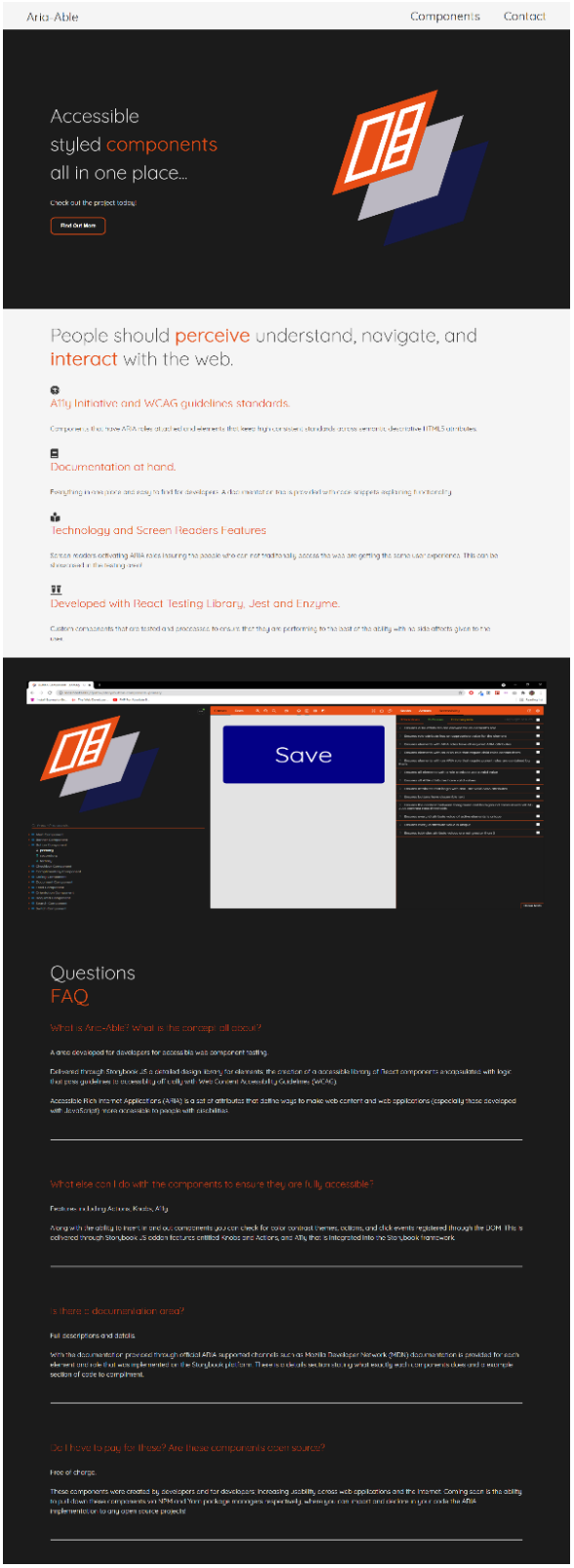
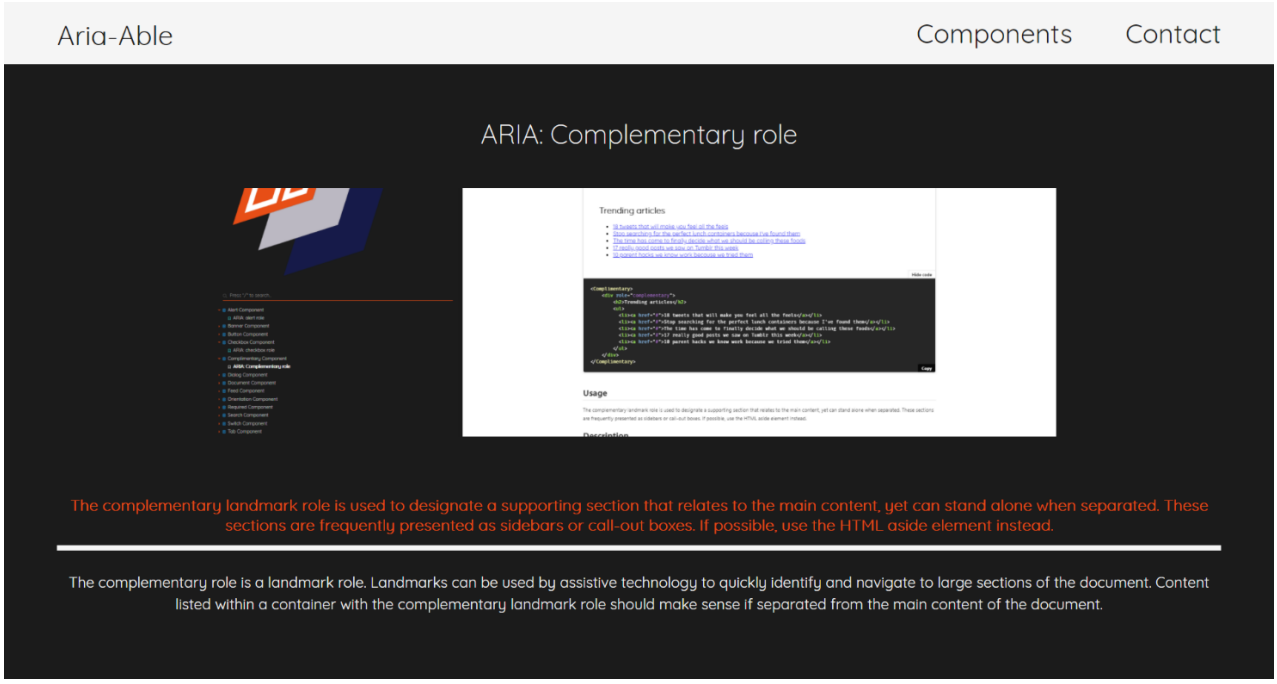



Fig.15 Final Landing Page Structure full of extended Features and FAQ Sections.

Final Landing page for the React Project was re-iterated and fine-tuned to provide a great user experience with sweeping animations and affects delivered through Framer Motion. From the top Call to Action (CTA) link you can then proceed into Storybook components testing library split up into three component sections. There is also additional links for “Components” holding the list of the elements and a “Contacts” page for future communication.




In these sections are the additional pages of the React application along with a 404 'not found' for a fallback page, that informs the user of a misdirection.

The list of components is directed into a component single entry page which gets replicated across a JavaScript style object to be rendered to the page via a feature based in React.

All styling is achieved with "Styled Components" where CSS gets translated and interpolated into visual style components.

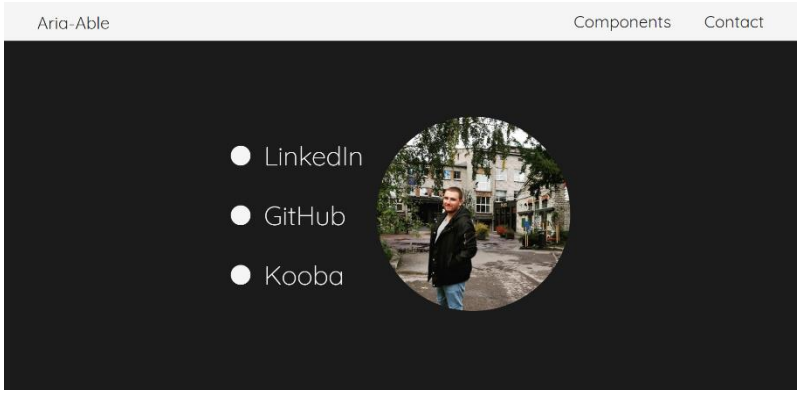
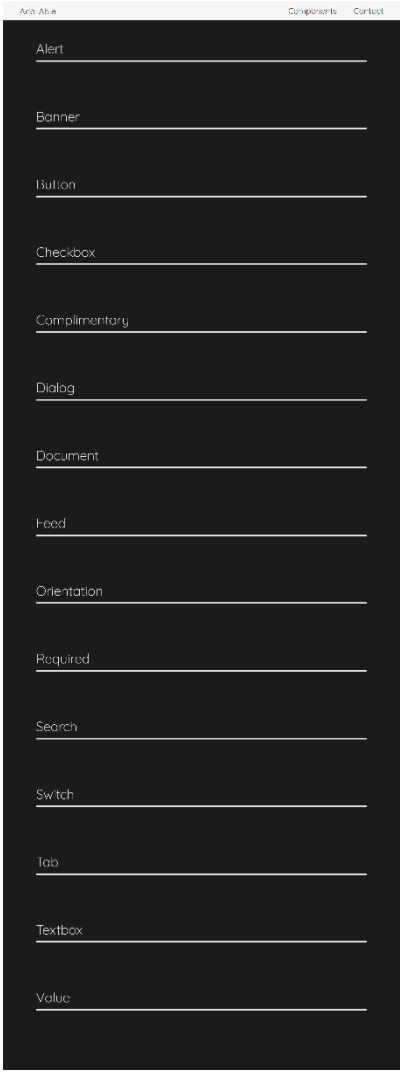



Fig.16 Final Component Pages (Static) for Aria-Able project.



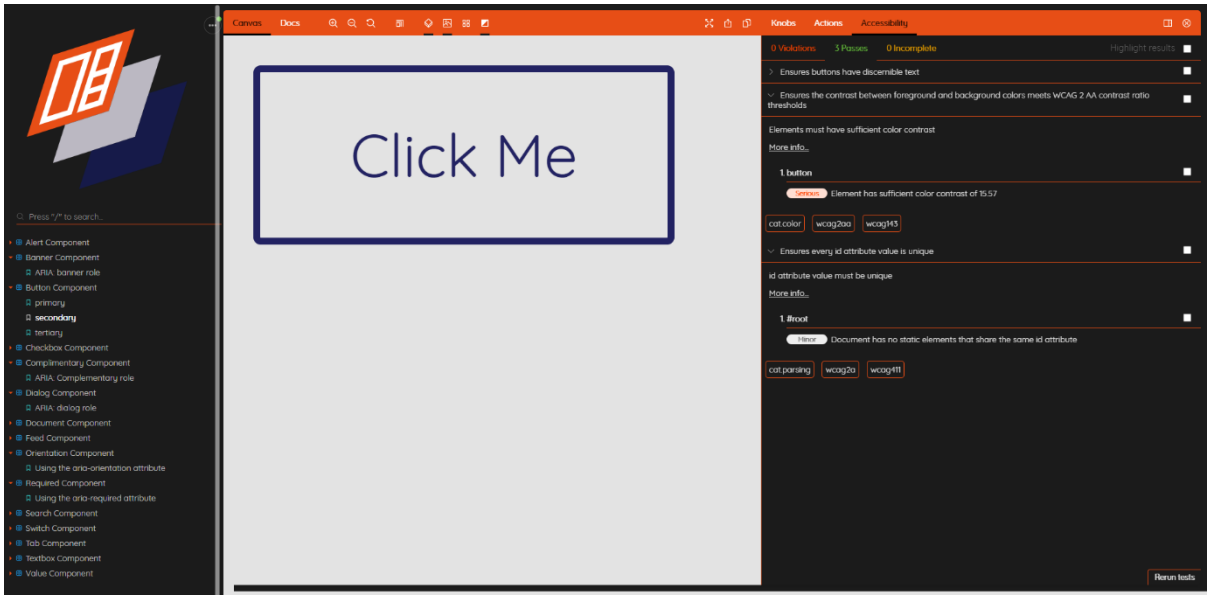



Fig.17. Testing Area inside Storybook (Canvas tab) with Secondary ARIA button.

