Noroff
School of technology and digital media

Exam Report

Project Exam 2

Sara Sørensen

Word count

Summary: 51 | Main text: 5059

https://sara-sorensen-holidaze.netlify.app/

Table of Contents

Summary

I have made a hotel booking website called Holidaze. To create this website, I chose to use React and SASS for coding and styling.

For debugging purposes I used Chrome DevTools, where I checked my console log during implementation. I also used Lighthouse, for improving the quality of the web site.

Body

Introduction

For my exam project, I was tasked with making a hotel booking website for a company called Holidaze. For this exam, my goal was to take on an extensive project where the end result should reflect my skills both visual and technical.

Planning

Gantt chart

For the first weeks, I was working on my Gantt chart (see Figure 1.0), style guide and my prototype. I started with creating the Gantt chart, as this would give me a good overview as to what I needed to do for the upcoming weeks.

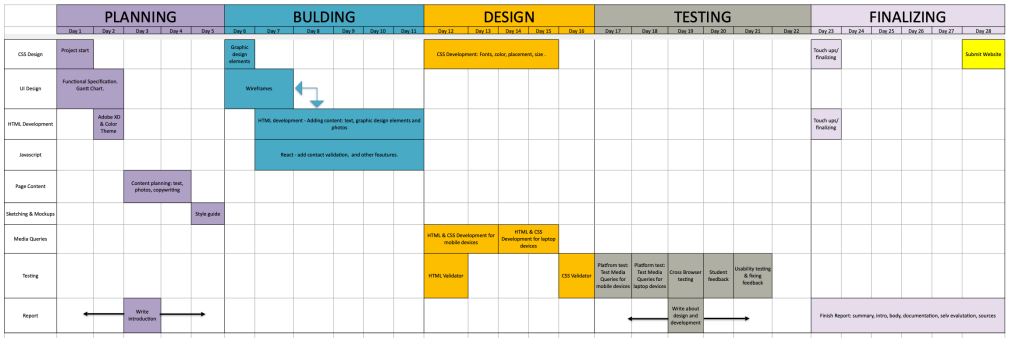


Figure 1.0

After creating the Gantt chart I saved it in my iCloud[1], and planned on saving everything within the project in that folder. Saving everything in my iCloud, would ensure that if anything where to happen to my computer, all my hard work wouldn't be lost.

Before I started planning my layout and design, I wrote a simple list of who I envisioned would visit this site.

The target audience:

-Over 18 and up (as most people traveling abroad is of legal age).

-Knows English.

-Has different financial situations.

-Different types of personalities - backpackers, families and older couples.

As you can see this website needs to take into consideration, that many different types of people will visit this website. This is very helpful, when creating a color scheme.

When creating a style guide, I created the color scheme (see figure 1.1). I went with neutral colors, as I wanted the website to cater to many different types of people.

According to Wikipedia[2], the color blue is associated with calmness and yellow with warmth and energy. These are therefor my two main colors for contrast - and they go well with the target audience.

My other colors are white, beige, grey and black. These colors are mainly for backgrounds and text, to ensure that everything is easy to see and read.

Figure 1.1

Design Elements

After planning out my design and layout using the style guide and my prototype, I knew roughly what design elements I needed to make. I started with making two logos (see Figure 1.2)

My main logo has the word Holidaze and a map icon as the letter O. This was to make an association with traveling and maps, the idea was to put Holidaze on the map sort of speak. Knowing that I used the map icon in my logo, I wanted to utilize that as my smaller logo for other placements but for a simpler look.

Figure 1.2

I went on to choose my fonts, and went to Google Fonts[3]. I landed on some fonts I thought were easy to read, in addition to interesting and contrasting to each other (see figure 1.3). For my main heading (h1), I went for Arial. I chose this font because it's easy to read, and my main headings are informative - therefor it is imperative that users easily understands what it says.

