Creating a new React based navigation for Frosmo Control Panel (FCP)

Biswas K C

Haaga-Helia University of Applied Sciences

Bachelor's Thesis

2022

Bachelor of Business Administration

Abstract

Author(s)
Biswas K C

Degree
Bachelor of Business Administration, BITE

Report/Thesis Title
Creating a new React based navigation for Frosmo Control Panel (FCP)

Number of pages and appendix pages
44 + 9

Frosmo Oy provides a SaaS platform called as Frosmo Platform, which is a personalization software used for improving website's functionality and personalizing online user experiences. Frosmo Control Panel (FCP) is the main user interface of the Frosmo Platform. FCP is used mainly by Frosmo's customers to customize their sites and monitor their site's performance.

This thesis describes and outlines the creation and implementation of a new React-based navigation for FCP. The new navigation is interactive and include a customizable section and functionality for adding, editing, and removing users' favorite items.

New navigation is created with React & TypeScript. The development team at Frosmo uses Git and GitLab for version control and development. For project management, Jira is used, and Slack is used for internal communications.

The need for the React-based navigation for FCP arose from FCP users, and their need to have an interactive navigation, which provides flexibility and options to customize. With the implementation of new navigation, FCP users can receive practical benefits. As they can find all the necessary FCP elements in the navigation and, they can customize the navigation elements according to their preference in Favorites menu section. Frosmo also receives benefits from this as this implementation develops and improves their platform.

Key words
Frosmo Control Panel (FCP), Navigation, Navigation Item, React.

Table of content.

Table of figures

List of tables

Key Terminology

This section explains some of the terms mentioned in the research paper.

Company **–** Company in FCP refers to customer or partner account. A company can have one or more sites and if a user has access to a company, then automatically has access to all sites of the company (Frosmo Limited, 2022a).

Conversion – Conversions are actions that site owners want their visitors to take. Actions can be buying a product, signing up for a different newsletter etc. Conversion can be defined as per business goals (Frosmo Limited, 2022a).

Document Object Model (DOM) – It is the data representation of the objects, which include the structure and content of a document on the web (Mozilla, 2022d).

Graniitti API – Graniitti API provides access to data in Frosmo back end and it can be used to receive, create, update, and remove different types of data such as company settings, modifications, sites, users etc (Frosmo Limited, 2022b).

Navigation Context – It is a TypeScript file inside FCP's react directory where all the Navigation Items are fetched and processed and is passed to other components using React's "useContext" variable.

Npm – It is the package manager for JavaScript run time environment node version.

Personalization – It is how the user customizes different web-based interfaces or desktop to assemble all the personal preferences. Personalization supplies more relevant user experience, which can drive conversions and create revenue (Frosmo Limited, 2022a).

Product attribute – It is generally a single characteristic of a product which can be such as name, id etc. The attributes of a product collectively make product data (Frosmo Limited, 2022a).

Product view – Product view is a visitor action whereby the visitor views product information on a site, typically on a product page (Frosmo Limited, 2022a).

SiteContext – It is a TypeScript file inside FCP's react directory, where all the site settings such as user settings, "add-ons" settings, company settings etc. are created. From here the settings are passed to other react components using React's "useContext" hooks variables.

Site – Site in Frosmo Platform refers to actual website of a company. Each site has its custom scripts which have Frosmo specific configurations as well as specific data and statistics on the website are tracked by the Frosmo Platform (Frosmo Limited, 2022a).

Transactions – It is simultaneous purchase of one or more products and Frosmo Platform registers each transaction as single conversion (Frosmo Limited, 2022a).

useGranittiApi hook – It is a custom hook available in react code of FCP. It is used to fetch data using Graniitti API.

1 Introduction

The purpose of the thesis is to create and implement fully functional navigation for Frosmo Control Panel (FCP), using only the technology available on the FCP.

1.1 About the Company

Frosmo Oy provides a SaaS platform, a personalization software for improving website's functionality and personalizing online user experiences. Frosmo's main target markets are e-commerce and iGaming sites. Customers of Frosmo range from webstores to betting companies such as Power, Swisslos, Dafabet, Elisa, etc. (Frosmo Limited, 2022e).

Figure 1. Frosmo Oy's banner (Frosmo Limited, 2022e)

1.2 About the Project

Frosmo Platform is a web UI (User Interface) development solution which Frosmo's customers use for improving their website's functionality and personalizing online user experiences. It performs two tasks on a website, i.e., enhance site's visitors' user experience and tracks the visitor's behavior and produces data that platform uses. (Frosmo Limited, 2022d). Frosmo Platform consists of three main components:

1. Frosmo JavaScript Library – It is the Frosmo presence in the Front-end and it handles all modifications to the site, manages segmentation of the user, collect usage data, and fetches content to display from the back end (Frosmo Limited, 2022d).

2. Frosmo Back end – It stores the usage data collected by Frosmo JavaScript Library and further processes the data for reporting and analytics purposes. In here, operational data related to modifications and other configurable figures are also

stored. And it also provides different APIs that allow Frosmo JavaScript Library to interact with the back end (Frosmo Limited, 2022d).

–

3. Frosmo Control Panel – FCP is the main user interface of the Frosmo Platform. It is used to customize site and check its performance. Any change made to a site in FCP are automatically reflected in the custom script for that site and under the hood FCP manages Frosmo JavaScript library (Frosmo Limited, 2022c). Mainly, FCP is used to:
   - Configure the content and user experience of a site based on who the user is how the user is using the site.
   - Configure how visitors are segmented by the platform on a site.
   - Create and manage user accounts for a company or organization.

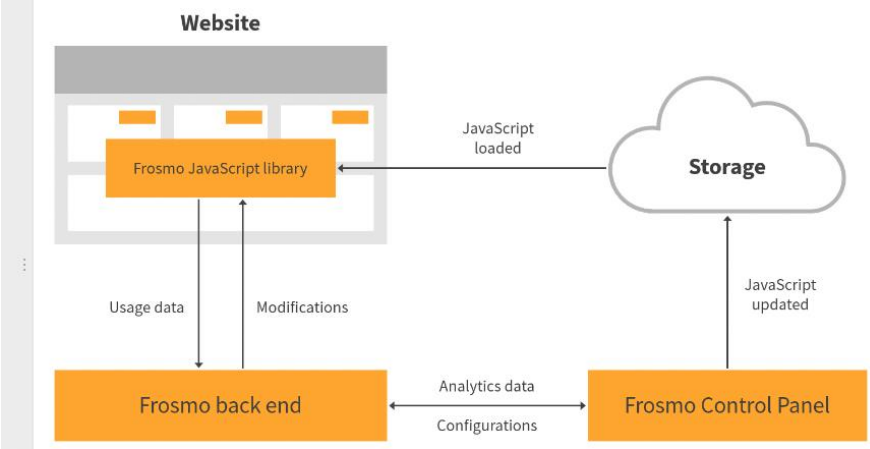

Figure 2. Frosmo Platform architecture and information flow (Frosmo Limited, 2022d)

The primary goal of this thesis is to create a new React based navigation for FCP. Users of FCP vary from Frosmo's customers, developers to marketing personnel at Frosmo, and their need to use the FCP can vary according to their roles. So, one of the main goals of this research is to create a user-based personalized navigation which would allow users more flexibility around the platform. More specifically, users should be able to access all the necessary elements in the FCP through navigation. Also, users can customize the navigation content and have an option to add, remove and edit the contents in navigation.

1.3 Product-oriented thesis

The thesis is a product-oriented thesis. Upon the thesis's completion, there will be a thesis report and a product, React-based navigation will be available in FCP.

1.3.1 Need for the product

Currently, the navigation in FCP does not provide any flexibility to users, as it does not have any customizable features or functionalities. And, since there are different kind of users, using FCP for different purposes, their needs also vary, and users have expressed their wish to have a user-based customizable navigation, which will give them freedom to have preferred items in the navigation and provides them more options to navigate around FCP.

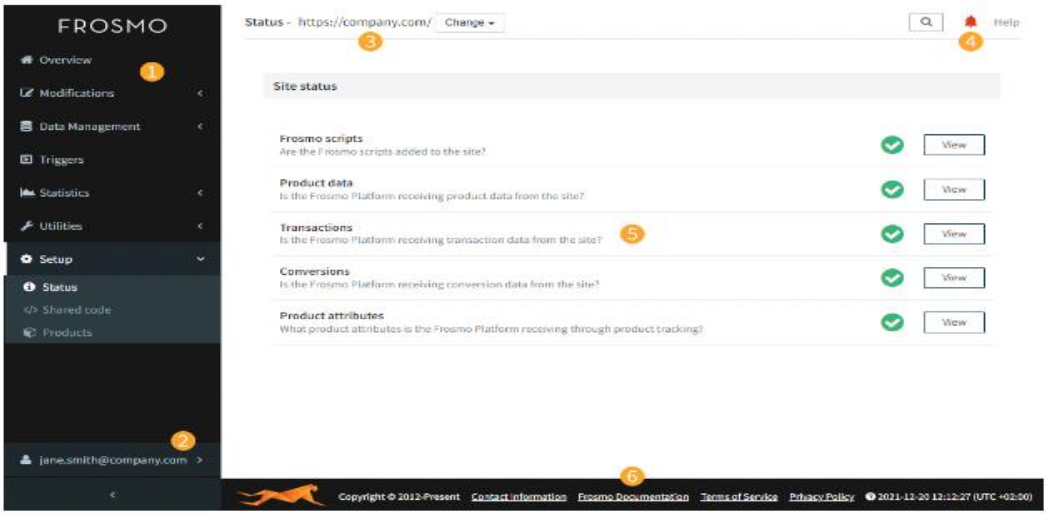

Figure 3. Current FCP (Frosmo Control Panel) with different numbers to show different parts of FCP.

Different part of current navigation is discussed below:

- Main navigation – In figure 3, the user menu is marked as number 1 and consists of all the links FCP pages.
- User Menu – In figure 3, the user menu is marked as number 2. The user menu is below the main navigation and contains all the links related to the user such as user activity and log out.
- Top bar – In figure 3, user menu is marked as number 3 and 4. Top bar navigation contains customer's information, a search bar, news section and help menu.
- Footer – Footer is marked as number 6 in figure 3.

