



**Noroff**

School of technology  
and digital media

# Technical Report

## Project Exam Two

James Farmer

Word count

Summary: 142 | Main text: 958

### Links:

Site: <https://neon-hamster-1b3f8e.netlify.app/>

Github: <https://github.com/Noroff-FEU-Assignments/project-exam-2-Jimbo-Farmer>

Prototype- mobile

<https://xd.adobe.com/view/9f9c158b-b204-4451-b446-d02c7aa6571d-cacf/?fullscreen&hints=off>

Prototype- desktop

<https://xd.adobe.com/view/98fb5670-4e17-476c-8042-8a512151f431-cda8/?fullscreen&hints=off>



# Table of Contents

1. Summary.....	3
2. Body.....	4
2.1. Introduction.....	4
2.2. Main section of report.....	4
Planning and research.....	4
Design and prototyping.....	4
Build.....	5
Review.....	6
2.3. Conclusion.....	7
3. References.....	8
4. Acknowledgements.....	9
5. Appendices.....	10



# 1. Summary

Over the course of seven weeks a website for browsing and booking holiday accommodation – Holidayze – was planned, designed, built, deployed and tested.

The first two weeks were used for planning, research, design and prototyping. The Gantt chart – appendix 1, style tile – appendix 2, and prototypes (see links above) were created using Adobe XD.

Weeks three to six were used for building the site. First the CMS was set up, then the site was built using Nextjs and deployed on Netlify. The site fetches accommodation information from a Strapi CMS that is hosted on Heroku with Cloudinary.

Week seven was used for testing, review, bug-fixing and writing this report. The site has been tested on mobile (iPhone 7 plus), tablet (iPad 10"), 15" laptop and 32" monitor and has no layout or design issues on any of these devices.



## 2. Body

### 2.1. Introduction

The project brief required the planning, design and building of a website called Holidaze which allows users to browse through holiday accommodation in the Bergen area of Norway. Users can get in touch via contact form and make a booking enquiry for a specific property via an enquiry form. An administrator can log in to view messages from users and create new property listings.

### 2.2. Main section of report

#### Planning and research

First, a Gantt chart was created (see appendix 1) which follows the recommendations given in the project briefing and adds a few details. Then a review of websites for holiday accommodation was conducted to look for common themes, useful features and pain points. Booking.com, Hotels.com, thonhotels.no, Nordic Choice Hotels were reviewed. Common themes include a prominent search form at the top of the homepage and imagery that evokes feelings of wanderlust and comfort. Search results are displayed in a single column with few distractions in order to encourage the user to focus on finding the accommodation that is right for them. These themes and features have been replicated in Holidaze. A pain point with the sites reviewed was that they contain rather a lot of noise and an overwhelming amount of options. Holidaze by it's specialized (small and local) nature will have more of a boutique feel so should be able to avoid this.

Target audience ranges from backpackers looking for cheap accommodation to couples searching for a weekend away and families planning a weeks-long holiday. Due to the variety of the target audience the styles will be kept fairly neutral to try to create a broad appeal.

#### Design and prototyping

A style tile (see appendix 2) was created with simple and clear sans-serif fonts, Norwegian nature-inspired colours and imagery to entice people to explore Norway's west coast. Much of the imagery used for the final website was generated using AI (DALL-E 2) which gave the ability to create exactly what I wanted and the images are free to use. The hero image on the homepage, for example, was generated from the description 'a small wooden sailing boat on a Norwegian fjord under a clear blue sky with a cottage in the distance'. This gave the blue sky to use as a background for white text together with the sense of exploration (the boat), some cosy accommodation (the cabin) and impressive Norwegian nature (or at least what the AI thinks a fjord looks like!).

The mobile prototype was created first, and then expanded to desktop size. The layout for most of the pages is broadly the same with wide margins on larger screen sizes to avoid clutter. In my experience Adobe XD has proved a very useful tool for trying different layouts and themes, but does not manage to give a completely realistic feel of what the final website will be like. For this reason there are a number of small discrepancies between the



prototypes and the final product as a number of details were decided on during the build phase.

## Build

The first task completed was setting up the Strapi API and deploying with Heroku. Following the video tutorial provided in the project scope made this a straightforward task.