To bring some contrasting font to my website, I chose to import a font called Kalam. It pairs well with the main headings and the body text, and helps break up some content as my accent font. The body text is the same as the heading, as this text should be clean and legible, key words being readable and user-friendly.

My font hierarchy goes as follow:

1. Main heading - Arial Sans Serif.

2. Sub Heading - Arial Sans Serif or Kalam.

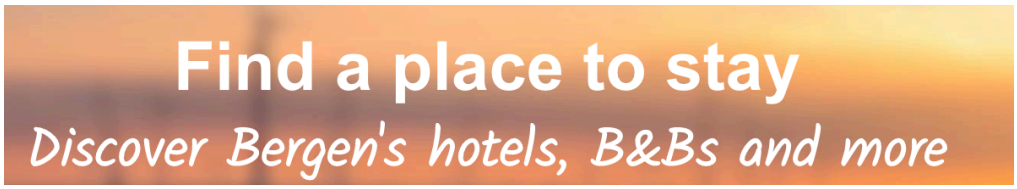


Figure 1.3

Graphic Elements

For other graphic elements, I installed Bootstrap Icons[4]. I went with this library as they are high quality, whilst not taking up too much loading time. For example I wanted to use these for linking social media, displaying informative text about hotels, using them on forms and on the administration page.

I find using icons to be a great way to draw attention to something, and that can be helpful for users to find important content. They can also be a universal language, where they help people in an informative matter without needing much context.

For the header I chose a picture using Pixabay[5], where I can find free pictures with no copyright issues. I found a picture with nature and windmills, and thought this would go perfectly on the front page. I believe many people associate Norway with nature and windmills, so I went for this picture.

I did edit the photo using Illustrator, where I blurred it. I wanted it to be almost completely blurred, so that you could just make out what the picture is (see Figure 1.4). This was to ensure that any content I would put on top, would be visible and readable. I made three versions of the header - one for big screens, one for mobile screens and landscape orientation on mobile screens.

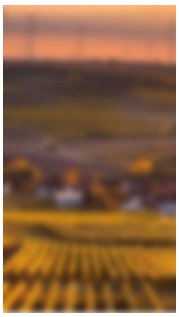


Figure 1.4

Initially in my style guide, I created a yellow and grey button. Upon coding this, I found that the yellow buttons did not score well on accessibility. Therefor I only have grey or red buttons as graphic elements on my website, as I want to meet the WCAG guidelines. The grey button functions as my main button, and the red button is an action button, such as delete hotel, followed by a confirm action. When you hover the buttons, the opacity on the background color goes down.

In my SASS I created the button as a mixin, that way I could easily use the style on buttons, whilst being able to add new classes to the buttons if I needed to change placement. This helped me work in an efficient matter, and also minimized my css as I didn't have to repeat styles for different buttons.

SASS

For styling my website I chose SASS and Bootstrap, and used BEM conventions when naming my classes. The main reason I like using SASS is because it allows me to create variables and mixins. The mixins in particular are great for reducing CSS size, as I don't have to repeat code. I made mixins for flex boxes, transition, links and buttons (See figure 1.5).

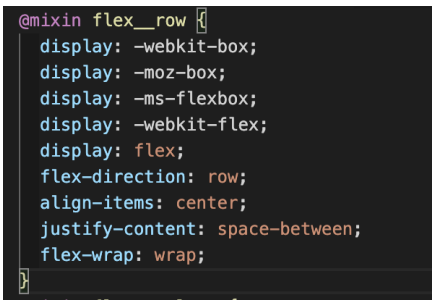


Figure 1.5

Building

Wireframes

I started with wireframes, which consists of setting up my layout. I took my time setting up my folder structure, as I knew going in to this I would be splitting up my code for an easier building and debugging process. I proceeded with adding a navigation, and installing a router

(react-router-dom). I sat up my navigation, and added toggle and styling using Bootstrap. I created a header, body and the footer as I had planned in my prototype.