The current navigation is outdated and has different usability issues. The goal of the new navigation is to replace the current navigation with new navigation. The new navigation will provide more flexibility to users while using the platform. With customizable, user-based personalized navigation every user could easily access any preferred FCP content from the navigation. Users could customize the navigation, can add or remove their favorite contents, i.e., links, tools, features, or statistics in the navigation. Also, new navigation will include various kinds of menus, which contains similar kinds of FCP elements.

1.3.2 Support from the commissioning party

The product team in Frosmo consists of nine full-time employees and during project development, team support is available. UX developer will be helping me in this project by creating and implementing the design and creating layouts for the navigation components. Most of the functional components and features of the project will be created by me and the code written to develop them will be reviewed by senior developers.

1.4 Scope of the Thesis

The thesis focuses on creating and implementing React-based navigation and the explains the process for doing that. And currently, pages in FCP are written in different languages and the new React-based navigation must be compatible with all the pages in FCP. The focus of the research is solely on the navigation created in React code, and any changes made in code other than React is out of the scope of this thesis.

Furthermore, the personalized links in the navigation will require creating an API, which will be created by other back-end developers and is out of the scope of the thesis. Also, some of the navigation elements are not visible to all the users and it requires different kinds of Frosmo's settings to be enabled first, and this research paper does not cover how those settings are enabled.

1.5 Project organizations and Copyrights

The stakeholders in this project are senior document specialists, product manager, product owner, and other team members of product team in Frosmo Oy.

Frosmo Oy holds all rights to the FCP, including the product described in the thesis. A thesis commissioning agreement was signed by the commissioning party and me on 9 September 2022.

2 Theory and research

I conducted research to create an overall project structure and how the navigation's data structure should be created before starting to implement the navigation.

The new React based navigation must work with all the pages in FCP, written in different languages, for instance, PHP, Angular, Angular2 and React. The old navigation that was written in custom JavaScript will be phased out once the new navigation is deployed in production. The navigation is going to be created using React and TypeScript.

There are many JavaScript frameworks and libraries at the current moment that are being used for the front-end UI development. Different frameworks are compared, and React JS is chosen, as it is continuously supported and developed JavaScript library and it has advantage over other frameworks. Another reason to choose React is that currently all the new features in FCP are created using React and TypeScript.

2.1 TypeScript

TypeScript is an open-source programming language created and maintained by Microsoft. It is a programming language that includes all the JavaScript syntax alongside TypeScript syntax while defining and using types. TypeScript runs a compiler to check the types of all the created JavaScript functions and variables, reports if there is any error and then outputs the JavaScript code. TypeScript helps developers by using type checkers which tell letting editors such as, VS code how it can provide helpful services to developers (Goldberg, 2022).

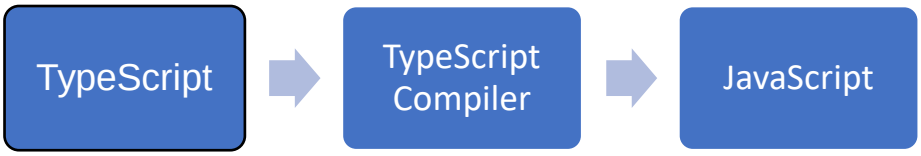

Figure 4. How does TypeScript work

Type in TypeScript: It describes what the form of JavaScript value might be, meaning what the built-in "typeof" operator would describe it as and what properties and methods would exist on a value (Goldberg, 2022). For instance, if a variable is created with an initial value "Pigeon", then TypeScript can figure out that the bird variable is of type string.

TypeScript provides two ways to create custom types for our data and they include Type aliases and Interfaces (Agboola, 2022).

Type aliases: The type aliases are a name of a type that is used to represent different primitives, for instance, string, number, “Boolean”, etc. Furthermore, it is also used to stand for object type.

Object type refers to any JavaScript value with properties, and to define this, its properties and types are listed (Microsoft, 2022a). Object types can be named by using type aliases and interface declaration.

Interface: Interface is used to name object type. Its syntax is different than that of type aliases.

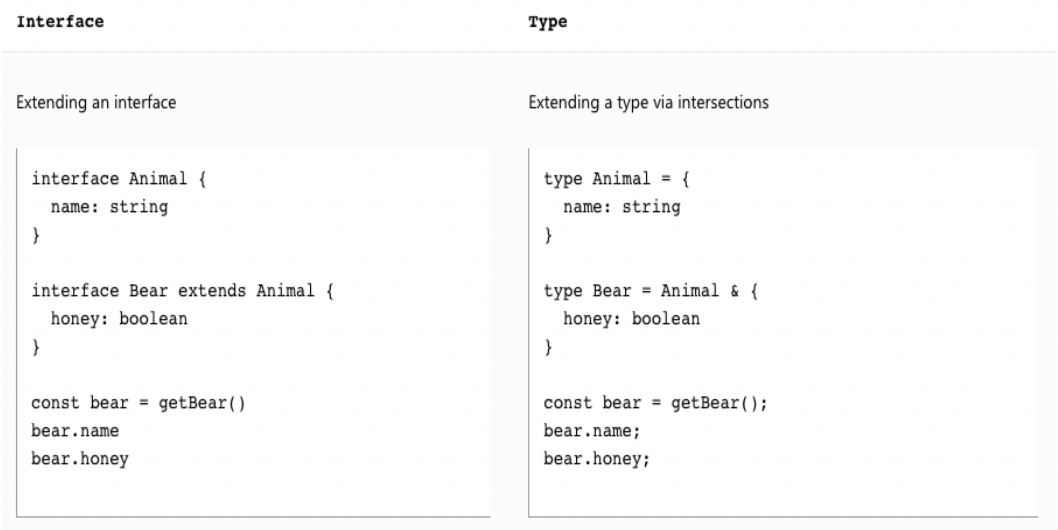

Figure 5. Difference between interface and type alias while naming an object type (Microsoft, 2022b)

2.2 React

React is JavaScript library which is used for building user interfaces. It is declarative, efficient, and flexible. It lets developers create complex UIs (User Interface) from small pieces of code called components. Components are reusable and independent and conceptually components are like JavaScript functions which accepts a single props or properties object as argument with data and returns a React element (Meta Platforms, Inc., 2022c). An example of React component is shown in figure 6.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Figure 6. Example of react component

Furthermore, React is an open-source library and it is created and supported by Facebook. React manipulates the DOM efficiently and which then combined with JavaScript's speed and efficiency helps to render web pages faster and to create dynamic and responsive web pages. (David Herbert, 2022). In traditional JavaScript application, it requires manual DOM manipulation to reflect any changes to the data, which needs finding the changed data and updating the DOM which results in full page reload. React's single-page application (SPA) helps to overcome this by loading only single HTML document on the first request, and then updating a particular part or section of the webpage by using JavaScript.

React uses virtual DOM, which is a copy of actual DOM, and whenever there is a change in state, React's virtual DOM is quickly reloaded. React compares virtual and actual DOM to figure out what has changed and then without rendering the DOM, React uses least expensive way to patch the DOM with the update. So, if something is updated the whole page does not need to be reloaded, and React's UI and components quickly reflect the change.

2.3 React hooks

React hooks were introduced in React version 16.8 to allow developers to use features of the React library like lifecycle methods, state, and context in functional components (Meta Platforms, Inc., 2022a).

React hooks help to solve the problems caused by writing and supporting many components. React did not offer a way to attach reusable behavior to a component and higher order components and render props required components to be restructured which could make code harder to follow.

React hooks extract stateful logic from a component which allows it to be tested independently and be reused without changing the component hierarchy. So, React hooks can be shared among many components or community. Hooks also let components be split into smaller functions based on their goals.

To solve this, Hooks let you split one component into smaller functions based on what pieces are related (such as setting up a subscription or fetching data), rather than forcing a split based on lifecycle methods. You may also opt to manage the component's local state with a reducer to make it more predictable components (Meta Platforms, Inc., 2022a).

1. UseContext Hook

React has unidirectional data flow, meaning the data can only be passed from parent to child. The data needs to be passed manually as props, and depending on the level of the child, it needs to be passed through all the levels. To tackle this, a React hook called use Context has been introduced, which provides data which is accessible to all the components including deeply nested components.

2. UseState hook

The use of State Hook allows you to create, update, and manipulate the state inside functional components.

React has this concept of state, which are variables that hold data that components depend on and may change over time. Whenever these variables change, React updates the UI by re-rendering the component in the DOM with the current values of the state variables (Ikechukwu, 2022).

This hook takes a single argument and is optional: an initial value for the state. Then it returns an array of two values:

- The state variable
- A function to update the state

3. UseEffect hook

React hook defines "componentDidMount", "componentDidUpdate", and "componentWillUnmount" lifecycle methods of previously used React class and combines them all in one function. In functional components this hook lets React's lifecycle methods to replicate.

Various kinds of actions such as fetching the data, different external API interactions and changing state variables are called side effects. Side effects can run alongside the main

operation of a component and this hook lets you use side effects in function components. The use Effect hook accepts two arguments:

- A function where the code runs.
- An array that has a list of values from the component scope (props, context, and state variables), known as a dependency array (Ikechukwu, 2022). Every time the array's value is updated it tells the Hook to run. If no array elements are provided, then the Hook will run after every render.

4. UseReducer hook.

This hook is also an alternative to the "useState hook" because as like the "useState", this hook lets you create state-like variables, and upon their change they cause the UI change. If the state and state updates involve multiple sub-values, this hook allows more complex logic.

This Hook accepts two arguments: a reducer function and an initial state, i.e., use Reducer (reducer, initial State).

It returns an array of two values which can be restructured to the current value of the state and a dispatch function (Ikechukwu, 2022).