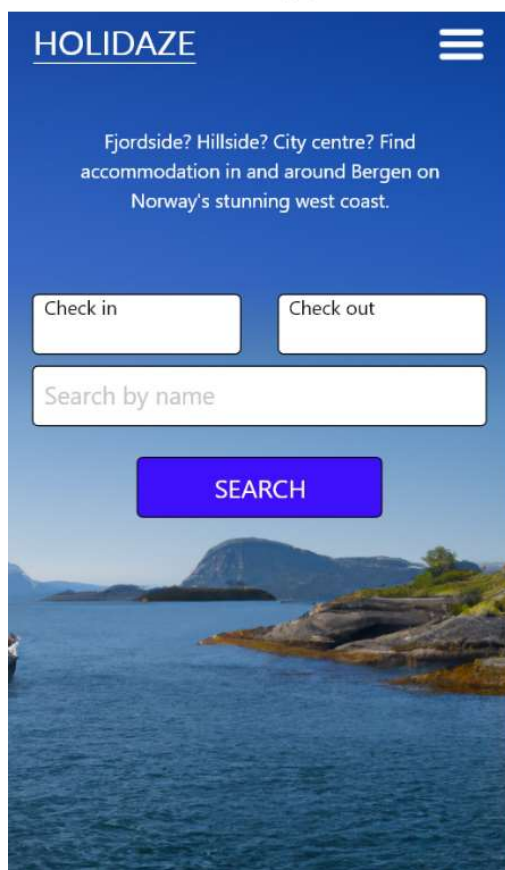
Nextjs was chosen as the content of the site is not very dynamic so server side rendering makes sense to speed up response times. Certain areas such as routing and image optimisation are also made easier with Next's built-in features. SCSS was used for styling as this is the method I am most proficient with.

Pages were created and the first components were Layout and Head to get the basic page layout. The homepage was created first, followed by the rest of the customer-facing pages and then the administration pages.

Online guides and tutorials were used for some of the more difficult areas, such as creating the enquiry modal and the create new accommodation form, see section 3 below (references) for a complete list.

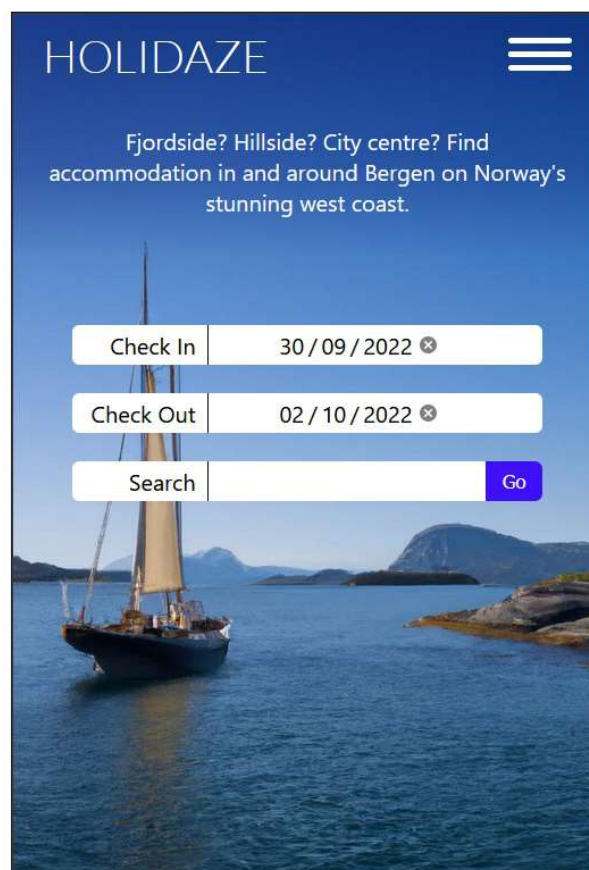
The functionality of the site was the first priority, as this is where most things can go wrong and take time to fix. Styling was added afterwards as this is generally more straightforward. As mentioned above, the design of some pages was changed slightly during the build as the weeks-long build process gave time to reflect on and improve certain elements. For example the search fields and other minor details on the homepage as seen below.

Prototype



The prototype shows a dark blue header with the 'HOLIDAZE' logo and a hamburger menu. Below the header, a light blue banner contains the text: 'Fjordside? Hillside? City centre? Find accommodation in and around Bergen on Norway's stunning west coast.' The search section consists of two separate input fields for 'Check in' and 'Check out', followed by a single large input field for 'Search by name'. A prominent blue 'SEARCH' button is positioned below these fields. The background is a scenic image of a fjord with a boat.