Above are some examples of the Storybook platform that facilitates the detailed document area, actions, add-ons and more. To the left is the easily accessible list of components a user can scroll down to along with the "Canvas" acting as a testing ground for the components and "Docs" acting as a code snippet and information area. Along with this the A11y initiative measuring, and testing accessibility is also rendered to the page giving each element a series of "violations" "incompletes" and "passes" according to the guidelines of the Web Content Accessibility Guidelines (WCAG)

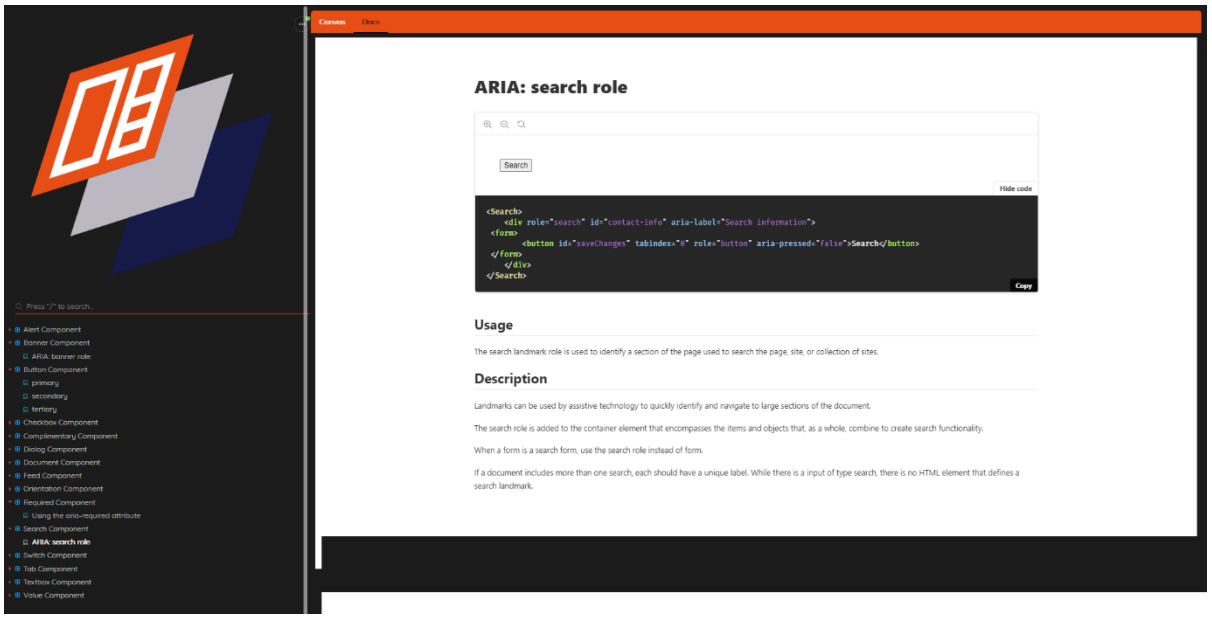



Fig.18. Documentation Area inside Storybook (Docs Add-On Tab) with Search ARIA role snippet.

## 1.5. Test Driven Development & Deployment

In the areas related to testing there was a big emphasis given to red, green and refactor concept that comes related to Test Driven Development. There was also a document created (that still remains attached detailing an outline to ongoing TDD strategy) Below is an outline of the original mind map for testing strategy for my components as well as research given into types of testing, and what suites would be more beneficial.

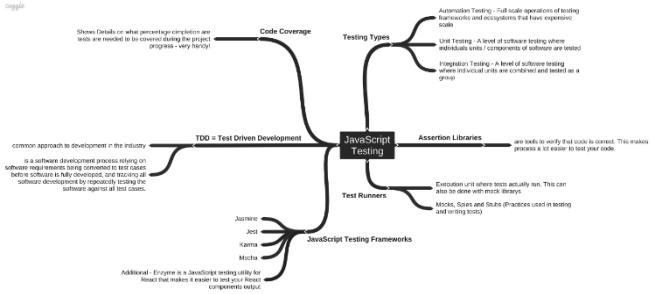
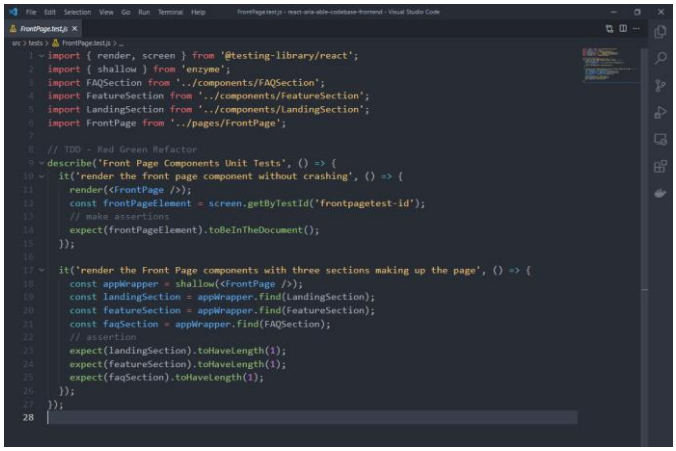



Fig.19. Testing Area MindMap for original planning of Testing library research.

In my React application there was the setup of a multitude of testing libraries that all get initialised into one *setUpTests.js* file that is located in the "*src*" directory of the project codebase. Every time a JavaScript file gets declared then all that is needed on the testing to run and is the same naming of the JavaScript file but with a test extension at the end. With a command such as *create-react-app* you can usually get the React Testing Library by default that run basic testing commands. My React application was created from a fresh empty folder, so there was the matter of installing dependencies myself to the *package.json* file. I proceeded with Jest and Enzyme Jestjs.io. (2021). as a combination of testing tools for my project. Jest is fully fledged JavaScript framework library which deals in unit integration and E2E based testing. Enzyme is a JavaScript utility for React that makes it easier for your components to test with handy dot notation function calls. Accessing elements simply by declaring them to render was not so easy at first as the React DOM tree traverses down child elements causing the tests to fail. There was a function in the Enzyme library called "*shallow*" Fraser, D. (2018) that calls a test on the parent component only and not the child components which was a nice learning curve to overcome the challenge of further testing in the project.



```
import { render, screen } from '@testing-library/react';
import { shallow } from 'enzyme';
import FAQSection from '../components/FAQSection';
import FeatureSection from '../components/FeatureSection';
import LandingSection from '../components/LandingSection';
import FrontPage from '../pages/FrontPage';

// TDD - Red Green Refactor
describe('Front Page Components Unit Tests', () => {
  it('render the front page component without crashing', () => {
    render(<FrontPage />);
    const frontPageElement = screen.getByTestId('frontpagetest-id');
    // make assertions
    expect(frontPageElement).toBeInTheDocument();
  });

  it('render the Front Page components with three sections making up the page', () => {
    const appWrapper = shallow(<FrontPage />);
    const landingSection = appWrapper.find(LandingSection);
    const featureSection = appWrapper.find(FeatureSection);
    const faqSection = appWrapper.find(FAQSection);
    // assertion
    expect(landingSection).toHaveLength(1);
    expect(featureSection).toHaveLength(1);
    expect(faqSection).toHaveLength(1);
  });
});
```

Fig.20. Shallow implementation on <FrontPage /> component inspection with assertion.

The approach to my component building was to start with the basic testing function and make it fail purposely before then proceeding with enough of test to pass before entering a refactor phase, thus delivering the concept of TDD. Assertions were made in the React and Storybook project, along with assertions to areas such as elements being present on the page, stylistic elements and CSS elements changing and more. With each file created a test suite is identified where it gets declared at the top to which then receives additional output in the terminal to determine which of my tests were passing and failing. With a batch of unit testing occurring across all components then the research into snapshot testing was carried out. Snapshot testing at first proved problematic with unusual output in the terminal, but then developed into an efficient way of checking differences or flagging renders in my codebase, as my snapshots eventually made their way into a pipeline that kicked off a build deploying to my primary repository on GitHub. This was very convenient as it allowed me to constantly refactor the code and given a better stricter codebase, that allowed adjustments to testing strategies while also not being frustrated with lack of progress at same time.

In the areas related to deployment across the two projects, there was one approach where everything was attempted to give the most professional build: while another simply targeting the build folder on a serverless strategy. Continuous Integration and Deployment (CI/CD) strategies served as a terrific workflow for my project going forward. Below is a Docker script detailing a number of commands that were originally present in my development scripts locally; but then transferred over to the platform of Circle CI Madness Labs (2019). Circle CI is a platform where you can build pipelines to host your applications on different platforms. Circle CI spun up on a container-based environment where it was installed with project dependencies, linted, formatted and tested using my NPM commands; before then finally deployed on Google Firebase platform. This platform is fully secure with HTTPS protocol and, facilitates further potential for expansion with real-time database and lambda function capability. In the future developing projects this would be seen as a very ideal approach to satisfying workflow approaches when it comes to software development strategy.

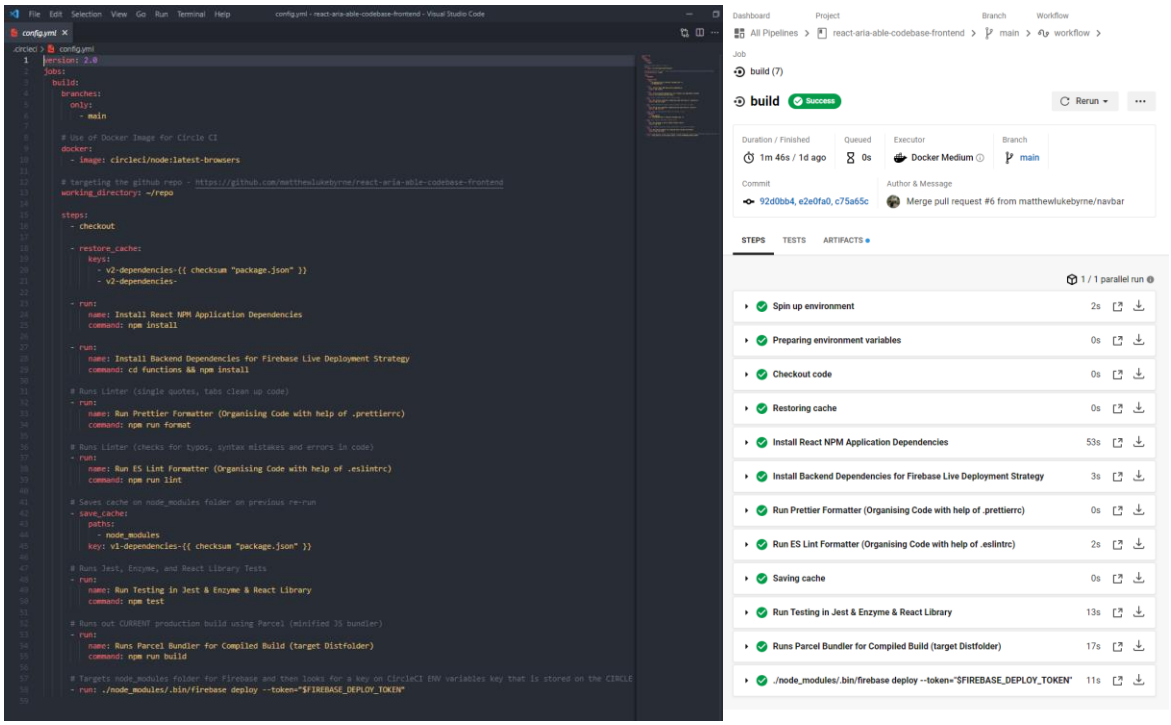



Fig.21. Circle CI & Docker configuration file for deployment strategy delivered through Google Firebase in root folder.

## 2.0 Conclusions

With the project and the completion of its current lifecycle of submission there are a number of takeaways which are encouraging for the project moving forward. Huge areas were dominated in the realm of front-end development with an emphasis of jumping into the deep end; a production scale root folder across two repositories aligning with the difficulty and challenge requirements of the project rubric. When scoping out the project and final presentation I had an emphasis to focus on where the marks were targeting with the rubric and to even highlight the best project areas with its own dedicated slide.