For my different pages I used flex box properties, to ensure the content is responsive and works on different screens. I usually went with a full width and an automatic margin and or padding provided by imported containers, rows and cols from the bootstrap libraries. I used percent, vh, em and rem when sizing the body and content, for a better responsive design.

Where there is a lot of text, I made sure the with is no longer than 50-60%, which is an ideal range for reading on bigger screens. For an example, I did this on informative text on the admin text warning the administrator about deletion.

My page has a meta viewport, unique description, keywords and a title to generate results from search engines. In the description I opted for a call-to-action, such as discover more today! The key words include everything the page has to offer: bergen, hotels, booking etc. This helps to build an effective SEO content strategy, to convert traffic from Google searches to my website.

I made sure the code was neat and semantic, ensuring the headings provided a hierarchy and clarity to the page following the WCAG guidelines and providing alternative text and screen readers only text (for an example the loader or icons). This ensures that people with screen readers has a good user experience.

Navigation and footer

Above the header I have the navigation in a simple beige color with the logo and the links. The links match the color of the footer, which is grey. When you hover the links, they match the logo with a nice transition effect. I made the transition as a mixin, and include it where I find it necessary. The logo links to the front page, for better navigation.

When the navigation is on smaller screens, the toggle button is on the right. This is placed to the right, because most users are right handed and uses their right thumb to click. This contributes to a simpler user experience, and makes the page more operable meeting the WCAG principles.

In the footer I add the smaller logo, which also links to the front page. In my opinion the logo contrasts well to the dark background. I added contact information, copyright info and an additional navigation so that the user doesn't have to scroll to the top navigation if they want to navigate.

Home Page

On the home page I have two headings informing what this page is all about and a search input, this is placed on top of the header. If you write in the input, an auto dropdown shows up displaying hotels matching the input.

To import the dropdown, I utilize a lazy import. I did this because this ensures the dropdown is not loaded, before it is needed. This helps me save loading time and helped me score higher on Performance, in Lighthouse. The dropdown is wrapped with a suspense component, which has a fallback loader if the content is not finished loading.

In the Home drop down file, I fetch the hotels. This fetch checks the response status, and loads the content. If the content is not loaded, it displays and console logs the error.

For filtering the input result, I use an include method. This determines if the input value includes a value from the array. I use this method so that it's not expected from the user to know a hotel name, but to just search a letter and get some result, to ensure a better search engine. In the same function, I change the value of the search to true, which has an on Change and triggers the dropdown to be displayed.

Upon using the dropdown, I realized a flaw. I couldn't close the dropdown if I clicked outside of it. I did some research on how to handle this, and found a question from StackOverFlow[6]. I did get some inspiration from this article, but made it fit to my code. I fixed the potential clicking outside of the dropdown by adding an event listener, that listens for a mouse down click and removes it when it's handled.

The dropdown displays the hotels including the hotel image and name, that you can click on and it would take you to the specific hotel. In styling I added overflow - scroll, so that all the hotels would be displayed but you could scroll to them. This allowed for the user to see all the hotels, but the dropdown would not take up additional space on the page.

Upon finishing my website, I had friends and family go through the page and give me some feedback/ questions. One comment I got was that I should give users some message, if their search input didn't match any hotels.

I realized this was a flaw on my part, and that this would be essential to avoid any misunderstandings. I added an if statement to my code, where it checks if the filtered array of hotels comes back undefined or empty. If it does, a message is displayed saying it doesn't match anything and that the user needs to type something else.

I didn't include the boxes that I initially had in my prototype, to my home page. I opted not to include them, because they had a major effect on performance. It seemed redundant to add them just for repeating the links, that is already available in the navigation and footer. As I want a strong landing page for my website, I found it necessary not to include them. Research show that users are more likely to click out of your page, if loading time is too long.

Hotels Page

On this page I fetch all the hotels, and display them by importing a Hotel card component. Whilst I wait for the fetch, a spinning loader is displayed to avoid any misunderstandings

(this loader is displayed on every page that includes a fetch). I render the hotels using a map function, and they are displayed in responsive columns.