2.4 React User Interface (UI) component Libraries

As React is an open-source JavaScript library, it can easily combine with other JavaScript frameworks and libraries, including components. A component is a reusable bit of code. Due to high modularity the React component libraries gives flexibility and offer high degree of control while customizing. Since the developers do not have to write code from scratch, the development process is quick and most of the UI frameworks are responsive by nature. Some of the famous React components libraries in 2022 are Material UI (MUI), Ant design and React-Bootstrap.

3 Creating and Implementing the navigation

The new navigation has four critical parts, Sidebar, Favorites menu, Top Navigation bar and Footer. A navigation folder is created inside the existing React repository and all the navigation components are created inside that folder.

In this part I am going to discuss how navigation elements are created and are processed to different navigation components.

3.1 User customization

The user of the navigation must be able to find all the FCP elements in the navigation, and they can customize the elements in the navigation to show and hide items according to their choice. For explanation and demonstration purposes, the UX designer has presented a mock-up of the new navigation created using Figma.

The design requirements are as follows:

- A sidebar menu with Main navigation menu, Sub navigation menu, and Favorites menu.
- A Favorites menu editor which gives option to add, edit and change the Navigation Item’s order in Favorites menu.
- Top Navigation bar which has functionalities to change site, a search button which opens a search bar, user menu and a user guide.
- A Footer.

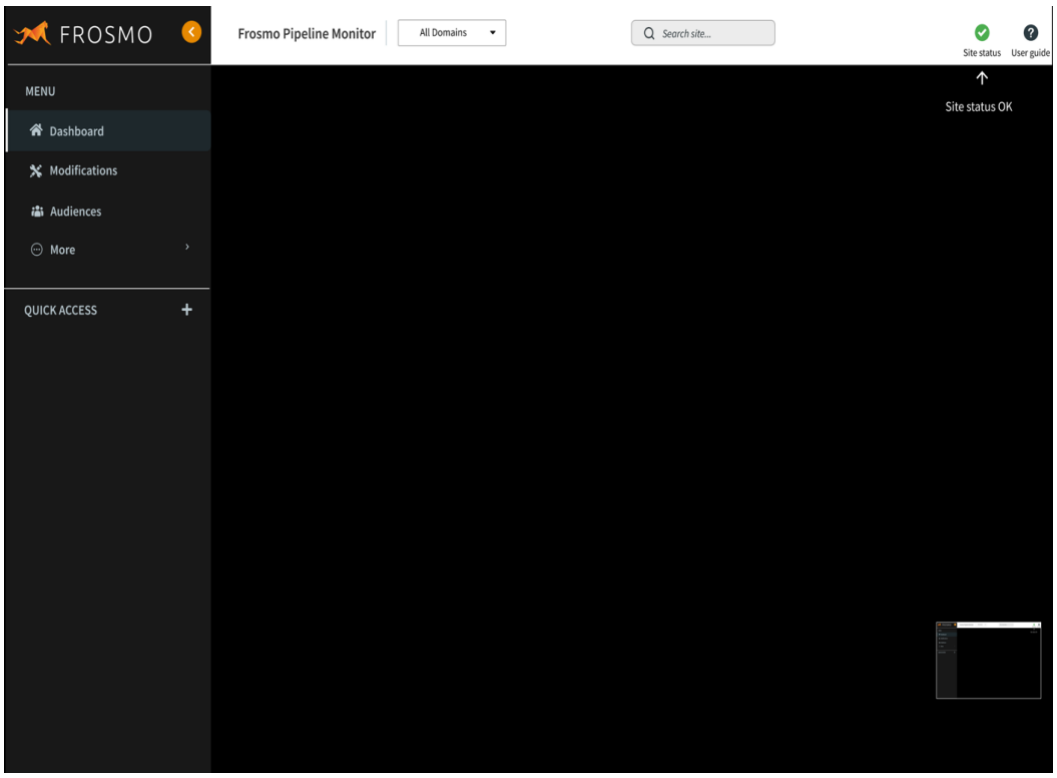

Figure 7. Mockup of the new navigation

3.2 Navigation elements

Before starting to create navigation components, it is important to figure out what the elements in different navigation components are going to be. The data structure of the elements is created and processed to components in the same place. And in this way, future changes in Navigation Items can be made by modifying only one file. In this section I am going to discuss how navigation links are created, processed, and rendered in different components.

3.2.1 Navigation Item

Navigation Item is a link to a FCP page. Besides being a link element in the Navigation, the item must have various attributes about the link such as its title, icon, category, and others. And for this purpose, Navigation Item object is created with all the necessary attributes as its properties.

Different attributes of the Navigation Item are listed below:

- “title” is the name of the Navigation Item.

- “id” is unique for each item. It should be defined in a way that if item's title is “Modifications”, then “id” for that item should be "ITEM_MODIFICATIONS".
- There are four available categories: Features, Statistics, Tools, Data tracking. And the item should have a category depending on the type of the item.
- “to” is the link of the Navigation Item, for instance, for Navigation Item Modifications, should be "/mahi/message#/list".
- “quickAccess” is a “Boolean” and it should be given value “true” or “flase”, depending on if the item can be shown in Favorites menu.
- “icon”, which is the item's icon, i.e., for “Modifications” Navigation Item’s icon is “fa fa-pencil-square-o “. A distinct icon is chosen for Navigation Item.
- *“*RequiredSettings” is optional. It is only needed if the given Navigation Item is not commonly visible for all the FCP users and requires certain “add-ons” or settings to be enabled first. It is divided into two types: **“**“add-ons”” and “siteSettings”.

Different Navigation Items might require various kinds of settings or “add-ons” to be enabled before it is visible in FCP to the user. For example, some Navigation Items are features or “add-ons” which are only available to paid customers. So, each Navigation Item must be checked, if the user has required permission to access it before it is shown in different navigation components.

The Navigation Items should contain the information about what settings or “add-ons” it requires. And it will be checked if the Navigation Items can be shown or is hidden by checking the required settings of Navigation Items with the settings variables created using “useContext” hooks. The “requiredSettings” property and how its elements are allocated for each Navigation Item is listed below:

- If only one “add-on” is needed it is put in as string. Else, it is array of strings, and first element of the array is "or" or "and" string. Other elements of the array are string which are the name of the addons needed for the Navigation Item.
- If only one “siteSettings” is needed, it is a variable from “useContext”, and the variable has “Boolean” value. Else, it is array of variables from “useContext” hook, with “Boolean” values and first element of the array is a string with either "or" or "and" as its value. For example, Affinity groups Navigation Item requires that the site has Affinity groups enabled. And “useSite” context variable has “affinityGroups**”**, and this variable will be used in “siteSettings” of the Affinity groups Navigation Item.

- If only one of either ““add-ons”” or “siteSetting” is required, then the “condition” property is not needed. Otherwise, depending on the settings required the “condition” is string and value is either “or” or “and”.

- “subNav” is needed when the item has Page Navigation Menu in the page. For e.g., Recommendations have Strategies and Configurations in as “subNav”,

-

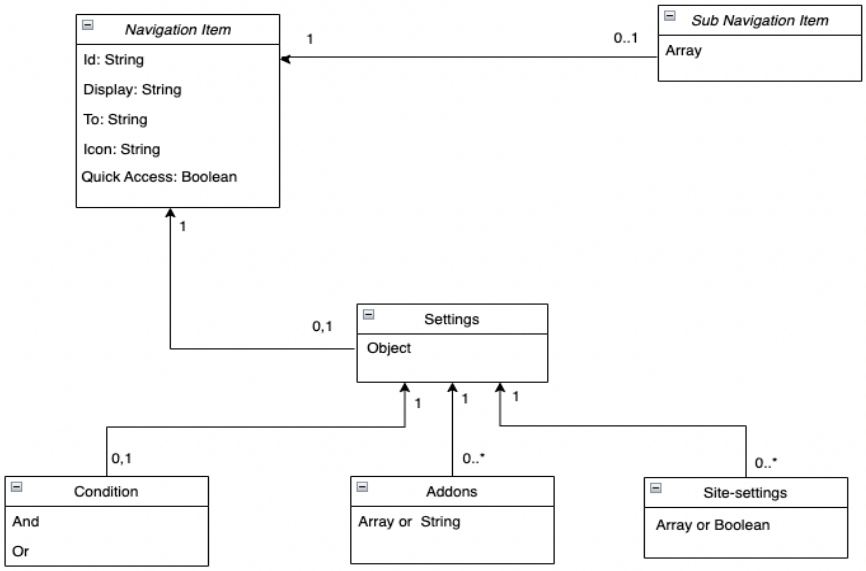

Figure 8. Uml diagram of Navigation Item

```
export interface NavigationItems {
    [id: string]: NavigationItem;
}

export interface NavigationItem {
    id: string;
    display: string;
    icon: string;
    to: string;
    category?: string;
    subNav?: string[];
    requiredSettings?: RequiredSettings;
    quickAccess?: boolean;
}

export enum CONDITION {
    OR =  "or",
    AND =  "and"
}

export interface RequiredSettings  {
    addons?: string | string [];
    condition?: CONDITION;
    siteSettings?: boolean | (string|boolean)[];
} []
```

Figure 9. Preview of different interfaces and enum in Navigation.tsx file

3.2.2 All Navigation Items List

Sidebar, Sub Navigation and Favorites menu all contains Navigation Items. For swiftly positioning and editing the items inside different menus, Navigation Items are placed together in a list known as “AllNavigationItems” (check Appendix 2). The list can be found in “NavigationContext.tsx” file in “/application/assets/react/ui/src/contexts/NavigationContext”.

To add a new Navigation Item to All Navigation Items list, all the attributes of the Navigation Items should be known beforehand. After that, the Navigation Item is manually added to all Navigation Items list. Since all navigation list is an object, a new element is added by having element with its iid “property” as property and Navigation Item as key. An example of Navigation Item element inside all navigation menu is shown in figure 10.