With advantages I am happy to have achieved a level of component building that was visual and interactive, delivering more importantly to the expectation of the recorded presentation. I have felt as an advantage also that the creation of a product or a concept as a brand identity was achieved well. By keeping and aligning to the original design, while improving the UI with iterations, the creation of something that can be viewed as a potential product was always the intention. The library currently has fifteen developed components that are stored and deployed online as an advantage. The main disadvantage here was the scope of the project in terms of the knowledge and the logical capabilities in JavaScript to achieve a system of elements that target a specific area such as accessibility. The biggest disadvantages were the adjustments of tools needed to get to this stage, such as StencilJS. There was a limitation in logic building when it came to the many complex components. Despite this I enjoyed my time on the creation of the project, working on the application inside React and also with the components inside the Storybook platform. As highlighted in the report it was the best decision to take a step back and opt for a design library system approach over the bigger scale of complex functional components – that would have been delivered to a less professional standard. The strength of the project was the brand image and created identity with a solid foundation for the future set-in place.

## 3.0 Further Development or Research

In the future the main aim for the product would be to maintain the high level of brand identity and UI whilst improving the functionality by trying to introduce the MPM aspects. The structure of how people perceive the Aria Able product should not change aesthetically but should change into logical aspects of the application. In the future it would be the main priority to take the Storybook components and convert them into packages for Node Package Manager and Yarn fulfilling the original functional requirements of the intended project, thus giving, and receiving more finite completeness to the project and along with having the currently deployed design library accessible; the chance to deliver bigger on the open-source community aspects. I would research and add to my components folder with time and energy into not just ARIA elements, but additional semantic elements based on the HTML5 spec and CSS specifications. The vision to have everything streamlined and accessible to one area that can be downloadable and imported would be the primary aim and it is with these points mentioned I would hope to take the project forward in the future.

# 4.0 References


[1] R. Dahl, ‘npm | build amazing things’, *NPM | Build Amazing Things*, Dec. 22, 2020. https://www.npmjs.com/ (accessed Dec. 22, 2020).

[2] W. W. A. Initiative (WAI), ‘WAI-ARIA Overview’, *Web Accessibility Initiative (WAI)*, Dec. 22, 2020. https://www.w3.org/WAI/standards-guidelines/aria/ (accessed Dec. 22, 2020).

[3] W. W. A. Initiative (WAI), ‘Introduction to Web Accessibility’, *Web Accessibility Initiative (WAI)*, Dec. 22, 2020. https://www.w3.org/WAI/fundamentals/accessibility-intro/ (accessed Dec. 22, 2020).

[4] B. Ollendyke, ‘Using custom elements - Web Components | MDN’, *Using the lifecycle callbacks*, Dec. 22, 2020. https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements (accessed Dec. 22, 2020).

[5] J. Walke, ‘React – A JavaScript library for building user interfaces’, *React JS*, Dec. 22, 2020. https://reactjs.org/ (accessed Dec. 22, 2020).

[6] B. Holt, ‘Typed JavaScript at Any Scale.’, *TypeScript extends JavaScript by adding types.*, Dec. 22, 2020. https://www.typescriptlang.org/ (accessed Dec. 22, 2020).

[7] I. Team, ‘Stencil’, *Stencil is a toolchain for building reusable, scalable Design Systems. Generate small, blazing fast, and 100% standards-based Web Components that run in every browser.*, Dec. 22, 2020. https://stenciljs.com/ (accessed Dec. 22, 2020).

[8] P. Bigger, ‘Continuous Integration and Delivery - CircleCI’, *Circle CI*, Dec. 22, 2020. https://circleci.com/ (accessed Dec. 22, 2020).

[9] B. Ollendyke, ‘webcomponents.org’, *Web Components*, Dec. 22, 2020. https://www.webcomponents.org/ (accessed Dec. 22, 2020).

[10] M. Kulkarni, ‘Digital accessibility: Challenges and opportunities’, *IIMB Manag. Rev.*, vol. 31, no. 1, pp. 91–98, Mar. 2019, doi: 10.1016/j.iimb.2018.05.009.

[11] M. Mays Espino, ‘Website Accessibility for Persons with Disabilities: The Why & How’, *Website Accessibility for Persons with Disabilities: The Why & How*, Dec. 22, 2020. https://www.americanbar.org/groups/business_law/publications/blt/2016/12/07_espino/ (accessed Dec. 22, 2020).

[12] C. Roland and J. Smith, ‘WebAIM: Web Accessibility In Mind’, *Web Accessibility Initiative (WAI)*, 22.12/20. https://webaim.org/ (accessed Dec. 22, 2020).

[13] M. Williams, ‘Home - The A11Y Project’, *The A11Y Project is a community-driven effort to make digital accessibility easier.*, Dec. 22, 2020. https://a11yproject.com/ (accessed Dec. 22, 2020).

[14] J. W. Hyatt, ‘Polymer Project’, *Polymer*, Dec. 22, 2020. https://www.polymer-project.org/ (accessed Dec. 22, 2020).

[15] A. Software, ‘Jira | Issue & Project Tracking Software | Atlassian’, *Jira By Agile Temas*, Dec. 22, 2020. https://www.atlassian.com/software/jira (accessed Dec. 22, 2020).

[16] S. B.-H. CPACC, ‘Accessibility and Dark UI Themes’, *Medium*, Sep. 16, 2019. https://uxdesign.cc/accessibility-and-dark-ui-themes-f01001339b65 (accessed Dec. 22, 2020).

[17] J. Dijk, ‘Production-Ready Animation Library for React | Framer Motion’, *When animating between two separate components, Framer Motion will take care of everything in between.*, Dec. 22, 2020. https://www.framer.com/motion/

[18] J. F. Lima, G. M. Caran, L. F. R. Molinaro, and D. F. Garrossini, “Analysis of Accessibility Initiatives Applied to the Web,” *Procedia Technology*, vol. 5, pp. 319–326, 2012, doi: 10.1016/j.protcy.2012.09.035.

[19] WAI), “Introduction to Web Accessibility,” *Web Accessibility Initiative (WAI)*, 2019. https://www.w3.org/WAI/fundamentals/accessibility-intro/ (accessed Dec. 22, 2020).

[20] “How Businesses Can Defeat Website Accessibility Lawsuits | JD Supra,” *JD Supra*, 2019. https://www.jdsupra.com/legalnews/how-businesses-can-defeat-website-accessed Dec. 22, 2020).

[21] “Web Accessibility,” *European Commission*, Nov. 30, 2012. https://ec.europa.eu/digital-single-market/en/web-accessibility (accessed Dec. 22, 2020).

Additional References Area (Harvard)

Js.org. (2021). *Accessibility Addon | Storybook*. [online] Available at: https://storybook.js.org/addons/@storybook/addon-a11y [Accessed 2 Aug. 2021].

Frontendmasters.com. (2020). *Design Systems with Storybook & React*. [online] Available at https://frontendmasters.com/workshops/design-systems-storybook/ [Accessed 2 Aug. 2021].

Developedbyed.com. (2021). *The Creative React and Redux Course*. [online] Available at: https://developedbyed.com/p/the-creative-react-and-redux-course [Accessed 2 Aug. 2021].

Js.org. (2021). *@storybook/addon-contexts Addon | Storybook*. [online] Available at: https://storybook.js.org/addons/@storybook/addon-contexts [Accessed 2 Aug. 2021].

Js.org. (2017). *Knobs Addon | Storybook*. [online] Available at: https://storybook.js.org/addons/storybook-addon-knobs-color-options [Accessed 2 Aug. 2021].

Fraser, D. (2018). *Testing React with Jest and Enzyme I - CodeClan - Medium*. [online] Medium. Available at: https://medium.com/codeclan/testing-react-with-jest-and-enzyme-20505fec4675 [Accessed 2 Aug. 2021].

Jestjs.io. (2021). *Getting Started · Jest*. [online] Available at: https://jestjs.io/docs/getting-started [Accessed 2 Aug. 2021].

Github.io. (2019). *Introduction · Enzyme*. [online] Available at: https://enzymejs.github.io/enzyme/ [Accessed 2 Aug. 2021].

Parceljs.org. (2021). *Getting Started*. [online] Available at: https://parceljs.org/getting_started.html [Accessed 2 Aug. 2021].

ReactRouterWebsite. (2021). *React Router: Declarative Routing for React*. [online] Available at: https://reactrouter.com/web/guides/quick-start [Accessed 2 Aug. 2021].

Reactjs.org. (2020). *Introducing Hooks – React*. [online] Available at: https://reactjs.org/docs/hooks-intro.html [Accessed 2 Aug. 2021].

Js.org. (2021). *Theming*. [online] Available at: https://storybook.js.org/docs/react/configure/theming [Accessed 2 Aug. 2021].

CircleCI. (2019). *Automatically deploy a Gatsby site to Firebase Hosting*. [online] Available at: https://circleci.com/blog/automatically-deploy-a-gatsby-site-to-firebase-hosting/ [Accessed 2 Aug. 2021].

Circleci.com. (2021). *Configuring CircleCI - CircleCI*. [online] Available at: https://circleci.com/docs/2.0/configuration-reference/ [Accessed 2 Aug. 2021].

Holt, B. (2021). *Complete Intro to React, v6*. [online] Frontendmasters.com. Available at: https://frontendmasters.com/courses/complete-react-v6/ [Accessed 2 Aug. 2021].

Madness Labs (2019). *Continuous Integration with Stencil Firebase and CircleCI. YouTube*. Available at: https://www.youtube.com/watch?v=LCECTPbSZnM [Accessed 2 Aug. 2021].

# 5.0 Appendices

This section should contain information that is supplementary to the main body of the report.

## 5.1. Project Proposal

Project Proposal
Project "ARIA-able"
08/11/2020
Computing BSc Honours
Software Development Stream
2021
Matthew Byrne
**X17138744**
x17138744@student.ncirl.ie

**Contents**



### 1.0 Objectives

With my final year project, I am setting out to create an accessibility component library, making the internet easier for everyone. To reach my objective the project will consist of two areas of delivery. This is due to the ambitious scale of the project, and types of integrated technologies used. The first area of delivery would be the accessibility tool itself. The tool will act as an installer or bundled package that lets you add instant functionality with HTML tags. How this works is by using technologies and tools that are already available open source in the industry and an official standard. The main technology for creating this accessibility tool is Web Components [9]. Web components have access to ARIA elements and attributes. ARIA stands for Accessible Rich Internet Applications [2].

ARIA supplements HTML tags with extra functionality for accessibility. These elements are a set of attributes that get added by a developer in HTML that define accessible content with better clarity for people who cannot normally access websites in the traditional way. An example of this ARIA functionality would be a screen reader which calls out the information text on a web page. An attribute would have to be added within the HTML code for semantics, to make screen reader functionality activated in the browser. With JavaScript and a web component we can abstract this logic from the developer into a software tool that manipulates the document object model. Logic will be written and bundled, that gives functionality to the user without adding further attributes to HTML mark-up. This tool can then be imported and declared for use to make developers lives easier when bringing in accessibility to websites. There are many ARIA attributes, and these attributes extend different types of functionalities. With this proposed tool, functionality can not only be included with ARIA as a standard but built upon to form a library that can be constantly updated. This idea can also be extended to make UI accessible elements, potential for an accessibility library that could be done in the future. The second area of the project will be a documentation style website on how to use my accessibility tool with guides and steps. This website will be built using a component style framework called React [5] that deals in reusable components and JavaScript logic to handle state in web applications. Because of this component functionality in React and with web components JavaScript, we can bring in our little pieces of logic into React. We can combine these together resulting in an accessibility tool and accessible webpage simultaneously.

