



Technical Implementation

Introduction

Purpose: This document provides a comprehensive overview of the technical implementation of the project, establishing a setup using Next.js and Contentful CMS. This setup allows non-technical users and the marketing team at Cape to create, edit, and delete pages and components on Cape's marketing website without needing assistance from developers.

Scope: The document covers the background, objectives, project overview, implementation details, key features, challenges, solutions, and future enhancements of the setup.

Project Overview: The current setup using Next.js and Contentful allows Cape's marketing team and non-developers to create, edit, and delete pages and components on the marketing website without the help of developers.

Background and Objectives

Background: Currently, every time the marketing team at Cape wants to update the marketing website, they rely on the developers. However, the developers are mostly busy with other projects, causing delays in updating the marketing website. Therefore, the marketing team needs a way to update the website without the help of developers.

Objective: To find a way for the marketing team and non-developers at Cape to create, edit, and delete new pages and components on their own without the help of developers.

Project Overview

The current setup of the project allows Cape's marketing team to create new pages with any title they want, choose which components to include on the page, and have the freedom to choose their order and edit them. This flexibility applies to most elements of the marketing website.

Key Features:

- **Changing the order of components:** The marketing team at Cape has the freedom to arrange the order of the components as they like by dragging them up and down to change the order.
- **Dynamic components:** Every component they create can be used on any page they create on the marketing website, and it will adapt to the page where it is used.
- **Rely less on developers:** The marketing team at Cape doesn't need any background knowledge of development. They can manage the website on their own without the help of developers.

Implementation Details

Architecture Overview:

The project uses a two-tier architecture:

- **Front-End Layer:**
 - **Technology:** Next.js

- **Function:** Responsible for rendering the website's user interface, handling client-side interactions, routing, and integrating with the CMS to fetch and display content.
- **Content Management Layer:**
 - **Technology:** Contentful CMS
 - **Function:** Manages the creation, editing, and storage of content. Provides APIs to deliver content to the front-end.

Technology Stack:

- **Next.js:**
 - A React framework used for building the front-end of the website.
 - Provides server-side rendering, static site generation, and client-side rendering capabilities.
- **Contentful CMS:**
 - A headless content management system used for creating, managing, and delivering content.
 - Provides APIs to integrate with the front-end.
- **Version Control(Gitlab):**
 - For code management and collaboration.

Development process:

I followed an Agile development methodology with bi-weekly sprints. I create my tickets with guidance from the company mentor. Once I finish a ticket, I request a code review. After the code review, a functional review is conducted, and then the branch is pushed to the development branch and finally to production.

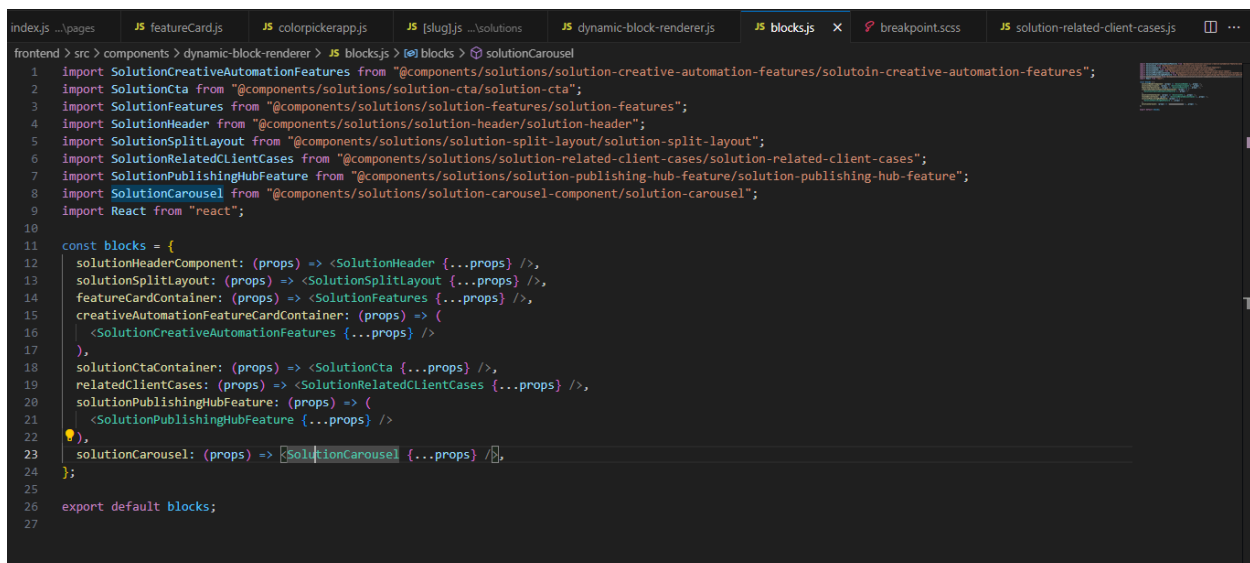
Challenges:

The main challenge I faced was after brainstorming the solution and discussing it with the developers, it was time to implement it. I proposed creating separate blocks and adding them to a reference field. The challenge was to render them based on the order of the array in the reference field, where the order of objects could be changed. Another challenge was finding a way to create external apps and integrate them into the Contentful CMS interface. I have explained the solution using some of the code snippets below.

Render the components based on the order of the array:

Step One:

I created dynamic reusable components and passed the props to the components based on the content type ID of each component.



```
1 import SolutionCreativeAutomationFeatures from "@components/solutions/solution-creative-automation-features/solution-creative-automation-features";
2 import SolutionCta from "@components/solutions/solution-cta/solution-cta";
3 import SolutionFeatures from "@components/solutions/solution-features/solution-features";
4 import SolutionHeader from "@components/solutions/solution-header/solution-header";
5 import SolutionSplitLayout from "@components/solutions/solution-split-layout/solution-split-layout";
6 import SolutionRelatedClientCases from "@components/solutions/solution-related-client-cases/solution-related-client-cases";
7 import SolutionPublishingHubFeature from "@components/solutions/solution-publishing-hub-feature/solution-publishing-hub-feature";
8 import SolutionCarousel from "@components/solutions/solution-carousel-component/solution-carousel";
9 import React from "react";
10
11 const blocks = {
12   solutionHeaderComponent: (props) => <SolutionHeader {...props} />,
13   solutionSplitLayout: (props) => <SolutionSplitLayout {...props} />,
14   featureCardContainer: (props) => <SolutionFeatures {...props} />,
15   creativeAutomationFeatureCardContainer: (props) => (
16     <SolutionCreativeAutomationFeatures {...props} />
17   ),
18   solutionCtaContainer: (props) => <SolutionCta {...props} />,
19   relatedClientCases: (props) => <SolutionRelatedClientCases {...props} />,
20   solutionPublishingHubFeature: (props) => (
21     <SolutionPublishingHubFeature {...props} />
22   ),
23   solutionCarousel: (props) => <SolutionCarousel {...props} />,
24 };
25
26 export default blocks;
```

Step Two:

I imported the blocks created in the previous step. Here, I fetched the data for each component based on its content type and ID.

```
pages JS featureCard.js JS colorpickerapp.js JS colorPickerApp.js JS [slug]js ...solutions JS dynamic-block-renderer.js X JS blocks.js breakpoint.scss JS solution-related-cli ...

frontend > src > components > dynamic-block-renderer > JS dynamic-block-renderer.js > [e] DynamicBlockRenderere

1 import React from 'react';
2
3 import blocks from './blocks.js';
4
5 const DynamicBlockRenderere = ({ pageBlocks, color }) => {
6   return (
7     <div>
8       {pageBlocks.map((pageBlock, index) => {
9         const dynamicBlock = blocks[pageBlock.sys.contentType.sys.id];
10        if (dynamicBlock) return dynamicBlock({ pageBlock, index, color });
11        return null;
12      })}
13     </div>
14   );
15 };
16
17 export default DynamicBlockRenderere;
18
```