I decided to add the search component at this page too, just to allow for some specific searches. This has the same filtering function as the home page, and also an if statement to handle if the search comes up empty.

The hotels are displayed in responsive columns, in three's on larger screens, and goes down to two and one on smaller screens. I utilize the bootstrap library, and style the hotels as cards.

The hotel name on the card, is in the color blue, and has the font Kalam. I think this adds some personalty, and helps to break up the text functioning as a contrast. The button is in the dark grey color, creating a clean and modern look. If you click the button, it takes you to the specific hotel, using a link from react router Dom and the id of the hotel.

Hotel specific page

On this page I fetch the specific hotel in question using their id. I get the id using use Params, which returns the id for me. On this page you get to see a bigger version of the hotel image, the name, how many guests it can have, price per night, some description, email and some icons displaying wifi, central location etc.

The icons are mainly decorative, but in my opinion it grabs your attention and can work as a persuasive technique when a user is deciding on wether or not to book the hotel.

I had planned for this page to be laid out as a two column, with the picture on the left as a circle. I displayed the hotel with two columns initially, but it didn't work out as I'd envisioned. The picture had to be much smaller, than it would be if I had it on top. I think it's important as you design a page, to be open to changes. The two column layout was a good idea, but execution said other wise. I thought it was more important for the user to actually see the picture, then it was to keep the original idea.

The text is placed in a column with a white background, this was to ensure readability. The text is aligned to the left, as this is recommended when displaying longer informational text. Starting from the left, is also the most common reading direction, so I opted for this to cater to the majority of people and fit the target audience.

In my code you can see that I set the hotel image and name, to the local storage. I did this because I wanted to display the hotel in question on the enquiry page, and this was the most efficient way to do so in my opinion. If you click the book hotel button, it takes you to the enquiry page with the hotel id using router dom.

Enquiry

When you land on this page, you see a two column layout. To the left I display the hotel image and name from the local storage, as well as some informational text. To the right is the enquiry form, where I import the main heading. Below is the inputs such as name, email, check in and check out. On the bottom of the form I display the hotel id, as a read only using the use params method.

I installed a few libraries to help me validate the form. I use Yup and React hook form, react hook helps me validate the form and requires less code compared to what I would have to write my self. Form validation rules are defined with the Yup schema validation and is passed to the hook form (use form function).
The use form function returns an object including registering input, handling submission, displaying errors as well as the Yup schema as mentioned above.

If the form is submitted, it firsts checks if the inputs are valid by handle submit, and if it's valid, the on submit is called. The inputs are valid if you have submitted a name, a valid email, and the check in must be from today and cannot be an older date, as well as the check out date must be after the check in. This was because I did not want a user to be able to select a date of the past, and select a wrong check out date.

After this, I use JSON stringify on the input value and post this to enquiries using fetch. After this I call a function called handle show, this displays a modal after the form is submitted.

The modal is styled using Bootstrap libraries, and displays a message to the user saying thank you and how Holidaze will get back to them. When this modal is exited, the enquiry page is reloaded, clearing the form.

Initially I had the user being redirected to the admin page, but this was changed after some feedback from another student. She thought it was weird to be redirected to the admin page, as a customer, and I agreed. Adding the modal was an easy fix to this feedback, and it felt more professional. The additional message in the modal to the user, ensures good communication and helps avoid any misunderstandings.

In the fetch enquiries file the enquiries are fetched and displayed in the admin page. I created an if statement handing if the enquiries came back empty. I made this into a constant using an arrow function that takes in two parameters - the enquiries and a fallback. This list is returned to be displayed, displaying either the enquiries or the fallback. The fallback displays a message saying it can't fetch enquiries, try to refresh as refreshing the page can help with this in some cases.

The enquiries are displayed with all the input information, as well as two buttons - delete and reply. If you click reply, this has the enquiry email set as mail to, so you will automatically get a new email with their address and you can contact them.