## 2.0 Background

This concept was derived from a passion of web development leading onto other technologies. This is where I discovered that website accessibility is a vital area for inclusion and diversity of individuals. Accessibility is an area in development that overlooked when building out features of an application. There can be times, for junior developers to only realise when working in an established company, how important accessibility features is to be implemented in an application. If no implementation is included, there can be problems with legality and even global human rights issues for companies to be exposed [10]. One is five people in the world have a disability, which can equate to a certain percentage who interact with the internet daily [11]. One found definition of accessibility as a definition is people with disabilities can perceive, understand, navigate, and contribute to the internet [3]. With the word of contribution in mind, I decided to research and exploring the idea of developing a software tool. This tool can make it easier for people developing websites, to implement accessibility strategies better in their applications.

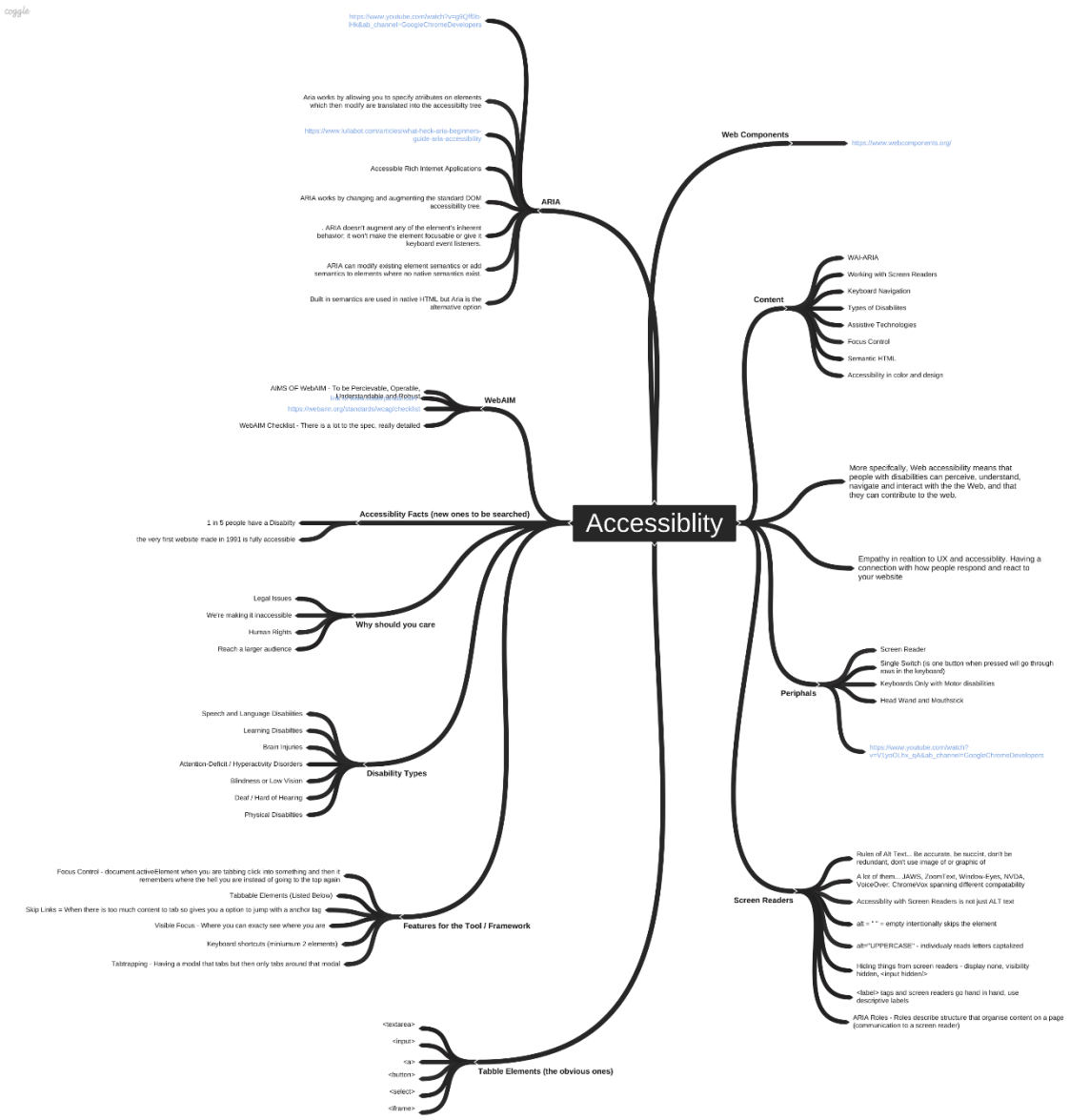



Fig.1. Accessibility Mind Map

From this point onwards and with some experience with HTML, I seen that ARIA attributes are many and are scattered everywhere on websites. ARIA works by allowing you to specify certain attributes on elements which in turns gets updated in the browser. I wanted to go through some websites I have created myself and then bring in these ARIA elements. This is where the idea for this project originated, to say why can't this be easier for people to implement ARIA elements? With Node and the Node Package Manager library (NPM) you can install a tool, and then import the named tool at the top of your code [7]. So, with this knowledge I decided to shape my idea for accessibility and my final software project. Important organisations such as W3C and ECMAScript bodies facilitate accessibility information in documentation on the internet. It is their job to keep web standards up to date and collaborate with other organisations that represent accessibility awareness. Another of these organisations is Web AIM [12], which provides a checklist and a specification standard to ensure developers have full accessibility in their applications.

After research on these organisations, there was further evaluation to find out who are the biggest leaders when it comes to accessibility. One company which are popular, Google who have podcasts, tools, and plugins dedicated to accessibility in browsers and devices. Also, several companies such as Netlify and GitHub have pushed campaigns for constant easier digital accessibility through their "a11y" initiative [13].

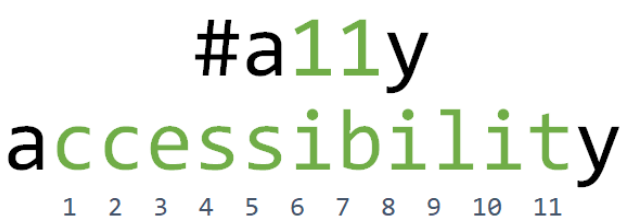



Fig.2. The A11y accessibility initiative. [9]

It took time to understand and implement the docs effectively as they are so detailed, and this is probably by design. At times official documentation can be overly complicated when it does not need to be. By creating this tool, I hope that I manage to abstract the complexity of initialising accessible elements and in turn make it easier for everyone to install a useful software tool. Documentation was a very important aspect when it came to the idea of the project, as through my time developing software, I have learnt its vital to progress with decent coding material. There is a consensus within the developer community that there can be very bad and great documentation at scattered across the web. Personally, I think it is a required skill and a technique. Ethically I feel this is an important tool to create, as with the current global climate more people have a higher dependency on internet connections; and so, it is important to ensure everyone has equal access.

## 3.0Technical Approach

The primary technology I intend to use to create accessible logic will be web components. Components are pieces of logic created in vanilla JavaScript that allow you to create separate pieces of functionality on a webpage. The technical approach is to simply create accessible web components. Web components are built up of four primary technologies, the Shadow DOM, HTML template, Custom Elements and ES Modules [4]. The Shadow DOM which is an imitator of the document object model, isolates whatever styling and behaviour is dedicated to a certain component. A HTML Template is the HTML tags of the component. Custom elements are new HTML tags that you define yourself, a visual sample is displayed below to illustrate this. As per W3C guidelines it is recommended to always have a hyphen in the name. This hyphen allows the browser to tell the difference between custom elements and regular DOM elements. Lastly are ES Modules, with which you can convert your web components to a module style package. You can declare them as an import at the top of the page and make the code open source on Node and NPM. (Node Package Manager) [7]. This will be first time I will work on web components technology, a new tool to expand my areas of knowledge. There are versions of what I intend to do with adding elements for specific accessibility functionalities. Most of these examples would be

UI based libraries that have remained popular over time. The closest example of similarity would be Google's Polymer [14] project as this was one of the original concepts of web components, creating tiny pieces of functionality to users, breaking up a page into sections rather than one big section of rendered code. The difference with Polymer and the project, is the intention of creating a software tool primarily on accessibility functionality and logic rather than a range of numerous UI areas that Polymer give.

Fig.3. A Web component defined with the custom element "user-card" tag.


With web components as my primary way of making my accessibility logic using ARIA elements, code will be split up into multiple files and then bundled together to form one module or package. This will be the primary technical approach taken with the building of the software tool. Once this ES Module style package is formed, functionality can be implemented with import declarations [4]. The second stage of the project is a JavaScript framework to be built complimenting this software tool. This framework will have state management logic throughout its pages and will show a command to download the accessibility tool, plus a detailed documentation area on how to use it. The technical approach is creating components but in the logic of React, which requires different syntactic rules and structure compared to the first half of the project.

## 4.0 Project Plan

After working on previous projects with Microsoft Project, a paid licence is usually required to take advantage of the software without Citrix. Because of this I decided to look for alternative options with project planning software. I was determined in this search to go with a platform that is respected and a standard, so the choice was with Atlassian and Jira [15]. This software has been used by myself in a previous project and can be beneficial for documentation and version control. Currently with Jira and for the future time of the project, I will run off three main areas for project management. These areas are a visual roadmap, a Kanban dedicated to weekly sprints, and a Gannt chart plugin that updates across every area of Jira for flexibility. One main reason I also wanted to work with Jira is upcoming integration & unit testing research that I hope to implement in the coming weeks. This is where a lot of project planning such as sprint documents, code checks, and commits in Jira will really come into their own as a plus for a project, delivering highly detailed pieces of documentation in one resting place.