Final



The final version features a more refined search interface. The header and banner text remain the same. The search section now includes three distinct input fields: 'Check In' with the date '30 / 09 / 2022', 'Check Out' with the date '02 / 10 / 2022', and a 'Search' field with a 'Go' button. Each date field has a small 'x' icon for clearing the input. The background image of the fjord and boat is consistent with the prototype.

Deploying the site to Netlify revealed several bugs that were stopping the site from being built, but the build logs were helpful in identifying and fixing the problems. Some of which involved localStorage and its inaccessibility from the server side. These problems were overcome fairly easily.

## Review

Having deployed the site, testing was carried out on mobile, tablet and desktop and bugs were fixed. JsDocs were added to make the code more legible, and a code review was undertaken by classmate Sophie Haugland, see appendices. Sophie made some excellent recommendations that were implemented such as new components where there was a lot of repetition. The ContactLayout, LoginLayout, CreateLayout and EnquiryModalLayout were created as the pages that involve forms to be submitted return different HTML depending on success, failure, sending of the forms they contain. See below.

```
157 if(submitting){
158   return (
159     <Layout pageId="create-page">
160       <Head title='Create' description="Create a new accommodation listing."/>
161       <div id='create-page__container' className='main'>
162         <div className='create-intro page-intro'>
163           <Link href='/login'><a className='dashboard-link'>Back to Dashboard</a></Link>
164           <h1>Create New Listing</h1>
165           <p>Submitting</p>
166         </div>
167         <div className='loading'></div>
168       </div>
169     </Layout>
170   )
171 }
172
173 if(success){
174   return (
175     <Layout pageId="create-page">
176       <Head title='Create' description="Create a new accommodation listing."/>
177       <div id='create-page__container' className='main'>
178         <div className='create-intro page-intro'>
179           <Link href='/login'><a className='dashboard-link'>Back to Dashboard</a></Link>
180           <h1>Create New Listing</h1>
181           <p>New accommodation created successfully!</p>
182         </div>
183         <div className='button-container'>
184           <button onClick={handleClick}>Add Another</button>
185         </div>
186       </div>
187     </Layout>
188   )
189 }
190
191 if(sendError){
192   return(
193     <Layout pageId="create-page">
194       <Head title='Create' description="Create a new accommodation listing."/>
195       <div id='create-page__container' className='main'>
196         <div className='create-intro page-intro'>
197           <Link href='/login'><a className='dashboard-link'>Back to Dashboard</a></Link>
198           <h1>Create New Listing</h1>
199           <p>An error has occurred.</p>
200         </div>
201       </div>
202     </Layout>
203   )
204 }
```

Before – 47 lines for various outcomes on the create page depending on submitting, success or sendError.



```

98   if(submitting){
99       return (
100         <CreateLayout intro={'Submitting'} loading={true} />
101       )
102   }
103
104   if(success){
105       return (
106         <CreateLayout intro={'New accommodation created successfully!'} loading={false} >
107           <div className='button-container'>
108             <button onClick={handleClick}>Add Another</button>
109           </div>
110         </CreateLayout>
111       )
112   }
113
114   if(sendError){
115       return(
116         <CreateLayout intro={sendError} loading={false} />
117       )
118   }

```

After – 20 lines of code.

## 2.3. Conclusion

The site was planned, designed and built without too many hindrances along the way. The site functions as per the specification.

There are some areas for improvement: the exact presentation and styling of some components could possibly be improved and some functionality could be added such as letting an administrator update accommodation information. Overall I am satisfied with the end result.



### 3. References

'Modal component with Next.js'. Alexander Rusev. January 2021

<https://devrecipes.net/modal-component-with-next-js/>

'How to upload an image to strapi'. Bassel Kanso. July 2021

<https://dev.to/bassel17/how-to-upload-an-image-to-strapi-2hhg>

'Accessing LocalStorage in NextJS'. Ibrahim Adeniyi. October 2020

<https://dev.to/dendekky/accessing-localstorage-in-nextjs-39he>

'Context and localStorage using hooks'. Jimode. February 2020

<https://gist.github.com/jimode/c1d2d4c1ab33ba1b7be8be8c50d64555>

'CSS Underline using pseudo styles'. Boudi. <unknown date>

<https://codepen.io/elb96/pen/aXKXBx>

<Multiple pages>

<https://developer.mozilla.org/en-US/>





## 4. Acknowledgements

Thanks to Sophie Haugland for providing a thorough code review with multiple helpful suggestions for improvement.