If you click the delete, you have to confirm this. I installed react confirm alert, which is displayed when you click the button. If you click confirm, this calls an async function named delete enquiry, which gets the id of said enquiry and has a method of delete and updates the enquiries by fetching them. After this the page reloads, and the administrator can see the changes.

Having a confirm deletion was vital for avoiding any mistakes. If I didn't have the confirm alert, it would be very easy for me do delete something without noticing, thus possibly creating a conflict between Holidaze and it's customers.

For displaying the check in and check out date, I installed React Moment. This library allows for easy formatting of dates, as the default format looked busy in my opinion.

Contact Page

The contact page has the same layout as any other page on my website that has a form. To the left I display an icon and a message. In the form I include icons as I do in the other forms, functioning as visual information.

The contact page has the same form and set up as my enquiry page, with the modal at the end of submitting the form. The messages are posted and fetched at the admin page, with a fallback if the content is not properly fetched. Just as the enquiry page, the contacts has reply and delete buttons.

Authentication

Register

The register form has the same set up as the log in form, linking to log in, in the left column if you already have an account. You have to create an email as username, and a password. When you click to register it sets the username and password to local storage, and redirects you to log in. The redirection is done by use History, and pushes you back to login.

If you click the eye icon in the password input, you can toggle between seeing the password text and not. I thought this would be a helpful feature, for people typing their password.

Log in

When you click on the log in in the nav link, it takes you directly to a log in form. The log in form has the same set up as previously explained, where the validation for the password is what's different. The password requires at least a number, and must be 8 characters or more.

In the left column there is an icon of a person, and a link to where you can register a new user if you'd like to.

If you try to log in without having a registered user, this is handled in the on submit of the form. It checks to see if username and password exists in the local storage, and if it doesn't a modal is displayed. The modal displayed says something went wrong, and to check your

input. It also informs you that if you don't have an account, you can register by clicking on a link.

The on submit gets the username and password from local storage, and if your input matches your registered user information, it displays a modal saying log in successful. When you close the modal it uses history to push you to admin and I use history go to refresh the page. I do the refreshing as this was necessary to be able to display the new links in the navigation.

In my layout file, I check for a user from the auth context file. If there is a user, the admin navigation link is activated as well as a log out link. This allows for the administrator to navigate between pages and accessing admin, and logging out whenever. I decided to add this feature to the footer, for additional navigation.

In the auth context file I take usage of create context. This allows for me to share the user data, and to be able to authenticate the user. In here I check for an existing user, registering a new user and logging out a user by removing their info from local storage.

In the log out file I import the confirm alert, as I wanted the user to confirm that they want to in fact log out. In addition to this, I wanted to inform that if you log out, you will gave to register again to access admin. I added this because of feedback I got from the people I had test my page, as they found it confusing that they couldn't log in with the same information again, after logging out.

I import the log out function from my auth context file, and after you confirm logging out, do log out is invoked and uses history push to redirect the user to the home page. Initially I had them redirected to the log in page, but some feedback I got from a student said that seemed confusing. I changed the log out redirection to the home page, as it makes more sense to be redirected to the starter page.

Admin

On every page of the administration, the first thing I do is check to see if the user is logged in. If they're not logged in, a component named Access msg is displayed, saying you have to be logged in to view this page.

If you are logged in, you will see your email and a log out button below the sub navigation on the dashboard. Below this is a message to the administrator saying that all deletions are final, just as an extra warning to the admin. Under this is where the messages from contacts and enquiries are displayed.

In the edit hotels page, is where you can add, update and delete hotels. The add hotel form is at the bottom of the edit page, and has the same libraries as the other forms. The validated input updates the establishments and fetches them, then the page reloads and you can see your added hotel.

If you want to delete a hotel, this has the same set up as deleting an enquiry. You get a confirmation alert, and once this is clicked it updates the array and deletes the clicked hotel by id. The page is reloaded and you can see that your hotel is gone.