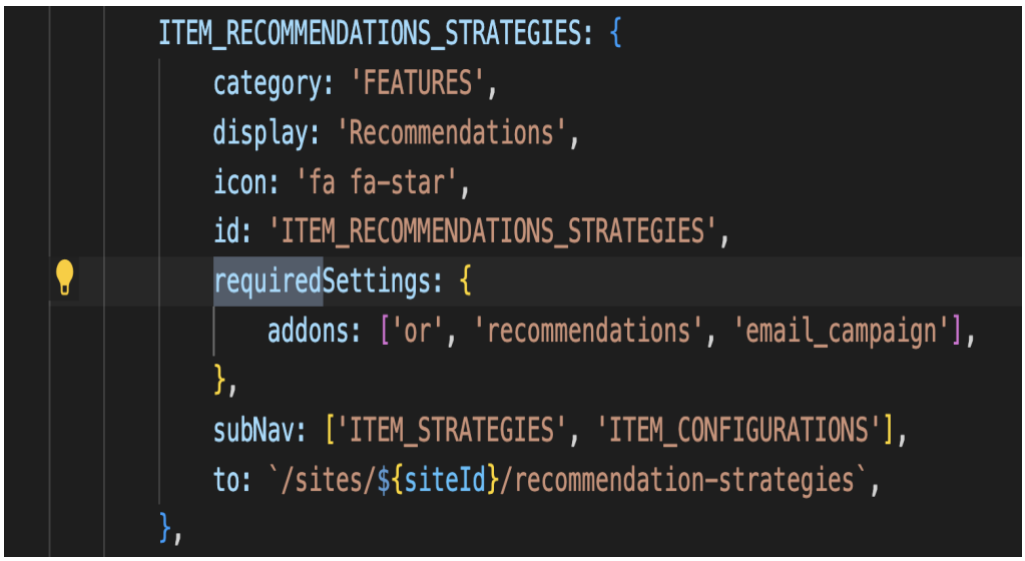

Figure 10. Example of a Navigation Item Recommendation Strategies in All Navigation Item list

After the Navigation Item is added to all Navigation Items list. The item is ready to be processed to components.

3.2.3 Main Navigation Items

Since the Main Navigation Items and Sub Navigation Items are not customizable by users an, the main Navigation Items and Sub Navigation Items are stored in a variable so that data can be passed further to the Main navigation and Sub navigation components.

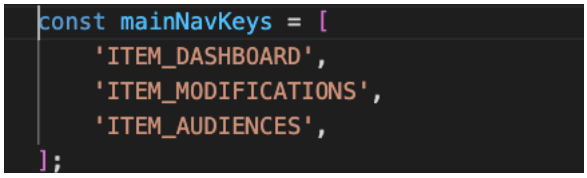

Figure 11. mainNavKey variable

```
const subNavKeys = [
    'ITEM_CONVERSION_DEFINITIONS',
    'ITEM_CUSTOM_ACTIONS',
    'ITEM_GOOGLE_ANALYTICS',
    'ITEM_PRODUCTS',
    'ITEM_EMAIL_RECOMMENDATIONS',
    'ITEM_RECOMMENDATIONS_STRATEGIES',
    'ITEM_SHARED_CODE',
    'ITEM_TRIGGERS',
    'ITEM_ALL_REPORTS',
    'ITEM_ANNOTATIONS',
    'ITEM_CUSTOM_EXTENSIONS',
    'ITEM_WORKSPACES',
];
```

Figure 12. subNavKeys variable

3.2.4 Required Settings

Each Navigation Item from All Navigation Items list are checked if they consist required settings before the Navigation Items are passed to components. And for that, the items in all navigation lists are filtered using the function “evalSettings”. The information of all the required settings for the site, company, “add-ons”, users are stored and passed in as context variables. Furthermore, if a Navigation Item requires settings, features or some other different settings enabled, then it must be checked to the context varables.

```
const filteredNavigationItemsList = Object.keys(allNavigationItems)
    .map((navItem) => {
        return allNavigationItems[navItem];
    })
    .filter((item) => {
        let filteredItem = item;
        if (item.subNav) {
            const filteredSubnav = item.subNav.filter((item) =>
                evalSettings(
                    allNavigationItems[item].requiredSettings,
                    addons
                )
            );
            filteredItem.subNav = filteredSubnav;
        }

        return evalSettings(item.requiredSettings, addons);
    });

const filteredNavigationItems = Object.fromEntries(
    filteredNavigationItemsList.map((navItem) => [navItem.id, navItem])
);
```

Figure 13. Filtering all Navigation Items based on the required settings

3.2.5 Checking addons

Required “add-ons” for the site are checked using function “checkAddonsValue”. It checks if the given Navigation Item in the parameter has required “add-ons” and returns a “Boolean” value. This function returns “false” in following cases:

- If the Navigation Item has ““add-ons”” in required settings but there are no required “add-ons” provided in “siteContext” context variable.
- If the required “add-ons” is a string and it is not in the “add-ons” of “siteContext” context variable which is provided as parameter.
- If the required addons is an array and the first element of array is “or”, and if none of the required “add-ons” is in the “add-ons” of “siteContext” context variable which is provided as parameter.
- If the required “add-ons” is an array and the first element of array is “and”, and if one of the items from the required items are not in the “add-ons” of “siteContext” variable which is provided as parameter.

Otherwise, the function will return “true”. Implementation of the function is shown in figure 14.

```
const checkAddonsValue = () => {
    let match: boolean;
    // If no addons provided then return false
    if (!addons) {
        return false;
    }
    const requiredAddons = settings.addons as string | string[];
    // If addons required is array then
    // check what if conditionis provided
    // in the first element of the array
    if (!isArray(requiredAddons)) {
        match = addons.includes(requiredAddons);
        return match;
    }
    const condition = requiredAddons[0];
    if (condition === 'or') {
        match = addons.some((val: string) => requiredAddons.includes(val));
        return match;
    }
    match = requiredAddons.every((val) => addons.includes(val));
    return match;
};
```

Figure 14. Preview of function checkAddonsValue

3.2.6 Checking site settings

Required site settings for the site is checked using function “checkSiteSettingsValue”. The function checks if the given Navigation Item in the parameter requires any kind of settings

for the site to be shown in FCP and returns a “Boolean” value. If the return value is “true” then the item will be processed further to be used in navigation components and if the returned value is “false”, the navigation will not be shown in the navigation. This function returns “false” in following cases.

- If all the required “site-settings” is “Boolean” and value is “false”.
- If the required “site-settings” is an array and the first element of the array is “or”, and if none of elements in the array has value as “true”.
- If the required “site-settings” is an array and the first element of array is “and”, and if one of the elements in the array has value as “false".

Otherwise, the function will return “true”. The implementation of the function is shown in Figure 15.

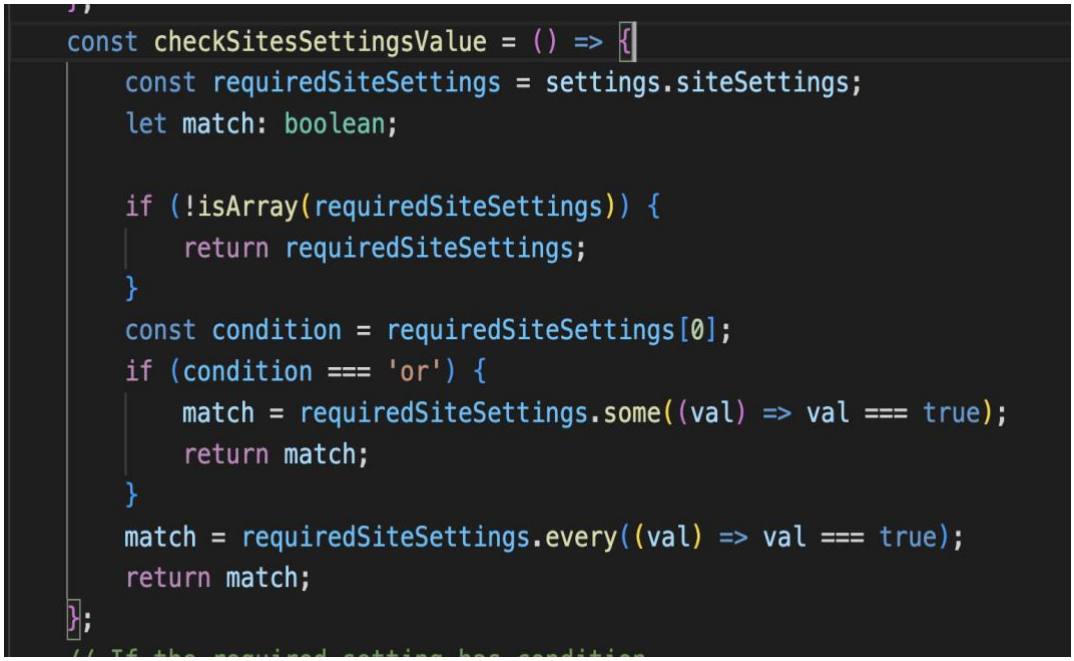

Figure 15. Preview of Function checkSiteSettingsValue

The “evalValue” function returns a “Boolean” value and returns “true” in these cases:

- If the “requiredSettings” property have condition and the condition value is “OR”, and one of the functions “checkAddonsValue” or “checkSiteSettings” returns “true”.
- If the requires settings have condition and the condition value is “AND”, and both function “checkAddonsValue” and “checkSiteSettings” returns “true”.
- If only “addOns” is provided and it returns true.
- If only “siteSettings” property is provided, and it returns “true".

- if a Navigation Item does not have any required settings.

In other cases, the function will return “false”.

```
// If the required setting has condition
// then check the condition with required addons and required settings
if (settings.condition && settings.addons && settings.siteSettings) {
    if (settings.condition === 'or') {
        value =
            checkAddonsValue() || checkSitesSettingsValue() ? true : false;
        return value;
    }
    value = checkAddonsValue() && checkSitesSettingsValue() ? true : false;
    return value;
}
// If no condition is provided
// then required settings is either addons or settings
if (settings.addons) {
    value = checkAddonsValue();
    return value;
}
if (settings.siteSettings) {
    value = checkSitesSettingsValue();
    return value;
}
return value;
```

Figure 16. Checking various conditions in evalSettings function

3.3 Sidebar

Sidebar navigation is shown in the left-hand side of the FCP (Frosmo Control Panel). It is the main part of the navigation where users can find most of the navigation links that they require to navigate in FCP.