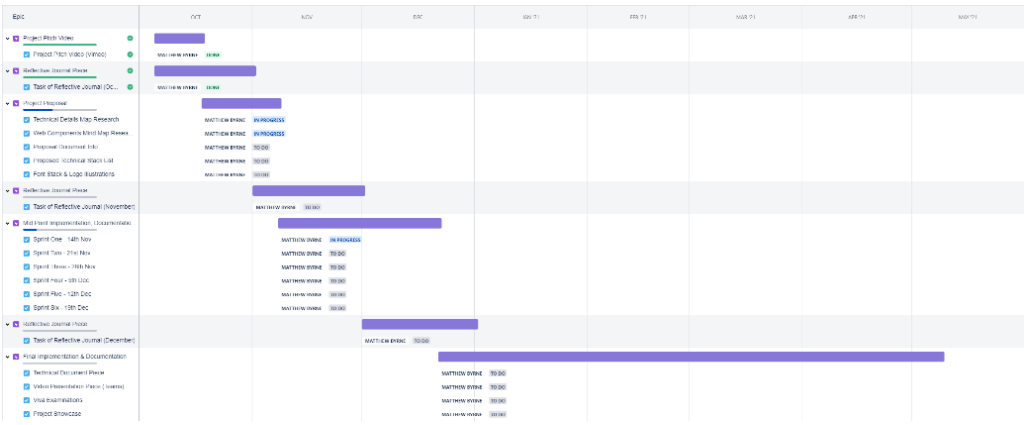
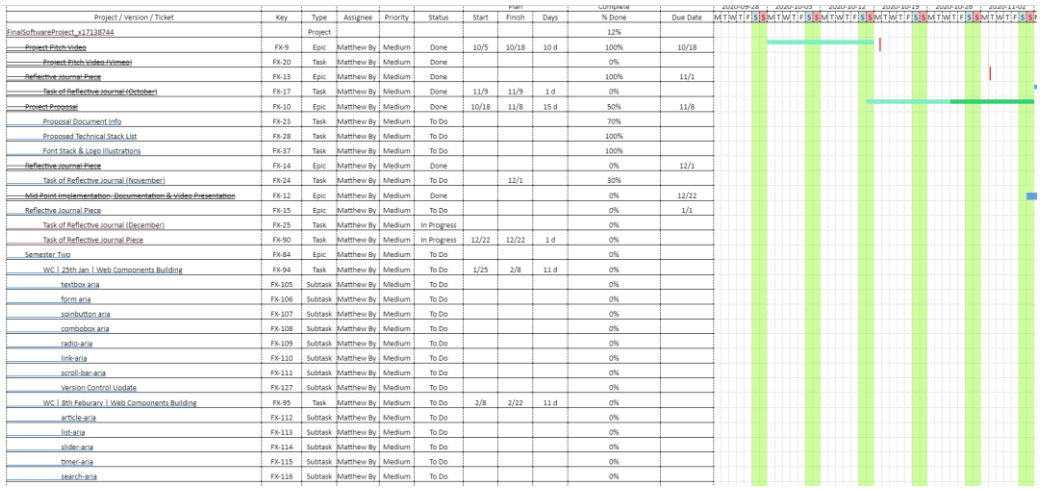
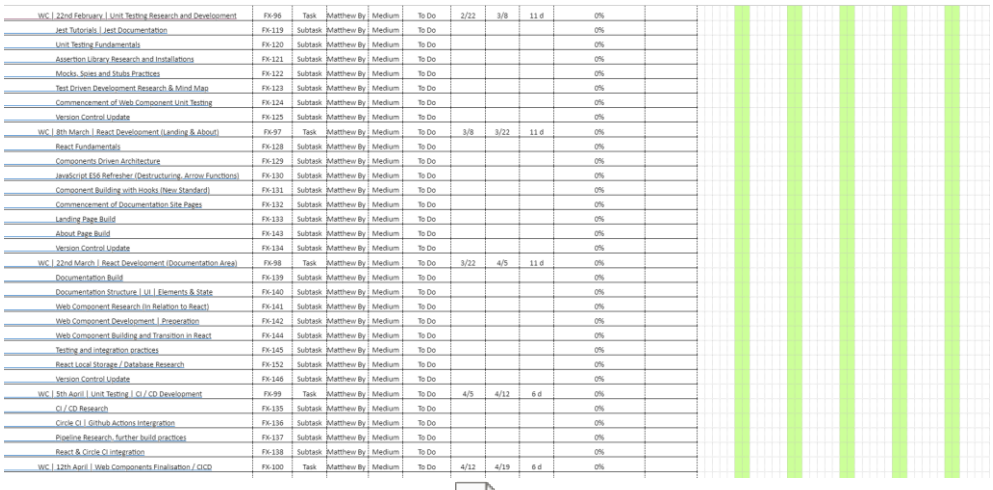


Fig.4. Jira Project Management Overview.



| Project / Version / Ticket | Key | Type | Assignee | Priority | Status | Start | Finish | Days | % Done | Due Date |
|---|---|---|---|---|---|---|---|---|---|---|
| FinalSoftwareProject_x17138744 | | Project | | | | | | | 12% | |
| ~~Project Pitch Video~~ | FX-9 | Epic | Matthew By | Medium | Done | 10/5 | 10/18 | 10 d | 100% | 10/18 |
| ~~Project Pitch Video (Vimeo)~~ | FX-20 | Task | Matthew By | Medium | Done | | | | 0% | |
| ~~Reflective Journal Piece~~ | FX-13 | Epic | Matthew By | Medium | Done | | | | 100% | 11/1 |
| ~~Task of Reflective Journal (October)~~ | FX-17 | Task | Matthew By | Medium | Done | 11/9 | 11/9 | 1 d | 0% | |
| ~~Project Proposal~~ | FX-10 | Epic | Matthew By | Medium | Done | 10/18 | 11/8 | 15 d | 50% | 11/8 |
| Proposal Document Info | FX-23 | Task | Matthew By | Medium | To Do | | | | 70% | |
| Proposed Technical Stack List | FX-28 | Task | Matthew By | Medium | To Do | | | | 100% | |
| Font Stack & Logo Illustrations | FX-37 | Task | Matthew By | Medium | To Do | | | | 100% | |
| ~~Reflective Journal Piece~~ | FX-14 | Epic | Matthew By | Medium | Done | | | | 0% | 12/1 |
| Task of Reflective Journal (November) | FX-24 | Task | Matthew By | Medium | To Do | | 12/1 | | 30% | |
| ~~Mid Point Implementation, Documentation & Video Presentation~~ | FX-12 | Epic | Matthew By | Medium | Done | | | | 0% | 12/22 |
| Reflective Journal Piece | FX-15 | Epic | Matthew By | Medium | To Do | | | | 0% | 1/1 |
| Task of Reflective Journal (December) | FX-25 | Task | Matthew By | Medium | In Progress | | | | 0% | |
| Task of Reflective Journal Piece | FX-90 | Task | Matthew By | Medium | In Progress | 12/22 | 12/22 | 1 d | 0% | |
| Semester Two | FX-84 | Epic | Matthew By | Medium | To Do | | | | 0% | |
| WC \| 25th Jan \| Web Components Building | FX-94 | Task | Matthew By | Medium | To Do | 1/25 | 2/8 | 11 d | 0% | |
| textbox aria | FX-105 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| form aria | FX-106 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| spinbutton aria | FX-107 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| combobox aria | FX-108 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| radio-aria | FX-109 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| link-aria | FX-110 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| scroll-bar-aria | FX-111 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| Version Control Update | FX-127 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| WC \| 8th Febuarary \| Web Components Building | FX-95 | Task | Matthew By | Medium | To Do | 2/8 | 2/22 | 11 d | 0% | |
| article-aria | FX-112 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| list-aria | FX-113 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| slider-aria | FX-114 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| timer-aria | FX-115 | Subtask | Matthew By | Medium | To Do | | | | 0% | |
| search-aria | FX-116 | Subtask | Matthew By | Medium | To Do | | | | 0% | |

Fig.6. UPDATED: Jira Project Management generated Gannt chart (attached with this document below)



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WC \| 22nd Febrauary \| Unit Testing Research and Development | FX-96 | Task | Matthew By | Medium | To Do | 2/22 | 3/8 | 11 d | 0% |
| Jest Tutorials \| Jest Documentation | FX-119 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Unit Testing Fundamentals | FX-120 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Assertion Library Research and Installations | FX-121 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Mocks, Spies and Stubs Practices | FX-122 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Test Driven Development Research & Mind Map | FX-123 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Commencement of Web Component Unit Testing | FX-124 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Version Control Update | FX-125 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| WC \| 8th March \| React Development (Landing & About) | FX-97 | Task | Matthew By | Medium | To Do | 3/8 | 3/22 | 11 d | 0% |
| React Fundamentals | FX-128 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Components Driven Architecture | FX-129 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| JavaScript ES6 Refresher (Destructuring, Arrow Functions) | FX-130 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Component Building with Hooks (New Standard) | FX-131 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Commencement of Documentation Site Pages | FX-132 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Landing Page Build | FX-133 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| About Page Build | FX-143 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Version Control Update | FX-134 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| WC \| 22nd March \| React Development (Documentation Area) | FX-98 | Task | Matthew By | Medium | To Do | 3/22 | 4/5 | 11 d | 0% |
| Documentation Build | FX-139 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Documentation Structure \| UI \| Elements & State | FX-140 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Web Component Research (In Relation to React) | FX-141 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Web Component Development \| Preperation | FX-142 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Web Component Building and Transition in React | FX-144 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Testing and integration practices | FX-145 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| React Local Storage / Database Research | FX-152 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Version Control Update | FX-146 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| WC \| 5th April \| Unit Testing \| CI / CD Development | FX-99 | Task | Matthew By | Medium | To Do | 4/5 | 4/12 | 6 d | 0% |
| CI / CD Research | FX-135 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Circle CI \| Github Actions Intergration | FX-136 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| Pipeline Research, further build practices | FX-137 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| React & Circle CI integration | FX-138 | Subtask | Matthew By | Medium | To Do | | | | 0% |
| WC \| 12th April \| Web Components Finalisation / CICD | FX-100 | Task | Matthew By | Medium | To Do | 4/12 | 4/19 | 6 d | 0% |

x17138744_GanntMidpoint.xlsx

Jira has several plugins to extend the platform on top of its areas and will be used to the fullest potential in the project as a means of total integration. Above are screenshots of the Gannt Chart created and integrated with Jira. In the midpoint presentation weekly sprints

are broken down with tasks. In the future I really hope to keep up with this level of documentation, as it will be of great benefit in the final stages of the project.

### 5.0Technical Details

The primary coding language for the project will be JavaScript. Due to its availability in the browser and with the current style application I was aiming for, it made sense to opt for a JavaScript style project. Web components are made primarily in vanilla JavaScript but there will be potential to offer a tool called Stencil.js to create and build web components faster [7]. Stencil.js is essentially a toolchain for building and compiling web components. With Stencil.js you can even write your components in TypeScript or create your components very similar to how the React framework declares its component code [6]. The aim in the technical aspect is to master the elements of Web Components first in vanilla JavaScript before advancing this knowledge into a web component framework. Gradual learning will be made throughout the process of the sprints, and the ability to add further technologies will occur over the duration of the project building process. Node.js must be introduced to the project if I have any intention in creating a bundled module that is open source available. Node.js is a library that allows JavaScript to access files locally on the local machine. The prospect of this is quite exciting as I have never created open-source project officially before, but a technical learning curve will be necessary, and this will be considered. This will be highlighted in Sprint documentation and in Jira for the future. Along with JavaScript there will be the two remaining front-end technologies HTML and CSS included. For more advanced styling and practice SCSS (Sass) could be introduced to get styling off the ground quicker for compiling. SCSS can be used in the application to create "custom variables" that hold colors, fonts, and background colors [16]. I have some experience in styling, and there can be opportunity to create a light and dark theme for the React page as an added accessibility feature for contrast [16]. This example is a popular 2020 feature in web development.

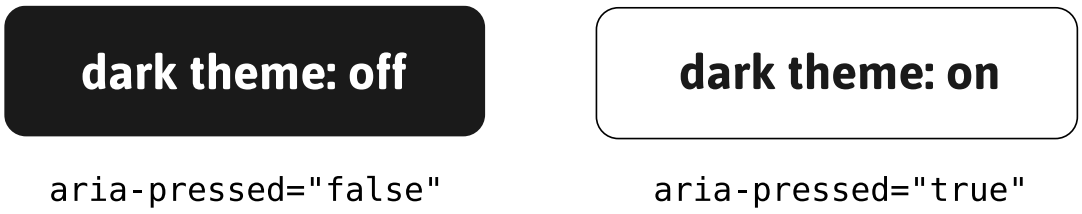



Fig.7. ARIA examples of pressed button attributes to change theme.

The documentation site will be created in React [5], and normally the language of choice in for this framework is JavaScript. With input forms and information, data needs somewhere to go and be stored. A database or "backend" will be implemented in the form of Firebase or Postgres. With regards to creating applications in React, I have never implemented a database area with a React project, so again there will be a learning curve, plus documentation needed to state this. I have been also been researching animation libraries

such as Framer Motion [17], which gives swooping style animations, cards, elements, and textures to pages. This library works with React and can be installed via NPM, a package installer. Towards the end of the project, I hope to add animations to the final site and hopefully provide final touches to what will be the complete product. A full technical stack is provided in the next section and will either shrink or increase over time.