## 5. Appendices



# Appendix 1. Gantt Chart



## Appendix 2. Style Tile

# Holidaze

*Your gateway to the fjords*

---

### Brand adjectives

- *Adventurous*
- **Fun**
- Trustworthy
- Clear and informative

---

### Colour



Star Sky



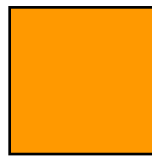
Forest Green



Warning Red



Trollsk Blue



Sunrise Orange

---

### Typography

Source Sans Pro - Headings

Segoe UI - Body text

lorem ipsum dolor Fools savory slow stuffing soft inspection magnificence Brandybuck village swore ghost seeks. Arwen Evenstar bandy heed stolen lingered.

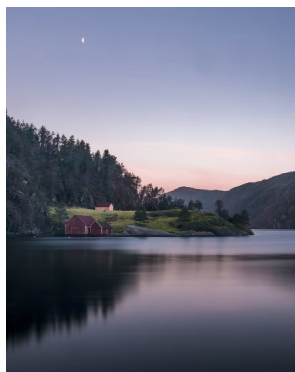
---

### Image and texture

Smooth texture to avoid feeling of noise and confusion. Images to induce wanderlust through a sense of exploration and adventure, using the stunning western Norwegian scenery.

Use 50/50 town/countryside.

Colours to roughly reflect chosen colour palette.



---

### Icons and buttons

SUBMIT

SEND

DELETE

## Appendix 3. Code Review

Firstly really nice app and really nicely coded, there isn't too much for me to comment on. The things that I have picked out may be partly personal preference, and mostly to do with keeping things neat and tidy. Everything works well and as I would expect it too. All in all there isn't much I would change.

### 1. Clearer variable names

Would be nice to have slightly clearer variable names e.g. when you have something like:

```
src={acc.attributes.images.data[0].attributes.url}
```

It would be nice to create a variable like `imgUrl = ...`

It just makes it a bit easier to quickly read and understand whats going on

### 2. Nice use of JSDocs

### 3. Passing many props through components

```
return <EnquiryCard key={enquiry.id} accommodation={enquiry.attributes.accommodationname}  
checkin={enquiry.attributes.checkin} checkout={enquiry.attributes.checkout} adults={enquiry.attributes.adults}  
noOfChildren={enquiry.attributes.children} name={enquiry.attributes.name} email={enquiry.attributes.email}  
phone={enquiry.attributes.phone} query={enquiry.attributes.query} />
```

Just for readability I would maybe consider something like passing one prop,

```
return <EnquiryCard key={enquiry.id} accommodationInfo={enquiry.attributes} />;
```

and then destructure it inside the component rather.

### 3. Repeated code.

There seems to some functions that are fairly similar, such as `onSubmit`. I wonder if you could perhaps have a single function that can take a url and function of what to do with the response for example. But perhaps not if that makes for a very messy function.

I would be tempted to create separate files for some of your functions anyway or even make some hooks, especially for your api calls. It can keep the component looking a bit neater and could cut quite a few lines of code.

### 4. File extensions

I notice some of your files are `.jsx` and some `.js` I know it doesn't matter but for consistency I guess anyone where you are using react could be `.jsx`. It also makes it a bit clearer when looking through the files.

### 5. Create components for repeated code

For example the EnquiryModal page there are a lot of repeats there, you could maybe create a ModalContainer component or something that has the close button and the title for example and then you can pass it children, I think this could shave a few lines of code.

This would also be good for the CreateAccommodation page.

## 6. Resetting forms

A neat way to reset the form could be to use useRef, you can access the form elements then. This would allow you to add it to the onSubmit function for example. When it is successful you could reset the form. This would prevent you from needing to use querySelector (in Create.js).

## 7. Cleanup your eventListener

In your index.js file you use an eventListener inside your useEffect. It is important to always clear up these as you could end up listening for events many times and also it can cause memory leaks. Something like this:

```
useEffect(() => {  
  window.addEventListener('scroll', handleScroll);  
  
  return () => {  
    window.removeEventListener('scroll', handleScroll);  
  };  
})
```

## 8. Check out moment.js for your dateDefault function

It has some really nice ways of formatting dates could reduce quite a bit of this code.