In addition, the sidebar menu is dynamic, and its length can be widened and shortened by clicking the orange arrow button on the top right side of the sidebar. The widened sidebar shows the name and the icon of the Navigation Item, and the shortened sidebar shows only the icon of the Navigation Item alongside the tooltip which includes the name of the Navigation Item. User can navigate to each page by clicking the Navigation Item and upon clicking the Navigation Item, the item is set as active in the navigation by highlighting the Navigation Item. It consists of three parts:

- Main navigation menu
- Sub navigation menu
- Favorites menu

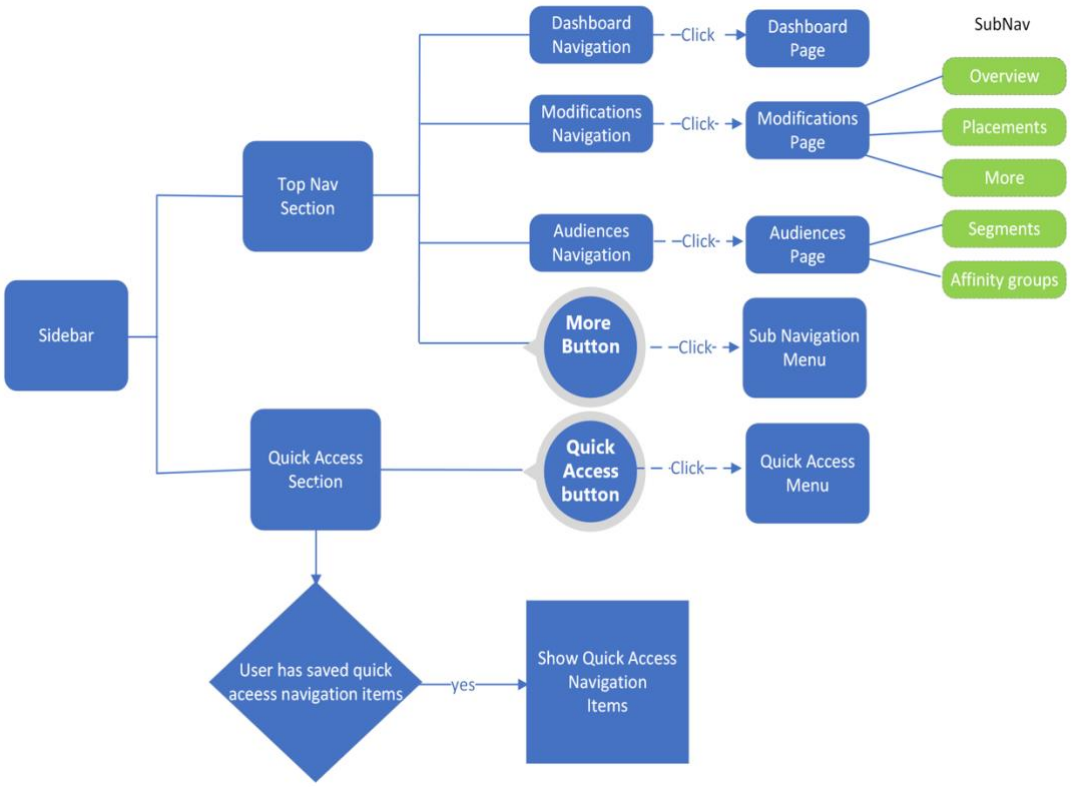

Figure 17. Uml diagram of the Sidebar

3.4 Main navigation menu

Main navigation menu lies on the top half of the sidebar menu and consists of Navigation Items that are often used by the users in FCP, and it is important that they are easily accessible. This part of the Sidebar navigation is not customizable, and the Navigation Items here are visible to all the users. It is composed of four Navigation Items:

- Dashboard
- Modifications
- Audiences
- More

“Dashboard” is the home page or where user lands up after logging in FCP. “Modifications” and “Audiences” open links to one of the common most notable features of FCP. Upon clicking the Navigation Items, the page is redirected to the respected link and the Navigation Item is highlighted.

Unlike others in the main navigation, “More” is a button and on clicking the button “Sub navigation menu” opens, which consists of more links. Upon clicking the links in the “Sub navigation menu” the link item is highlighted, and menu closes simultaneously.

“More” button will not be highlighted under the following conditions:

- If any item from the Main navigation menu is clicked.
- If the active Navigation Item doesn’t belong to the Sub navigation menu and Sub navigation menu is closed.

The links in the Sub navigation Items are highlighted in other different cases as follows:

- If the Sub navigation menu is open.
- If any Navigation Items from the Sub navigation menu is clicked.
- More buttons will be highlighted if a Navigation Item in the Sub navigation is clicked.
- If a Navigation Item clicked is Page Navigation and if it is page navigation of an item that belongs to Sub navigation menu, then the More button is highlighted.
- If an item from the Favorites menu is clicked and if Sub navigation menu includes the item.

3.4.1 Sub navigation menu

Sub navigation menu is part of the sidebar and opens after clicking the More button in Main Navigation and it appears on the right side of the side bar.

The Navigation Items that are in the Sub Navigation Items are fetched from use Navigation context where the items are passed in “subNavItems” variable. The items are further categorized according to their categories and items in distinct categories are mapped together and render in the Sub navigation menu as a link. If any of the links inside the menu is clicked, then the link is highlighted, and the menu gets closed automatically. For closing the menu, there is also a close button at the top right side.

3.4.2 Favorites menu

It is the personalized, customizable section of the sidebar where users could add the Navigation Items as per their preferences. Moreover, all Navigation Items than the ones in the Main navigation menu can be added to Favorites menu. Users can customize the Favorites menu, meaning users could be able to add, remove, and change the order of the Navigation Items. And the selected Navigation Items will be available to be accessed from the Favorites menu of sidebar.

While saving the Navigation Items to Favorites menu, a “POST” request is sent to “graniitti API/quick-access” endpoint to save the Navigation Items. And while user logs into FCP, the GET request is made to fetch the saved Navigation Items as shown in figure 18. The saved favorites items are a list of ids of the Navigation Item saved in Favorites menu, and the fetched data is the list of favorites item’s id. The favorite Navigation Items are processed to Favorites menu to render.

```
const favouritesMenuUrl = `/users/${userId}/storage/quick-access`;
const [result, setUrl] = useGraniittiAPI('', []);
const favouritesItemsData = result.data;
useEffect(() => {
    setUrl(favouritesMenuUrl);
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [favouritesMenuUrl]);
const [favouritesMenu, setFavouritesMenu] = useState<NavigationItem[]>([]);

useEffect(() => {
    if (!favouritesItemsData || favouritesItemsData.length === 0) {
        return;
    }
    // Check if the item exists in all navigation list.
    const filteredNavList = favouritesItemsData
        .filter((navId: string) => allNavigationItems[navId])
        .map((navId: string) => allNavigationItems[navId]);

    if (filteredNavList.length === 0) {
        return;
    }
    setFavouritesMenu(filteredNavList);
}, [allNavigationItems, favouritesItemsData]);
```

Figure 18. Code showing how the favorite Navigation Items are fetched from back-end

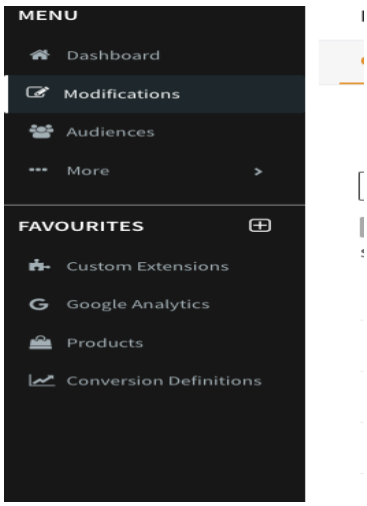

Figure 19. Navigation Items in Favorites menu in the Sidebar

3.4.3 Favorites menu editor

Favorites Menu editor opens after clicking the plus icon next to the “Favorites” text in sidebar. Main purpose of this editor is to add, remove, and change the order of the Navigation Items in the Favorites menu.

As shown in the figure 20, In the right-hand side of the Favorites menu editor all the Navigation Items which can be added to Favorites menu are listed. Furthermore, Navigation Items are categorized according to their types, for instance, Features, Data Tracking, Tools, and Statistics. On clicking a category, Navigation Items which have the category will be shown. And search options on the top-right hand side provides an option to find Navigation Items by their title.

In the left-hand side of the editor, the list of selected Favorites Navigation Items is shown. User has the flexibility to move the items up or down by clicking the respective button represented by arrow up and down located next to the title of each Navigation Item. Also, users can remove the selected Navigation Items by clicking the remove icon next to the Navigation Item’s title.