Step Three:

In this step, I fetched the data from Contentful and passed it to the dynamic block renderer component.

```
pages JS featureCard.js JS colorpickerapp.js JS colorPickerApp.js JS [slug]js ...solutions X JS dynamic-block-renderer.js JS blocks.js breakpoint.scss JS solution-related-cli ...

frontend > pages > solutions > JS [slug]js > [e] SolutionsPage

1 import DynamicBlockRenderere from '@components/dynamic-block-renderer/dynamic-block-renderer';
2 import Page from '@components/page/page.component';
3 import DynamicService from '@services/dynamic/dynamic.service';
4 import Head from 'next/head';
5 import PropTypes from 'prop-types';
6 import React from 'react';
7
8 const SolutionsPage = ({ pageInfo, solutionsPage }) => {
9   return (
10     <Page pageInfo={pageInfo}>
11       <div>
12         <DynamicBlockRenderere
13           pageBlocks={solutionsPage.pageBlocks}
14           color={solutionsPage.colorPicker}
15         />
16       </div>
17     </Page>
18   );
19 };
20
21 SolutionsPage.propTypes = {
22   solutionsPage: PropTypes.object
23 };
24
25 SolutionsPage.defaultProps = {
26   solutionsPage: {}
27 };
28
29 export const getServerSideProps = async ({ params }) => {
30   try {
31     const { item } = await DynamicService.getItem({
32       content_type: 'solutionsPage',
33       'fields.slug': params.slug,
34       include: 2
35     });
36     if (!item) {
37       throw new Error('No dynamic item found');
38     }
39   }
40 }
```

```

39     return {
40       props: {
41         solutionsPage: item
42       }
43     };
44   } catch (error) {
45     return {
46       redirect: {
47         destination: '/cases',
48         permanent: false
49       }
50     };
51   }
52 };
53
54 export default SolutionsPage;
55

```

Creating custom apps and adding them to Contentful:

I referred to the documentation of Contentful to learn how to implement an external custom field. I used the React Color Picker library for color picking and utilized the Contentful SDK to create a custom app.

<https://www.contentful.com/developers/docs/extensibility/app-framework/sdk/>

```

JS index.js ...pages JS featureCard.js JS colorpickerapp.js JS colorPickerApp.js X JS [slug].js ...solutions JS dynamic-block-renderer.js JS blocks.js JS breakpoint.scss JS solu ...
frontend > src > components > custom-apps > JS colorPickerApp.js > ...
1 import { useSDK } from '@contentful/react-apps-toolkit';
2 import React, { useEffect, useState } from 'react';
3 import { SketchPicker } from 'react-color';
4
5 const ColorPickerApp = () => {
6   const sdk = useSDK();
7   const [color, setColor] = useState();
8
9   useEffect(() => {
10     sdk.window.startAutoResizer();
11     setColor(sdk.field.getValue());
12   }, [sdk.window, sdk.field]);
13
14   const handleChange = (color) => {
15     setColor(color.rgb);
16     sdk.field.setValue(color);
17   };
18
19   return <SketchPicker color={color} onChange={handleChange} />;
20 };
21
22 export default ColorPickerApp;
23

```

Future Enhancements

More dynamic components can be created:

With the current setup, only a few components can be completely dynamic and created by Cape's marketing team. In the future, more dynamic components can be created and set up through Contentful.

Multilanguage:

The multilanguage feature was not able to be implemented due to the current setup of the project, but it can be implemented in the future.

Conclusion

Summary: The current setup of the project provides a comprehensive solution for Cape's marketing team to create, edit, and delete some pages and components on Cape's marketing website without the help of developers. This document has outlined the product's background, implementation, key features, challenges, and future enhancements.