## 6.0 Evaluation

With regards to evaluation techniques this section will be concentrated on testing practices. In the second semester there should be implementation of CI-CD (Continuous Integration / Continuous Development) to push the complexity of the project one step further. The web components will be unit tested using a JavaScript unit testing style framework. I must commence simple unit testing practices first on components to ensure tasks are running correctly. This area which leans into dev-ops territory will need to be researched before commitment is made. This is another area of the development industry altogether, but there can be potential to see more integration in the future. The aim evaluating accessible components would be some sort of autonomous system, a "pipeline" created that will have unit testing abilities. There is a cloud-based application called "Circle CI" [8] that let us achieve these future ambitions. For now, the concentration of the accessible components and the React site is what matters as a priority, to get things lifted off the ground first and foremost.

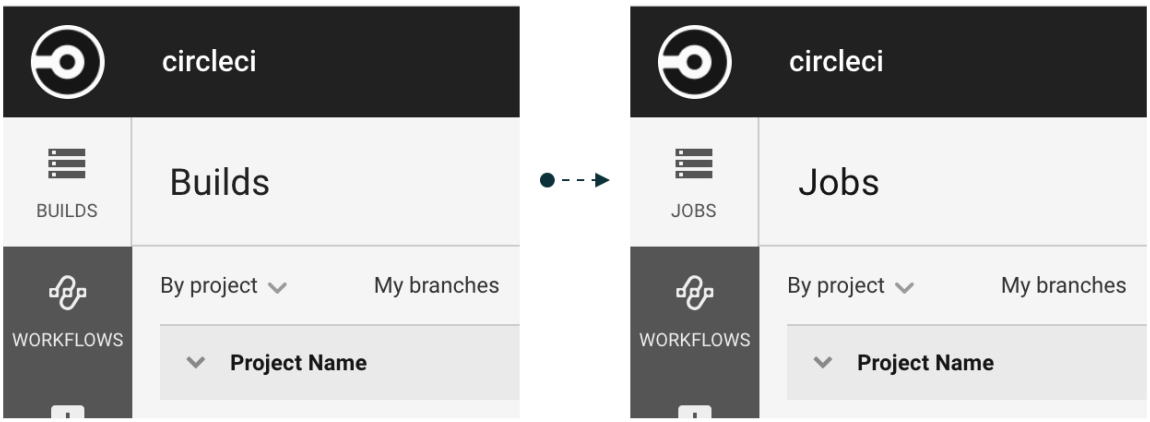



Fig.8. Circle CI User Interface

## 7.0 References Area

[1] R. Dahl, 'npm | build amazing things', *NPM | Build Amazing Things*, Dec. 22, 2020. https://www.npmjs.com/ (accessed Dec. 22, 2020).

[2] W. W. A. Initiative (WAI), 'WAI-ARIA Overview', *Web Accessibility Initiative (WAI)*, Dec. 22, 2020. https://www.w3.org/WAI/standards-guidelines/aria/ (accessed Dec. 22, 2020).

[3] W. W. A. Initiative (WAI), 'Introduction to Web Accessibility', *Web Accessibility Initiative (WAI)*, Dec. 22, 2020. https://www.w3.org/WAI/fundamentals/accessibility-intro/ (accessed Dec. 22, 2020).

[4] B. Ollendyke, 'Using custom elements - Web Components | MDN', *Using the lifecycle callbacks*, Dec. 22, 2020. https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements (accessed Dec. 22, 2020).

[5] J. Walke, 'React – A JavaScript library for building user interfaces', *React JS*, Dec. 22, 2020. https://reactjs.org/ (accessed Dec. 22, 2020).

[6] B. Holt, 'Typed JavaScript at Any Scale.', *TypeScript extends JavaScript by adding types.*, Dec. 22, 2020. https://www.typescriptlang.org/ (accessed Dec. 22, 2020).

[7] I. Team, 'Stencil', *Stencil is a toolchain for building reusable, scalable Design Systems. Generate small, blazing fast, and 100% standards based Web Components that run in every browser.*, Dec. 22, 2020. https://stenciljs.com/ (accessed Dec. 22, 2020).

[8] P. Bigger, 'Continuous Integration and Delivery - CircleCI', *Circle CI*, Dec. 22, 2020. https://circleci.com/ (accessed Dec. 22, 2020).

[9] B. Ollendyke, 'webcomponents.org', *Web Components*, Dec. 22, 2020. https://www.webcomponents.org/ (accessed Dec. 22, 2020).

[10] M. Kulkarni, 'Digital accessibility: Challenges and opportunities', *IIMB Manag. Rev.*, vol. 31, no. 1, pp. 91–98, Mar. 2019, doi: 10.1016/j.iimb.2018.05.009.

[11] M. Mays Espino, 'Website Accessibility for Persons with Disabilities: The Why & How', *Website Accessibility for Persons with Disabilities: The Why & How*, Dec. 22, 2020. https://www.americanbar.org/groups/business_law/publications/blt/2016/12/07_espino/ (accessed Dec. 22, 2020).

[12] C. Roland and J. Smith, 'WebAIM: Web Accessibility In Mind', *Web Accessibility Initiative (WAI)*, 22.12/20. https://webaim.org/ (accessed Dec. 22, 2020).

[13] M. Williams, 'Home - The A11Y Project', *The A11Y Project is a community-driven effort to make digital accessibility easier.*, Dec. 22, 2020. https://a11yproject.com/ (accessed Dec. 22, 2020).

[14] J. W. Hyatt, 'Polymer Project', *Polymer*, Dec. 22, 2020. https://www.polymer-project.org/ (accessed Dec. 22, 2020).

[15] A. Software, 'Jira | Issue & Project Tracking Software | Atlassian', *Jira By Agile Temas*, Dec. 22, 2020. https://www.atlassian.com/software/jira (accessed Dec. 22, 2020).

[16] S. B.-H. CPACC, 'Accessibility and Dark UI Themes', *Medium*, Sep. 16, 2019. https://uxdesign.cc/accessibility-and-dark-ui-themes-f01001339b65 (accessed Dec. 22, 2020).

[17] J. Dijk, 'Production-Ready Animation Library for React | Framer Motion', *When animating between two separate components, Framer Motion will take care of everything in between.*, Dec. 22, 2020. https://www.framer.com/motion/ (accessed Dec. 22, 2020).

**8.0 Proposed Tech Stack**


Fig.9. Project Proposal Technical Stack 08/11/20

## 9.0 Images & Illustrations Area

**Reference:**

All images, illustrations, colours, were created under Creative Commons Licence.

Adobe Suite and Canva Cloud Software were used to create logo and elements.

Fig.10. Project Proposal Logo Draft (08/11/20)

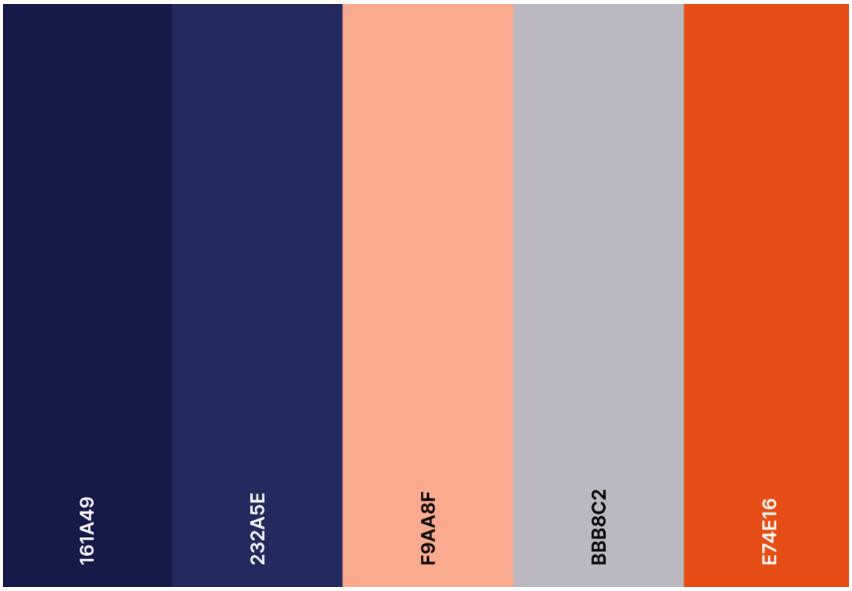



Fig.11. Project Proposal Colour Swatch (08/11/20)

Fig.12 Project Proposal Font Family (08/11/20)

https://fonts.google.com/specimen/Quicksand

https://fonts.google.com/specimen/DM+Sans

## 2.1. Ethics Approval Application (N/A)

## 5.2. Reflective Journals

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**October 2020 Document**
**Software Project**

So officially the first month of my final year in college has completed, and this is the first of reflective journals uploaded. Maybe I can look back on these documents to see how much optimism I had at one stage starting my final project, only to be in despair in another! There have been classes with Software Project, plus there has been documentation and rubrics given to find out exactly where will get the marks. *Frances Sheridan* has been a great lecturer in terms of delivering the content in a structured fashion, especially when it comes to the marking scheme particularly. A great document is the rubric to really showcase where you can gain and lose marks along the module process. This is very important to find out what my final grade will be in the process. One big focus on myself through October was to get the fundamentals of my idea off the ground with "Accessibility". Having an idea stuck in your head, going for it with the pitch, and looking at the rubric thinking *"Can I fulfil this to the greatest extent?"*. Hopefully, with some luck along the way, I can fulfil the project and push myself to develop a solid piece of software in the end. October for many people doing this module, is the mindset where you want to hope that your concept is "solid", something that can be contributable, and relevant. There could be some people deciding is this the right stream and right option, to be data analytics or a software developer? The best part about the industry is you have a passion for something in the here and now, and that could be very different in the future. Software development was my natural choice to make the idea I was sitting with happen, so I naturally went for it. I wanted to make a big commitment to myself to push myself into the deep end when it comes to use outside technologies with the project idea. To take tools, methods, and practices of not only from material presented in the course, but from my personal work also. This has been an important realisation with writing software and code in general. It is much better to throw yourself into the deep end, even if you do sometimes struggle. When the overall outcome of it pulls through, and you pieced the correct logic together, it turns out to be very fulfilling, and you can be ultimately proud of in the end.

So, with my accessibility project finally coming into realisation, I made and created various visual mind maps that will be in the final report, to find out areas of accessibility are most prominent going with my project pitch. I am not afraid to say I was using this document for help when pitching the idea on the monitor screen! These mind maps I have created are my information point past month, giving me reference ahead. Building on the project proposal that is due in a week's time, imagery can always come in handy.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**November 2020 Document**
**Software Project**

At the time of writing this piece there is a heavy workload of material due for each module in the semester. This combined with the ongoing research into web components, JavaScript frameworks and React, has allow me to work to a point where it must be encouraged to juggle tasks. Juggling tasks and module workloads is a key lesson to take from 4th year itself so far. Keeping on top things in the current home environment settings is a challenge, and it requires timetabling, charts, and even simplicity such as post it notes to ensure you do not lose your way. This from a reflective perspective is the toughest aspect of Software Project module, having the time and the ability, while not setting back other workloads. Gathering the required knowledge, to make the components logic happen and the elements of the project be in good shape for the mid-point presentation is the key focus. With the software project module, I have developed a master folder with everything combined ready to go for the upload, and it is a matter of conducting my own practical implementations with my chosen topics and then documenting extensively along the way. Currently I have two encapsulated pieces of information ready to be shown at the mid-point presentation in relation to screen readers and the React application is at the point where the information can be showcased from a high overview analysis. I managed to give good time in the research of Web Component syntax in JavaScript, and how to encapsulate and manipulate attributes which is one of the core ideas of my library of accessibility functions to come to fruition. Research on React and building applications was done by myself throughout the summer personally, but not on the scale that is needed for the requirement of the project. One major factor of the React application is that will be connected to a database to pass information and this month, I have conducted research into databases connected to the React library to go with my continuation of developing Web Components. Also, in this month the idea of testing these components to make sure they achieve the desired functionality will be important. Unit testing and integration testing practices are being researched along with a tool that could make my life easier creating web components. It is with this constant exploration there must be a deciding point which is the mid-point, to stick with the areas I will proceed in the second semester and not lose scope of the project.