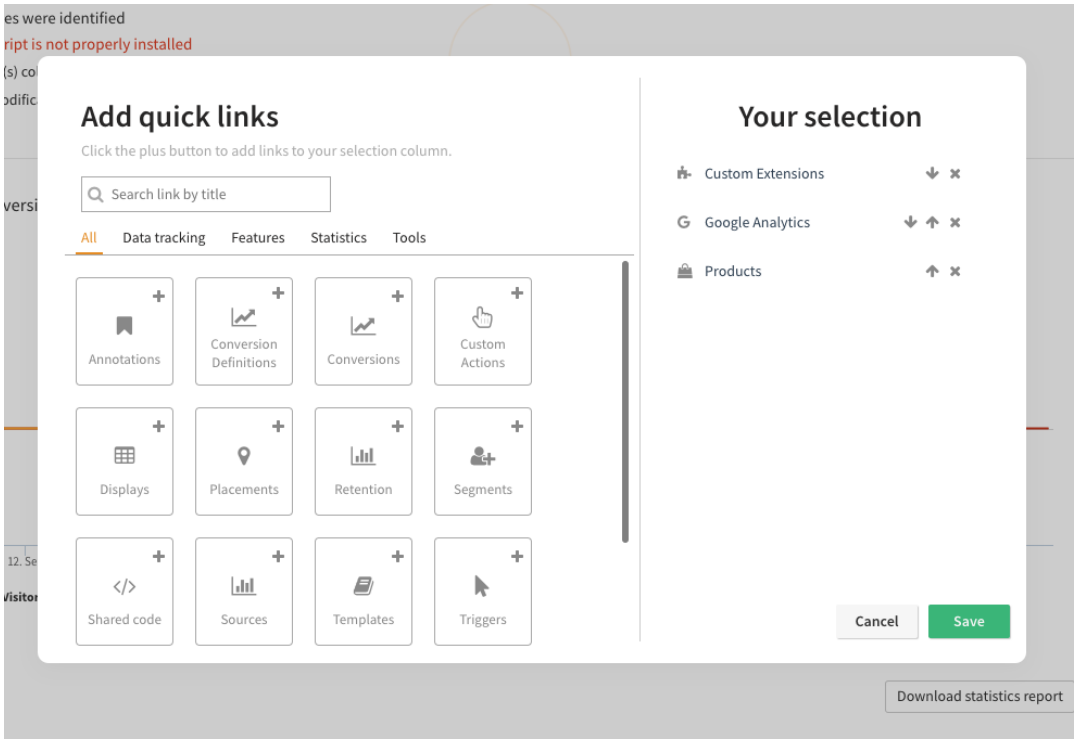

Figure 20. Favorites menu

The useReducer hook is used to add, remove, and move the items of the selected Favorites menu as shown in figure 21. Favorites menu items list is saved as initial state. With every action the state changes. For adding, a value is added at the end of the state. For removing an item, the selected item is filtered out. While moving items up and down in the list, the change in state takes place outside of the reducer and the modified state is retrieved through “payload.action.newState”.

```
function reducer(state: NavigationItem[], action: any) {
    switch (action.type) {
        case ACTION.ADD: {
            const value = [action.payload.item];
            return [...state, ...value];
        }
        case ACTION.REMOVE: {
            const newState = (state as NavigationItem[]).filter(
                (subMenu) => subMenu.id !== action.payload
            );
            return [...newState];
        }
        case ACTION.MOVE_ITEM_UP:
        case ACTION.MOVE_ITEM_DOWN:
        case ACTION.SAVE:
            return [...action.payload.newState];
        default:
            return state;
    }
}
```

Figure 21. Use of UseReducer hook to change the state of quick access menu items

3.5 Top Navigation bar

Top Navigation is a part of the navigation and is the topmost element in FCP. Top Navigation is fixed, and it contains varieties of elements ranging from: company name, sites menu, a search button, icon showing site status, user guide and user menu. Top Navigation is responsive and when the Side bar and Sub menu is open and closed the Top Navigation changes its width dynamically.

3.5.1 Sites Menu

In the Top Bar of the navigation, the site's information is shown and next to it will be a button which will open a drop-down menu with a list of sites that belong to the company. Users can change to any site by clicking the site, and the site in the FCP is updated and the page reloads with new site’s information. With this user gets the flexibility to check the available sites and give them the option to change the sites from anywhere in the FCP.

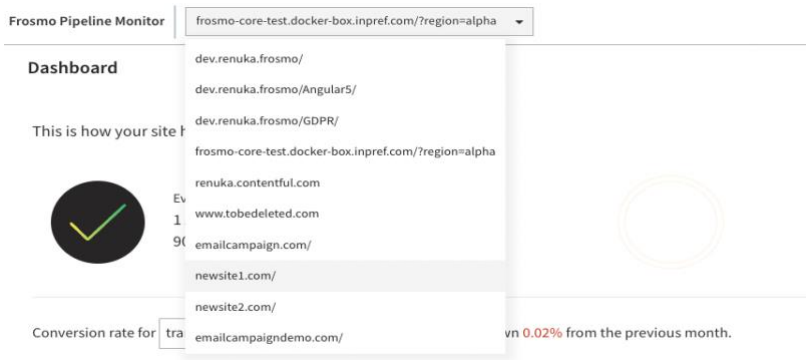

Figure 22. Overview of drop-down sites Menu in Top navigation bar

3.5.2 Site setup status

Since FCP is used to customize sites and check their performances, it is important that the required Frosmo setup is installed and working properly. Through FCP users can check if Frosmo's back end is receiving various kinds of tracking data from the site regularly. And users need to go to site-status page to check the status of the site setup and with the new navigation users could check site setup status directly from navigation. Moreover, different site setup statuses in a site are as follows:

- No setup done yet.
- Setup done and working (is marked by green checkmark).
- Setup done and not working (is marked by red exclamation error).

No site setup status not an error, but everything is not working either as the site has not been set up correctly. In that case, the problem for the issue can be checked and figured out what is not working properly. Below are the definitions for different cases.

1. No site setup status (No setup done yet):

- Frosmo's back end has not received any API calls from the site within thirty days *OR*
- Frosmo's back end has not received product views (meaning if any product in the site has been viewed) within thirty days *OR*
- Frosmo's back end has not received transactions within thirty days *OR*
- Frosmo's back end has no product attribute info from the site, meaning there is no information about any product.

2. OK status:

- Frosmo's back end has received any API calls from the site within twenty-four hours *AND*

- Frosmo’s back end has received product views within two hours, meaning any product has been viewed on the site *AND*
- Frosmo’s back end has received transactions within two hours *AND*
- Product attributes are available and have been updated within seven days.

3. ERROR status:
- Frosmo’s back end has received any API calls within the last thirty days but not within twenty-four hours *OR*
- Frosmo’s back end has received product views from the site within thirty days but not within two hours *OR*
- Frosmo’s back end has received transactions from the site within thirty days but not within two hours *OR*
- Product attribute info exists but has not been updated within seven days.
- We have received conversions within thirty days but not within two hours.

3.5.3 User guide menu and User Settings Menu

The User guide menu opens after clicking the “Help” text or the “?” icon above the text. The user guide menu contains links to company’s documentation such as frequently asked questions, glossary, user guides, developer guides and release information.

The User settings menu opens after clicking the user’s name or the icon on the right side of the Top Navigation bar. The user menu contains links to the user’s account, different API’s, user activities, company, users, and a log out option. API access link is only available to the users with extended permissions meaning it is available for certain users only.

3.6 Footer

Even though a new footer component in React has been created, it could not be implemented. Because of the height of the page content of a page written in code other than React could not be decided beforehand and for that Footer could not be positioned properly. So, the new Footer will not be implemented as part of the new navigation.

3.7 Changing page content's styles

In FCP there are pages written in PHP, Angular, Angular 2 and since the new navigation needs to work in all the pages. In pages written other than react, the page content's styles need to change dynamically when the style of navigation changes, i.e., if Sidebar navigation is opened or closed and if the Sub navigation menu is open, then the page content should change its width and have margin in the left accordingly. A React component is created to modify the styles of the page content of pages written in code other than React.

- getElementById - This method of the document object returns an Element object standing for the element whose id property matches the specified string. They are useful ways to quickly access a specific element as the element IDs must be unique if specified. Since element IDs must be unique if specified, they are a useful way to get access to a specific element quickly (Mozilla, 2022b).

- createElement - It creates the HTML element specified by "tagName", or an "HTMLUnknownElement" if "tagName" is not recognized in an HTML document (Mozilla, 2022a).

- appendChild - The appendChild method appends a node (element) as the last child of an element (W3Schools, 2022).

- addEventListener – Whenever specified event takes place, the addEventListener method of the EventTarget interface calls a function, which the method has set up (Mozilla, 2022c).

For the pages written in code other than React to work with navigation written in React code, the root html element of the non-react code is fetched using "getElementById" and assigned to variable "styleEl". A function called "handleStyle" is created which checks different conditions to provide styles for page content in pages written in code other than react as shown in Appendix 3. Style changes are related to adjusting the page content's size when the sidebar's width changes, sub navigation is opened/closed, or when the window's size changes.

A resize event handler is set on window's object inside "useEffect" hook's function. Alongside the function "handleStyle" is also called. The changes in sidebar and Sub navigation menu are checked through "isSideBarOpen" and "isSubMenuOpen" variables

and they are passed in the array of “useEffect”, which means that the function in “useEffect” will be called every time there are changes in sidebar’s width or if the Sub navigation menu is opened or closed.

```
let styleEl: HTMLElement | null = document.getElementById(
    'nonreact-content-style'
);
const wrapper: HTMLElement | null = document.getElementById('main-wrapper');
useEffect(() => {
    window.addEventListener('resize', handleStyle);
    handleStyle();
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [isSideBarMenuOpen, isSubMenuOpen]);
```

Figure 23. Overview of the variable styleEl and use Effect hook

4 Deploying in alpha site

After creating all the navigation components, new navigation was deployed to alpha site for testing purposes. Various kinds of tests were carried out by stakeholders and Frosmo Oy's product team members. Deploying the navigation to alpha site before deploying to production sites helps to perform different kind of functional and usability testing in a production like environment and it is easier to fix bugs and usability issues.

4.1 UI/UX Testing

Functional testing is a software testing process that checks if the application feature works as per the requirements, and it includes all testing techniques and methods that are used to test the behavior of the object and its input/output. Functional testing is built using the "black-box" technique, where test cases are derived from specifications (Andreas Spillner, 2021, s. Chapter 3).

Each function is compared and matched to the corresponding requirements and checked if the output is consistent with the end user's expectation or specifications. Testing is carried out by providing sample inputs and recording resulting outputs and varying if the outputs are the same as expected outputs. Unlike on-functional testing, functional testing focuses on the result than investigating speed, scalability, and reliability, and figures out whether the application satisfies smallest user expectations (Micro Focus, 2022).

4.2 UI/UX Testing

In an application, UI/UX testing evaluates the graphical user interface. The performance of UI components such as menus, buttons, text fields and more are verified to ensure that the user experience is ideal for the application's users. UI/UX testing is also known as visual testing and can be manual or automated (Shain, 2022).