If you want to edit a hotel, this takes you to a form with the sub heading edit hotel. The default state of the input, is the values of the hotel you just clicked on. This is done by fetching the hotel by id using use params, with the same error and loader handler as stated previously. The form has the same validation set up as explained in enquiry, only with different labels.

When you submit the validated form, it updates the hotel by id, and uses a method of patch. After the hotel is updated, I use history push, and you're pushed back to the hotel list.

Feedback

Student feedback list

- Two constants regarding the same thing -> header / fetch options.
- Enquiries pushes to admin on submit, same with log out.

- New hotel did not add photo, only photo url.
- Use footer tag for semantics instead of div.

I found the student feedback very helpful, fresh eyes can catch a lot of silly mistakes. Such as when I fetched arrays, I switched between two constants that are the same but with different names. In order for my code to be easier to read, I changed this to headers instead of fetch options.
She also pointed out that it was confusing when booking a hotel or logging out, to be pushed to admin or log in. This was also an easy fix. I realize now with the new changes, that this makes more sense. I think working on a project for so long, can make you forget that new visitors don't know anything about the page and needs information.

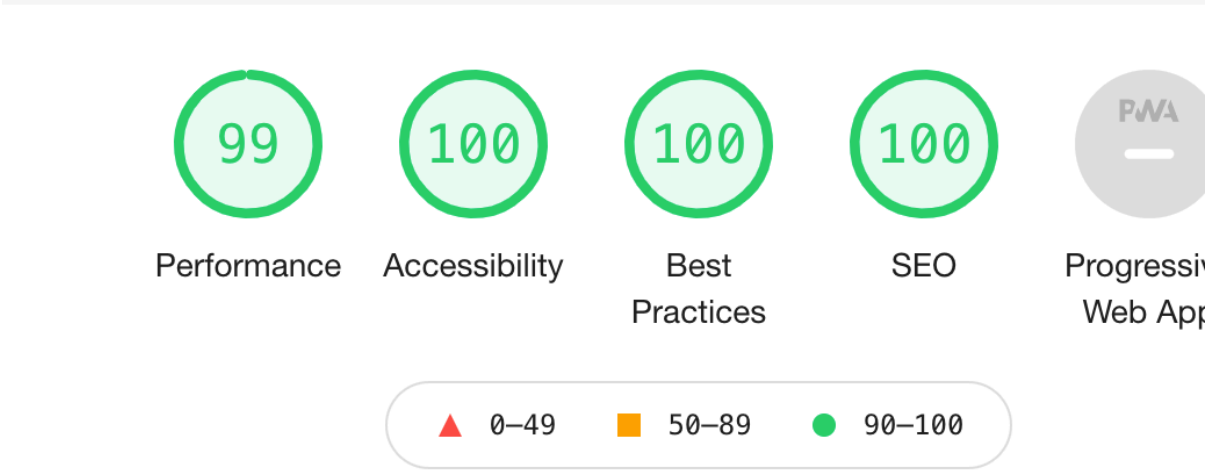
The third point is something I just didn't notice, but when I added a new hotel, it didn't add a photo only the url. Changing the name to image solved this, and it was a quick fix but an important one! When booking hotels, the images is almost the most important thing, at least it is for me.
The last point was that instead of wrapping my footer content in a div, I should change it to footer. This makes my code more semantic, and that is important for accessibility.

Testing

For testing my code my main debugging process is always checking the console. I would work on errors that came along during production. When I was done with every page, I would click back and forth on pages to see if I got errors. I got errors of memory leak on the hotel specific page. By getting the id with use params, I fixed this.

I got some feedback about some weird behavior my website did, on smaller screens. From this I found that my website would render from the bottom, when you clicked a new page. To fix this I created a component called scroll to top, and wrapped this around my router and navigation. This component listens to any changes within the wrapped content, and uses a scroll to top method which is set to the top of the page.