Despite the busy workload of the semester, I have been enjoying the various technologies that modules in the current semester have to offer, some of which I have no experience and are brilliant additions to my portfolio of work. With these constant submissions, this has increased my confidence ahead to deliver a mid-point presentation that displays and shows a range of new technologies and tools while also implementing the practices that the college has provided with the course content.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**December 2020 Document**
**Software Project**

Now that final submissions for the last semester in other modules are nearly finished, there will be a chance to recuperate and reflect at the workload that was incredibly busy throughout the semester. In the area of the software project, I hope that the amount of theory and material I submitted in the midpoint will be sufficient to make sure there is good understanding of what I am attempting to do for the project.

Now that the groundwork for what I am trying to achieve is on paper, I can now implement all the tools and technologies I intended to use for software project, which there was not much time to do in the first semester. The midpoint submission showed and demoed an example of the screen reader web component has to offer of a button and a checkbox tag with encapsulated functionality. Along with this, the slides and presentation were meant to show a pathway to where I am with tasks and what needs to be done with the Gannt chart, which there is a lot to do technically. I laid out a plan that hopefully will make logical sense to plan out each stage of development step by step, while same time also increasing the level of difficulty and level of integration each month to satisfy the challenging aspects of the module. An ambition was made for future continuous integration and testing of the web components I am currently building to be hopefully integrated into a JavaScript framework. This was the core aim of the midpoint submission. For myself I stated in the midpoint questions that documentation was the toughest content to cover over practical implementation and I stand by that statement. Due to the pressure of time constraints with other modules, it was hard at this moment in time to get a full picture of types of requirements and use cases you will need for the final version of the project. Thankfully, all of this can be adjusted or changed in the future to benefit later, but for the area of the midpoint submission and getting a vision I wanted, it was tough to write down what I needed on paper, while only having a demo or short example that was created or implemented in one area. With the midpoint submission now over the groundwork with the planning and the Gannt chart and the requirements has been made, now is the best time in the next few weeks to really get the project moving and pushing forward; to a point where most students in the class want to be and that is to be comfortable! Here is hoping that the realisation and the idea of being in full control of the project will become a reality in the coming months developing it.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**January 2020 Document**
**Software Project**

The past few weeks of project submissions were some of the toughest college schedules I have ever done. The first week of January was a surge to hand in as many submissions as possible back-to-back. Now that the exam period is finished and finalised there is the prospect of recuperating and awaiting results; that will determine where my mark will be leading into the final semester. This is important and in relation to the software project I am very much looking forward to getting a result to determine where my standing is; and what I can aim to improve for the future. Overall, I felt happy with my mid-point submission having enough information included to ensure what I was doing look promising ahead. In the month of January there was moments where I adjusted the software project Gannt Chart and planned my upcoming development more strategically. In the time of January, I made sure to dedicate a week or two not only for myself, but also for upskilling with the project. There was material in JavaScript as a refresher that I returned to which increased my confidence in the language. JavaScript is my primary language for the project and was an area where I made great strides in upskilling the previous summer in 2020. Along with JavaScript there was tutorial work given into the React library, a tool which I have introduced for the final year project. It is with these incremental developments, that bigger events can proceed in the future. In the final week of January college recommenced to a fresh start and brand-new modules to discover and enjoy.

Overall, I have felt lucky to the project and the certain type of attributes it holds; I refer to technical areas such as MVC frameworks and libraries for example. There is a good reason for this actually! The reason why is because the new semester module of *Cloud Application Development* happens to be a MVC framework architecture on based in Ruby. This was in fact great news and indications for my project development; the technical concepts were similar upskilling material with the final year project and the React library. The first week back also included area that will be discussed with weekly seminars in the software project class on the Monday. Topics such as “unit testing” and “frameworks” that are two predominant areas my project which I will really look forward to discovering more from. It is with these instances that I find myself having an indicator of luck going forward. If I happen to stick with the modifications made in the Gannt chart, I can push forward with project with optimism knowing I can get everything and more completed!

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**February 2020 Document**
**Software Project**

Development with the project is now in full swing and the schedule and timetable that has been designated for my classes is more than generous. And to make sure I have plenty of time to work on the project. This month there was information given on the website that will be launched to co-inside with the release of the project submissions. This is quite exciting as it allows employers to check out and discover our many months of hard work. While exciting it is also at the same time terrifying to know that whatever information is being handed into the notebook area in Microsoft Teams must stick in the future and in the month of May when the showcase begins. Which means that execution and delivery of the project will need to be as close to what I have written in this month of February! It will be nice to look back on this document and find out whether I delivered on the promises or not! The weeks of seminars with talks on frameworks, cybersecurity, and unit testing have been fantastic nuggets of information that are more importantly relevant to the future of the project. It is great to discover and check out material that is as intricate and complex; but luckily enough by the lectures of NCI to be explained in high level detail.

I look forward ahead to the upcoming talks every Monday in between my development of the software project to ensure that I am being kept up to date; attending class every Monday. Currently with the software project there is technical areas of web components, frameworks, testing and concepts like “continuous integration” in full swing. Continuous Integration is the latest upskilling I am trying to achieve to create an extra dynamic for my project. This information was all documented and included in the Gannt chart to ensure I have vision going forward in what will work and what will not work. Currently my development is all purely software so writing code and building is the dominant factor right now. This will be the same for several weeks. In-between this building I am thinking forward to return to my master document created from the mid-point; and to make sure whatever material is being built, be documented carefully. I know that from a marking perspective this will play an important factor, and not just the technical aspects of the project. It is all about the result, isn’t it? For now, it is truly head down (abide by my Gannt chart!) and get on with the workload.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**March 2020 Document**
**Software Project**

The first submissions of the year were submitted in this month along with the realisation of the last submissions to the college. The past few months has been surreal with the learning experience being remote and everyone who has switched to Streams. From this situation, it has been admittedly mentally tough to get a grid of things at times during the final year. There was times and moments where I felt honestly low, dipping points perhaps from the job-hunting process. There has been a lookout for roles and opportunities, and to further my career after college. Like everyone who is maybe in the same position I am, it has been hard to keep up the motivation to look for jobs and to go through the process of multiple rounds over and over for employers. This month it particularly came to the point where the applications needed to stop and the re-emphasis for finishing the degree needed to begin. The most important thing is to definitely keep on going with the end of the degree and the end this educational chapter of my life; to keep on pushing with the required uploads at the end of the semester and let us see what happens on the working end for the summer.

In this month, the final OneNote uploads with information were finalised along with talk for Software Project Showcase at the end of year. I am currently apprehensive about this and hope to have a decent project for it by the end of the year, otherwise it will be a disaster! The software development concentration into workload took a downturn recently due to the influx of requirements that was needed for the other modules and other submissions. In classic form have crept up out of nowhere! Things were spaced out far better for this semester, and there has been better control; but the journey to the finish line has felt a lot longer than I expected. Despite this, the state of the software project remains solid and at a point where I honestly cannot wait to return to it once the other submissions that have come out of nowhere, can finally go into the Moodle submission folder.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**April 2020 Document**
**Software Project**

With the software project everything has been fully finalised with due dates and submissions on the college's end was delayed. This delay and reposition of dates have arrived with good fortune admittedly, as now I can finalise everything that I need to ensure that my software project is the best position it can be come May 16th. There were recent talks to for the Graduate Showcase where there will be a series of prizes and events to be given out for students for the year. It is great to see the college hold the event for all of the 4th year students at the end year as this class and the maybe previous graduates of 2020 have prob been the most resilient individuals in the face of this new world; and since the world has entered lockdown.

At the time of writing this every submission for every other project along with exams have been submitted; so now it is all about uploading the software project and it is all about working tooth and nail to carve out the best result in the biggest module of the year. Hopefully! Despite some ups and downs with working and content I have an integrated CI/CD pipeline happening for my components and my React application. I have multiple technologies being integrated and being blended creating and bringing a nice workflow, which is something I been in and out for the past few weeks now. And most importantly I have two free to purely concentrate on the submission when there are no other submissions ahead. And to be honest that feeling is pretty fantastic. To have pure concentration on something that I have a current passion for is a nice thing to have. Before I will know it, it will be all over, and the degree will come to an end. But at this stage there is no way I am applying for a postgraduate role. I need a holiday. And a vacation. When is the world going to open back up again?! Either way I aim to have fun and build out the rest of this cool idea and software project.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**June 2020 Document**
**Software Project**

Sadly, throughout the summer months the continuation of the final project has occurred due to sheer burnout. In the late stage I applied to go forward with an extension to August to submit my final project. I was not happy with the direction and the condition of the material coming ahead with the May submission, so I took it upon myself to message and inform Francis and Paul my supervisors to extend my current deadline of submission.

The scale of the project was so ambitious at times I managed to run into a brick wall integrating different technologies that lie across the two areas of the project. These technologies all heavily integrate into the core aspect of my application. These two areas are the React application that has a lot of integrations running at the same time with testing emphasis in particular. And then on the other end is the pipeline of accessibility web components. With ARIA style HTML elements in the DOM there are many attributes, so it was a question of choice to go with the most viable and logical choices. This needed time and better scaling simply put. So, I took it upon myself, to find the time and get the extension to have the project go into a better direction; especially with these accessible components and having them link into my main React application. The extra time was warranted and needed; it gave me better results on the web components elements of the project. I am thankful for the college for allowing the extension in these tough and difficult times of studying and learning remotely. It is with these reflective journals I can read back and see I made the right decision in time in taking measures needed for a decent final project submission.

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**June 2020 Document**
**Software Project**

After extended research I have come to the realisation that on sheer scale it would not be possible to deliver the amount of types of logical components that interact within a JavaScript accessibility library. However, there is opportunity to have the project be envisaged into a design system for accessibility; on a different scale and that is the direction I am taking the project into the months of June and July. Throughout these months there was research made to take the components I have already built and turn them into a design library, and this is where I have managed to come across the tool of **Storybook JS** to essentially created a documentation and design library system of accessible components so that people can view these online. With the project showcase profile and criteria, I had written, this thankfully fell into my original concept and is still a solid implementation of the work I was attempting to achieve, especially with the idea of focused accessibility around components. It was also exciting to work with yet again another tool I was not familiar with and this time around I had the time to insure I was using it to the best capacity.

**Storybook JS** and a repository were setup and then tested in the month of June to then see can an accessible design library system be possible. Especially with the level of JavaScript I currently possess, I decided to drop **Stencil JS** (which was on my original proposal for component building) and use Storybook JS instead. On the other end the React application was always in good condition and has actually been getting better and better throughout the summer months. **React Router** was implemented for state management and bundler tools were introduced allowing for a better productive application that is on the scale of a production-based build. More on this will be in the next reflective piece!

**Matthew Byrne**
**X17138744**
**Reflective Journal**
**July 2020 Document**
**Software Project**

In the week of July, I am full steam ahead with the new testing based in Storybook JS for the components building. The React application is being finessed to the best possible shape it can be, and adjustments are being made to focus on certain areas of the rubric which will be explained in the presentation. I created my final presentation slides and to ensure I have a professional delivery that is informative to the workload I have been doing. At the same time during this month, I have managed to find work in the industry as Junior Developer so now there is a moment in my time where I am tackling multiple projects at the same time, whereas be in a work-based environment and now with finally collecting my degree with the National College of Ireland. It is truly exciting times for myself learning all new fresh concepts and aspects of web application development.