Table 1. UI/UX Testing of Sidebar Menu

Description	Expected Result	Output Result	Status
Click or hover on any Navigation Item.	The Navigation Item is highlighted.	The Navigation Item is highlighted.	Pass.
Click “More” → select any of the items that belong to the submenu	Sub menu closes. The “More/Dot” icon is highlighted The clicked item is highlighted in the “Submenu”	Sub menu closes. The “More/dot icon” is highlighted The clicked item is highlighted in the “Submenu”	Pass
Click “the plus” button next to the Favorites.	Favorites menu editor opens.	Favorites menu editor opens.	Pass
Search for item “Displays” in the search bar.	Only “Displays” items are shown from the list of Navigation Items.	Only “Displays” items are shown from the list of Navigation Items.	Pass
Click the searched item and save the item to selected Favorites menu.	Navigation Items appears in the selected menu on the right-hand side of the editor.	Navigation Items appears in the selected menu on the right-hand side of the editor.	Pass
Click the save button at the bottom-right corner of Favorites menu editor.	Navigation Item appears in the Favorites menu section in sidebar and Favorites menu editor is closed.	Navigation Item appears in the Favorites menu section in sidebar and Favorites menu editor is closed.	Pass

Table 2. UI/UX Testing of Top bar

Description	Expected Result	Output Result	Status
Go to reco pipeline page 2115 through search bar and open the site check if company name is showing properly on the left-hand side of Top bar.	The company name is shown correctly.	The company name is shown correctly.	Pass.
Open site selection by clicking the site selection box and click any site.	Site selection menu opens. While clicking on a site, the URL is redirected to the homepage of the clicked site.	Site selection menu opens. While clicking on a site, the URL is redirected to the homepage of the clicked site.	Pass
Click the plus button next to the Favorites.	Favorites menu editor opens	Favorites menu editor opens	Pass
Click plus button when Favorites menu editor is open.	Favorites menu editor closes.	Favorites menu editor opens	Pass

4.3 User Accepting Testing

Usability testing involves testing application with certain users in a production or production like environment. The feedback from these live users – who have no prior experience with the application and may discover critical bugs that were unknown to internal teams – is used to make further changes to the application before a full launch (Shain, 2022).

And since the navigation is available in production like environment, i.e., alpha, the other members of product team are asked to do the user accepting and provide any errors, feedback from the users.

4.4 Smoke testing

Smoke testing is a software testing process to check the stability of the built software. In simple terms, smoke tests mean verifying that the features are working and there are not any significant issues with the functionality in the software. It creates a way for further testing and decides if it is worth using more time and resources in further testing (Hamilton, 2022).

Smoke testing of the new navigation is being carried out by all the members of the product team in Frosmo Oy. Any bug in the application is reported and is addressed swiftly and is fixed before the Navigation is released to the production.

5 Discussion

By the time the research paper is completed, the react-based navigation is released to production. Most of the bugs have been already detected in the testing phases also the functionalities and usability has been checked through many times, so the chances of new bugs appearing is relatively low. Any detection of the bugs will come through Frosmo's support handle team and Frosmo's product team will address the issue according to the status and need of the issue.

The React based navigation allows users to customize the navigation and personalize their preferences in the FCP. All the page elements are included in the navigation so that the user can visit any page they want within a couple of clicks. The navigation has a significant impact on the user's flexibility in FCP as user can swiftly navigate around FCP. Also, since the navigation elements for different navigation components are processed at same place, any changes to the navigation can be done by modifying one file, which could be beneficial for the future when there is a need to modify the navigation components.

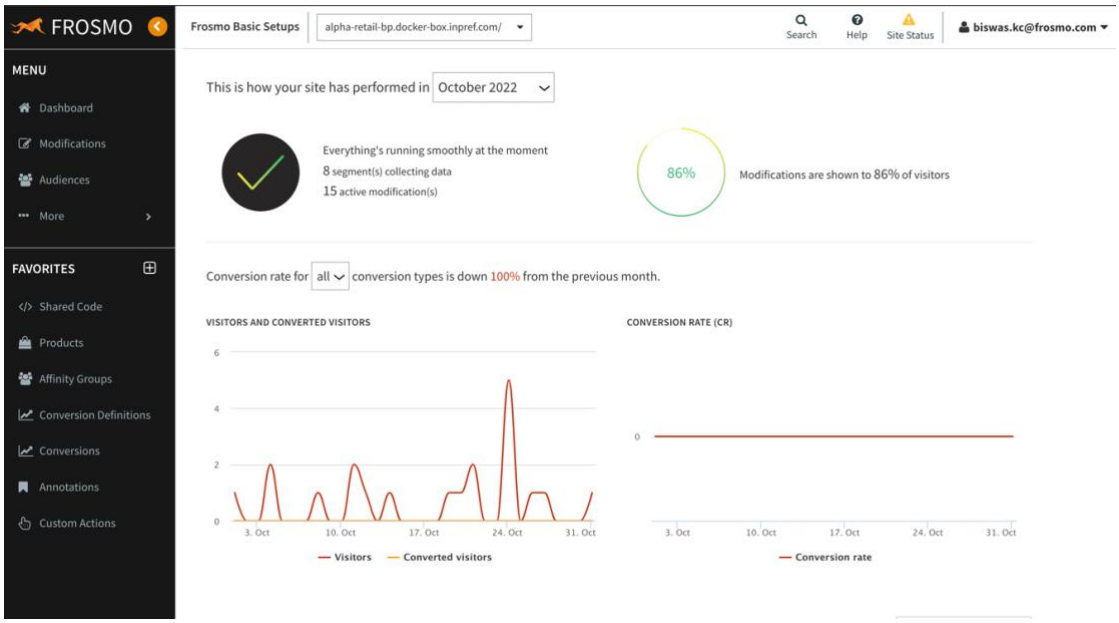

Figure 24. Overview of Side bar and Top bar navigation menu

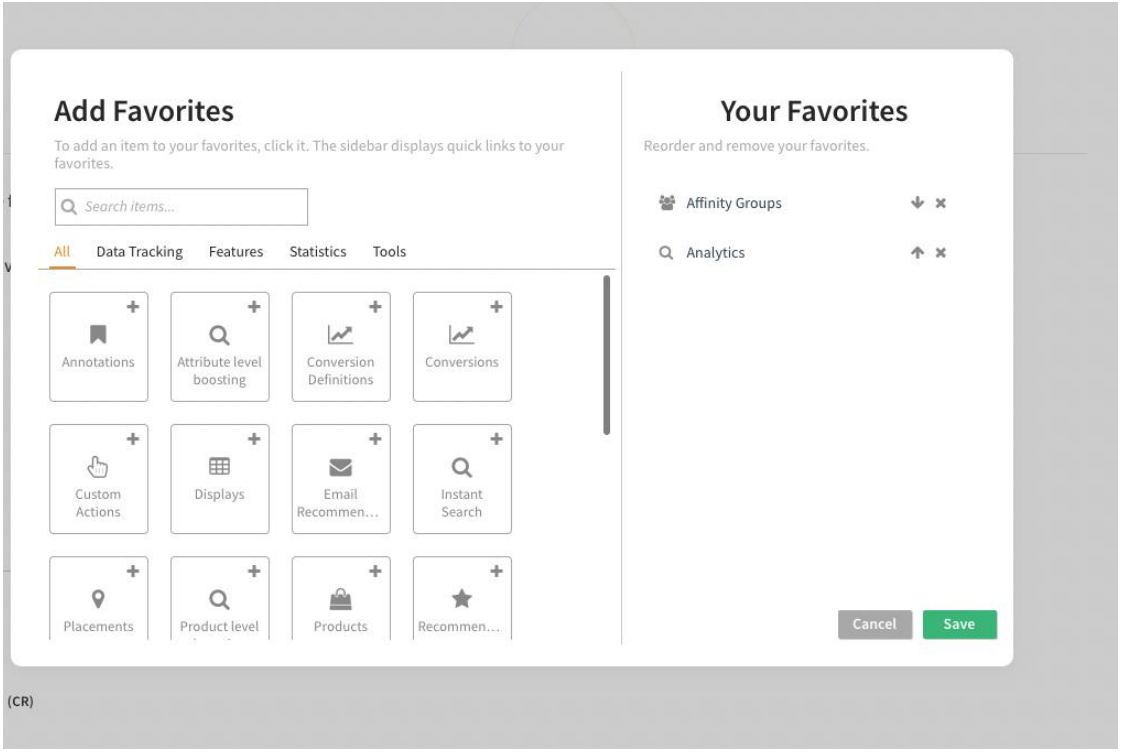

Figure 25. Overview of Favorites menu editor

5.1 User review and analysis

At the time this research paper is completed, the new navigation has been implemented to production sites for couple of weeks. Initial feedbacks for new navigation from Frosmo Oy's internal FCP users were collected. Following questionaries was used to collect the feedbacks.

- For what purpose have you used it?
- How long have you used it for?
- How has your experience been so far?
- Have you used the Favorites menu? If yes, then has it been useful to you?
- What are the things you liked the most about the new navigation?
- What are the things that could be changed or improved in new navigation?

5.1.1 Feedback 1

Name: Tomi Rinne.

Role: Customer Success Manager at Frosmo.

Yes, I have used the new navigation. I have used it to perform core activities in the FCP (creation/configuration/monitoring/analysis of modifications etc.), accessing additional features (Instant Search UI) and managing customers' user accounts. The overall experience has been great. It was relatively easy to adapt to it (i.e., learning to use it, compared to the older version).

I have also used the Favorites-menu. It has been the greatest improvement in the navigation UI, I personally think. The combination of more simplistic UI makes it easier to access features that are most relevant for in terms of daily use of the FCP UI. Combined with the Favorite-links menu, that allows quick access to the more "detailed" tasks I execute in the FCP UI. To summarize, the new navigation has accelerated my ability to navigate and carry out tasks in general.

I have nothing to add/suggest as the general user experience has improved remarkably.

5.1.2 Feedback 2

Anonymous user.

Role: Developer at Frosmo.

I have used it for developing solutions for clients. It is the primary tool in my case to provide personalized ads for clients. I have used the new navigation for around 1 and half months. I found it much easier to navigate through and it seems more organized than the previous version. I think it has been designed with a user centric view and is extremely user friendly.