For user experience I tested my website using Chrome DevTools Lighthouse[7]. This allows for me to generate a report and fix problems regarding performance, accessibility, SEO and best practices. I noticed I had forgotten important labels on my forms, regarding accessibility.



This was my result for my home page, which is almost perfect. This test result is very good, and goes on to validate my code.

I also reduced picture size to below 200kb, using an app called ImageOptim[8] (see Figure 1.6). This app reduces your image size to the lowest possible kb, without compromising the resolution. This helped my performance time, and saves rendering time.

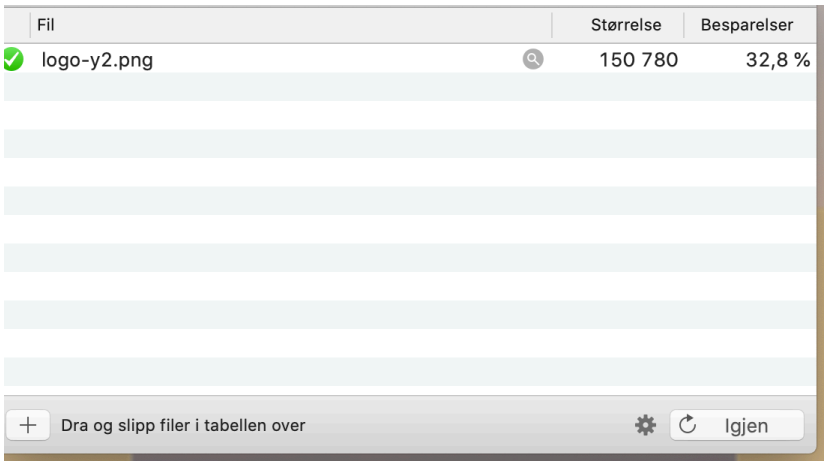


Figure 1.6

After this I did cross browser testing, where I checked my website on Google Chrome, Safari and Firefox. I checked that everything looked the same, is functional and that the design is consistent across multiple browsers.

I performed the same testing across different devices: a desktop, a laptop, Huawei, iPhone, Samsung and an iPad. I was satisfied with the outcome, and this testing is key to ensure user satisfaction and a smooth interaction.

I did get some warnings when opening the bootstrap modal or opening the collapsed navigation. Upon doing some investigation[9], I found this is a common problem that happens to some Bootstrap components. I did try some different offered solutions, but nothing worked. Hopefully this will be updated in the future, but as it is just a warning, I let it be.

Conclusion

In my opinion this website is a good display of my skills both visual and technical. For implementing my code and design, I kept the target audience in mind, and kept everything short, concise and manageable. I went with colors that fit the website's theme, whilst contributing with contrast and personality, but staying accessible.

From my perspective, I believe I have identified issues within my code, and implemented solutions for these issues. An example of this is when my hotel array came up empty, I added a message explaining what the user can do, instead of just a white box with nothing more.

I worked on creating a strong page, with a solid design and I pushed myself when it came to React and all that follows. I can proudly say my page is operable, perceivable, robust and understandable - which should always be the core of any interactive website.

My final product is a reflection of how far along I've come, and is something I'm proud of. Although there is always room for improvement, I can honestly say I've given it my best given my skills at this moment.

Sources are listed below.

Sara Sørensen.

References

1. iCloud - https://www.icloud.com/
2. Wikipedia - https://no.wikipedia.org/wiki/Bl%C3%A5
   https://nn.wikipedia.org/wiki/Gul
3. Google Fonts - https://fonts.google.com/
4. Bootstrap Icons - https://www.npmjs.com/package/react-bootstrap-icons
5. Pixabay - https://pixabay.com/no/
6. StackOverFlow - https://stackoverflow.com/questions/32553158/detect-click-outside-react-component
7. ChromDevTools Lighthouse - https://developers.google.com/web/tools/lighthouse
8. ImageOptim - https://imageoptim.com/mac
9. Bootstrap Error - https://github.com/styled-components/styled-components/issues/2154