Despite not being able to logically put together the original idea for my accessible components I am really happy that I can manage to still deliver a design system with a great record of the code I have written throughout my time building the project. In July and in the last week there the exciting moments of finally delivering submissions and with this, I was glad to divulge in all the original backstory as to how I ended up with the project in its final condition which was really satisfying for myself. With the Git repository submissions, I also attached my original code at the time I was developing with **Stencil JS** and now with the new tool of **Storybook JS,** managed to deliver a solid idea or a great foundation of what a real professional accessibility library could be like to users in the future. After submitting this final piece and attempting to record my final video, I am looking forward to receiving my degree later in the year for all of my hard work and accomplishments.

## 5.3. Other materials used (Solo Sprints, GitHub Repos)


# Solo Sprint

Project Owner: **Matthew Byrne**

Date of Cycle: **Sample**

**Evaluation & Breakdown**
This paragraph will evaluate previous sprint cycles development against what is currently available. What I have done, what I have worked on and what I intent to do for the next week forward in the project.

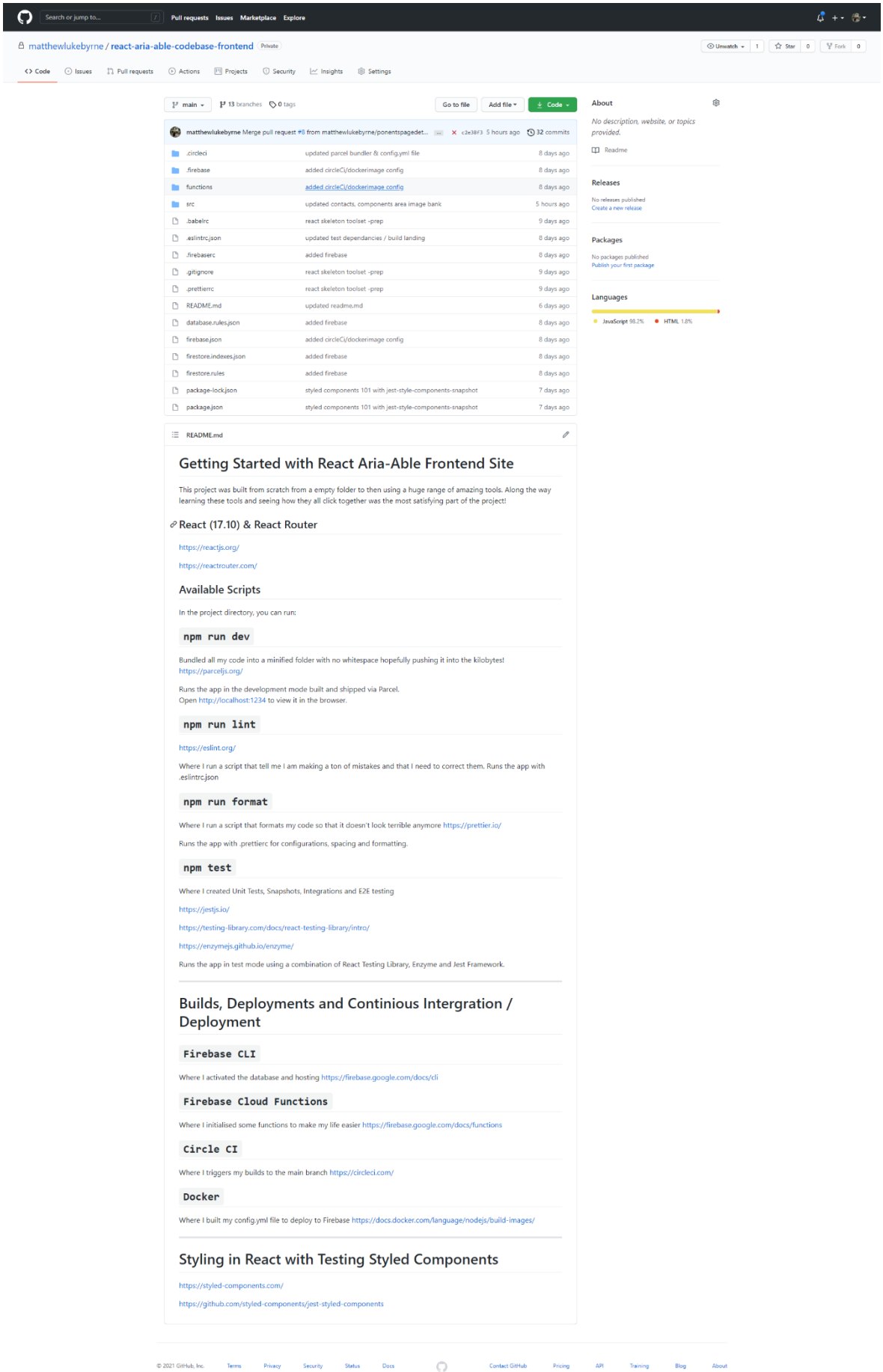
**Product Backlog**
The section will list the work that needs to be completed on the project with the current Jira roadmap completion image down to track what stages the project and other tasks are currently.

**Test Driven Development (TDD)**
Once a component is completed and has functionality, this will be the area that will record my component testing evaluation (RED GREEN REFACTOR), have all tests passed on a certain component and can they be refactored to be improved.

**Selection Phase**
This area states the rationale for choosing the selected components/ tasks from the complete backlog list in Jira to be the targeted, and to be completed for the current sprint

**Solo Sprint Tasks**

1. **Web Component Research Mind Map**
2. **Unit Testing Research Mind Map**
3. **Web Component Sample on GitHub (VS Code Demo User-Card)**


Search or jump to...
Pull requests
Issues
Marketplace
Explore
matthewlukebyrne / react-aria-able-codebase-frontend
Private
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
main
13 branches
0 tags
Go to file
Add file
Code
matthewlukebyrne Merge pull request #8 from matthewlukebyrne/ponentspagedet... c2e38f3 5 hours ago 32 commits
.circleci updated parcel bundler & config.yml file 8 days ago
.firebase added circleCi/dockerimage config 8 days ago
functions added circleCi/dockerimage config 8 days ago
src updated contacts, components area image bank 5 hours ago
.babelrc react skeleton toolset -prep 9 days ago
.eslintrc.json updated test dependancies / build landing 8 days ago
.firebaserc added firebase 8 days ago
.gitignore react skeleton toolset -prep 9 days ago
.prettierrc react skeleton toolset -prep 9 days ago
README.md updated readme.md 6 days ago
database.rules.json added firebase 8 days ago
firebase.json added circleCi/dockerimage config 8 days ago
firestore.indexes.json added firebase 8 days ago
firestore.rules added firebase 8 days ago
package-lock.json styled components 101 with jest-style-components-snapshot 7 days ago
package.json styled components 101 with jest-style-components-snapshot 7 days ago
About
No description, website, or topics provided.
Readme
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
JavaScript 98.2%
HTML 1.8%
README.md
Getting Started with React Aria-Able Frontend Site
This project was built from scratch from a empty folder to then using a huge range of amazing tools. Along the way learning these tools and seeing how they all click together was the most satisfying part of the project!
React (17.10) & React Router
https://reactjs.org/
https://reactrouter.com/
Available Scripts
In the project directory, you can run:
npm run dev
Bundled all my code into a minified folder with no whitespace hopefully pushing it into the kilobytes!
https://parceljs.org/
Runs the app in the development mode built and shipped via Parcel.
Open http://localhost:1234 to view it in the browser.
npm run lint
https://eslint.org/
Where I run a script that tell me I am making a ton of mistakes and that I need to correct them. Runs the app with .eslintrc.json
npm run format
Where I run a script that formats my code so that it doesn't look terrible anymore https://prettier.io/
Runs the app with .prettierrc for configurations, spacing and formatting.
npm test
Where I created Unit Tests, Snapshots, Integrations and E2E testing
https://jestjs.io/
https://testing-library.com/docs/react-testing-library/intro/
https://enzymejs.github.io/enzyme/
Runs the app in test mode using a combination of React Testing Library, Enzyme and Jest Framework.
Builds, Deployments and Continious Intergration / Deployment
Firebase CLI
Where I activated the database and hosting https://firebase.google.com/docs/cli
Firebase Cloud Functions
Where I initialised some functions to make my life easier https://firebase.google.com/docs/functions
Circle CI
Where I triggers my builds to the main branch https://circleci.com/
Docker
Where I built my config.yml file to deploy to Firebase https://docs.docker.com/language/nodejs/build-images/
Styling in React with Testing Styled Components
https://styled-components.com/
https://github.com/styled-components/jest-styled-components
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

matthewlukebyrne / storybook-aria-able-ponents

Private

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

main | 9 branches | 0 tags | Go to file | Add file | Code

matthewlukebyrne Update README.md 47407c1 yesterday 30 commits

| | | |
|---|---|---|
| .storybook | Merge branch 'main' into storybook/contexts.js | yesterday |
| imgBank | added some pics | yesterday |
| public | add tab, required, search orientation, value | yesterday |
| src | Merge pull request #12 from matthewlukebyrne/storybook/updates | yesterday |
| .gitignore | Initialize project using Create React App | 5 days ago |
| README.md | Update README.md | yesterday |
| package-lock.json | problems storybook 6 | yesterday |
| package.json | Merge branch 'main' into storybook/contexts.js | yesterday |
| yarn.lock | Merge branch 'main' into storybook/contexts.js | yesterday |

About

Final Project work for College with React, Storybook and more!

react | javascript | styled-components | jest | storybook

Readme

Releases

No releases published

Create a new release

Packages

No packages published

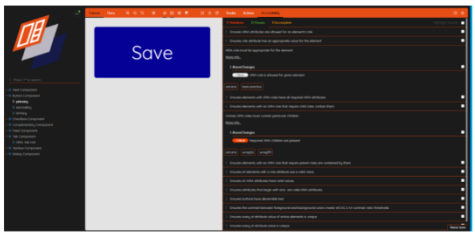
Publish your first package

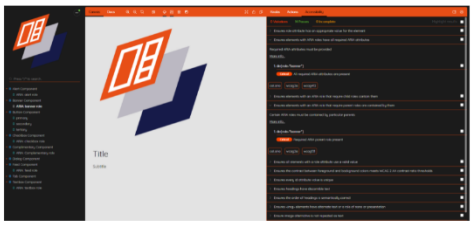
Languages

JavaScript 96.8% HTML 3.2%

README.md

# Getting Started with Create React App & Storybook JS components library




https://storybook.js.org/

Version 5.13.21 (Updated from Version 6)

## Available Scripts in Storybook JS

## Available Scripts in React App from 'create-react-scripts'

In the project directory, you can run:

`yarn start`

Runs the app in the development mode.
Open http://localhost:3000 to view it in the browser.

The page will reload if you make edits.
You will also see any lint errors in the console.

`yarn test`

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

`yarn build`

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

### Component Building using a series of tools and processes with information listed below

Styled Components -

https://styled-components.com/

Utilising tagged template literals (a recent addition to JavaScript) and the power of CSS, styled-components allows you to write actual CSS code to style your components. It also removes the mapping between components and styles – using components as a low-level styling construct could not be easier!

Polished.js -

A lightweight toolset for writing styles in JavaScript.

https://polished.js.org/

https://polished.js.org/docs/

Styled Components Modifiers

Styled Components are incredibly useful when building an application, but the community lacks guidelines and best practices for how to structure, organize, and modify a component library. Fortunately, the CSS ecosystem has several solutions for this, including the very well-thought-out Block, Element, Modifier (BEM) conventions.

https://www.npmjs.com/package/styled-components-modifiers

MDX -

MDX is an authorable format that lets you seamlessly write JSX in your Markdown documents. You can import components, such as interactive charts or alerts, and embed them within your content. This makes writing long-form content with components a blast.

https://mdxjs.com/

### Deployment Strategy for Web Components via Netlify Hosting

An intuitive Git-based workflow and powerful serverless platform to build, deploy, and collaborate on web apps.

https://www.netlify.com/