Yes, I have used the Favorites menu. It has been useful in my daily work. It allows me to pin. The most used features while the unused features are hidden and therefore do not distract me. I do not have to try and find the features that I need which may be at least a few clicks away. More organized and compartmentalized sections which contain only relevant features. Conceal other features which are rarely used. Much cleaner and intuitive flow. One thing I realized was the activities section could be added to the favorites section. Maybe try with different color combinations it has always been black and white.

5.1.3 Analysis

Early reception of the new navigation from FCP users is welcoming. It seems to be useful to users. And according to them, it has helped them to carry out their work efficiently.

Moreover, the customizable section also seems to be beneficial, as they can use it in their daily work, for instance they can add links to the navigation which they can access directly.

The objective of the thesis seems to be successful, and user seems to find the new navigation interactive. The usability and functionality issues around the old navigation are seemed to be addressed by the new navigation. More customer reviews are planned to be collected in the future, which should give a in depth analysis of how the new navigation is liked by the FCP users.

5.2 Challenges

In pages written in code other than React, changing the page content's style dynamically when the Sidebar's width changes and when the Sub Navigation opens, or closes was one of the challenging parts of the thesis.

Footer was created but it could not be implemented. Footer is supposed to come after the page content in FCP and in pages written in code other than React, height of the page content could not be allocated beforehand. So, placing the footer created using React was not possible in those pages. And for this reason, new Footer has not been implemented.

5.3 Evaluation

FCP uses React version 16.2, and the most recent stable version of React during the time the thesis was created was 18 (released on June 14, 2022) (Meta Platforms, Inc., 2022b). In some places such as movable list in Favorites menu Editor a React drag and drop library could have been used, which would have required more recent version of React. Upgrades of this sort were out of this thesis scope.

In doing the project I dwelled more into Front-end development and UI development. Even though I had previous knowledge of Front-end development, I get to learn more in-depth about React and how it works under the hood and how it manipulates the DOM.

The FCP has a large codebase, and it was often a challenge to look at the bigger picture and how the new navigation would fit into all the pages of FCP. As the project progressed, things started to come together. In the first development phase, navigation started working in pages created with React without any issues, but it took a while before it started to work properly in all the pages. There were styling issues and the page content would not fit

properly with React's navigation. After creating Sidebar, it was satisfying to see how new navigation would look like in FCP. It became clearer how the whole process would go on from there and how the new navigation could be implemented in production.

While doing the project I learned more about the FCP's code structure, even though I was not actively changing any PHP codes, I had to understand how the different elements of FCP are put together in PHP code. This project has improved my knowledge about FCP in general and, I had a chance to learn more about different aspects of Frosmo Platform.

References

David Herbert. 27.07.2022. What is React.js? URL: https://blog.hubspot.com/website/react-js. Accessed: 4 September 2022.

Frosmo Limited 2022a. FROSMO DOCUMENTATION. Glossary. https://docs.frosmo.com/display/platform/Glossary. Accessed: 3 September 2022.

Frosmo Limited 2022b. FROSMO DOCUMENTATION. Graniitti API. URL: (Frosmo Limited, 2022b) . Accessed: 3 September 2022.

Frosmo Limited 2022c. FROSMO DOCUMENTATION. Introduction to the Frosmo Control Panel.URL: https://docs.frosmo.com/display/getstarted/Introduction+to+the+Frosmo+Control+Panel. Accessed: 3 September 2022.

Frosmo Limited 2022d. FROSMO DOCUMENTATION. Introduction to the Frosmo Platform.URL: https://docs.frosmo.com/display/getstarted/Introduction+to+the+Frosmo+Platform. Accessed: 3 September 2022.

Frosmo Limited 2022e. Frosmo Website. URL: https://frosmo.com/company/. Accessed: 3 September 2022.

Josh, Goldberg. 2022. Learning TypeScript. E-book, From JavaScript to TypeScript, chapter 1. Accessed: 12 September 2022.

Josh, Goldberg. 2022. Learning TypeScript. E-book, From JavaScript to TypeScript, chapter 2. Accessed: 12 September 2022.

Meta Platform, Inc. React Docs 2022a. Introducing Hooks. URL: https://reactjs.org/docs/hooks-intro.html. Accessed: 3 September 2022.

Meta Platform, Inc. React Docs 2022b. React Versions. https://reactjs.org/versions/. Accessed: 13 September 2022.

Meta Platform, Inc. React Docs 2022c. Tutorial: Intro to React. https://reactjs.org/tutorial/tutorial.html. Accessed: 13 September 2022.

Microfocus 2022. What is functional testing? https://www.microfocus.com/en-us/what-is/functional-testing. Accessed: 30 October 2022.

Microsoft 2022a. The TypeScript Handbook. Advance Types. https://www.TypeScriptlang.org/docs/handbook/advanced-types.html. Accessed: 18 September 2022.

Microsoft 2022b. The TypeScript Handbook. Everyday Types. https://www.TypeScriptlang.org/docs/handbook/2/everyday-types.html#differences-between-type-aliases-and-interfaces. Accessed: 18 September 2022.

Mozilla 2022a. MDN web docs. Document.createElement(). https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement. Accessed: 13 October 2022.

Mozilla 2022b. MDN web docs. Document.getElementById(). https://developer.mozilla.org/en-US/docs/Web/API/Document/documentElement. Accessed: 13 October 2022.

Mozilla 2022c. MDN web docs. EventTarget.addEventListener(). https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener. Accessed: 14 October 2022.

Mozilla 2022d. MDN web docs. Introduction to the DOM. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Accessed: 3 September 2022.

Simplilearn 2022. The Best Guide to Know What Is React. URL: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs. Accessed: 3 September 2022.

Software Testing Foundations 5th Edition. Andreas Spillner, Tino Linz. 2021. E-book, testing through the Software Development Lifecycle, chapter 3. Accessed: 12 September 2022.

Tapas Adhikary. 15.03.2022. React Hooks Fundamental for Beginners. https://www.freecodecamp.org/news/react-hooks-fundamentals. Accessed: 13 September 2022.

Technopodia 2017**.** What Does Personalization Mean? URL: https://www.techopedia.com/definition/14712/personalization#:~:text=Personalization%20is%20the%20process%20by,a%20service%20provider%20and%20user. Accessed: 3 September 2022.

Thomas Hamilton. 05.11.2022. What is smoke testing? https://www.guru99.com/smoke-testing.html. Accessed: 30 October 2022.

Tolu Agboola. 29.9.2021. Type Aliases vs Interfaces in TypeScript. https://dev.to/toluagboola/type-aliases-vs-interfaces-in-TypeScript-3ggg. Accessed: 18 September.

Top React Ui frameworks to consider in 2022. https://www.uplers.com/blog/top-react-ui-frameworks. Accessed: 1 October 2022.

W3schools 2022. HTML DOM Element applendChild (). https://www.w3schools.com/jsref/met_node_appendchild.asp. Accessed: 30 October 2022.

Victor Ikechukwu. 02.14.2022. A Beginners Guide. https://www.freecodecamp.org/news/the-beginners-guide-to-react-hooks/. Accessed: 18 September 2022.

Appendices

Appendix 1. Abbrevations

FCP – Frosmo Control Panel.

UI – User Interface.

UX – User Experience.

Appendix 2. All Navigation Items

```
const allNavigationItems: NavigationItems = {
    ITEM_AFFINITY_GROUPS: {
      display: 'Affinity Groups',
      icon: 'fa fa-users',
      id: 'ITEM_AFFINITY_GROUPS',
      quickAccess: true,
      requiredSettings: {
        siteSettings: affinityGroups,
      },
      to: `/sites/${siteId}/affinity-groups`,
    },
    ITEM_ALL_REPORTS: {
      category: 'STATISTICS',
      display: 'All Reports',
      icon: 'fa fa-area-chart',
      id: 'ITEM_ALL_REPORTS',
      quickAccess: false,
      subNav: [
        'ITEM_CONVERSIONS',
        'ITEM_TRAFFIC',
        'ITEM_SEGMENTS_SUMMARY',
        'ITEM_TRENDS',
        'ITEM_DISPLAYS',
        'ITEM_RETENTION',
        'ITEM_SOURCES',
      ],
      to: '/insight/conversions',
    },
    ITEM_ANNOTATIONS: {
      category: 'TOOLS',
```

```
    display: 'Annotations',
    icon: 'fa fa-bookmark',
    id: 'ITEM_ANNOTATIONS',
    quickAccess: true,
    to: '/settings/annotations',
  },
  ITEM_AUDIENCES: {
    display: 'Audiences',
    icon: 'fa fa-users',
    id: 'ITEM_AUDIENCES',
    quickAccess: false,
    subNav: ['ITEM_AUDIENCES_OVERVIEW', 'ITEM_AFFINITY_GROUPS'],
    to: `/new segments/site#/list`,
  },
  ITEM_AUDIENCES_OVERVIEW: {
    display: 'Segments',
    icon: 'fa fa-users',
    id: 'ITEM_AUDIENCES_OVERVIEW',
    quickAccess: false,
    to: `/new segments/site#/list`,
  },
  ITEM_CONFIGURATIONS: {
    display: 'Configurations',
    icon: 'fa fa-cogs',
    id: 'ITEM_CONFIGURATIONS',
    quickAccess: false,
    requiredSettings: {
      addons: 'recommendations',
    },
    to: `/recommendations#/recommendations`,
  },
  ITEM_CONVERSIONS: {
    category: 'STATISTICS',
    display: 'Conversions',
    icon: 'fa fa-line-chart',
    id: 'ITEM_CONVERSIONS',
    quickAccess: true,
    to: '/insight/conversions',
  },
```

```
    ITEM_CONVERSION_DEFINITIONS: {
        category: 'DATA_TRACKING',
        display: 'Conversion Definitions',
        icon: 'fa fa-line-chart',
        id: 'ITEM_CONVERSION_DEFINITIONS',
        quickAccess: true,
        to: '/settings/conversion-definitions#/',
    },
    ITEM_CUSTOM_ACTIONS: {
        category: 'DATA_TRACKING',
        display: 'Custom Actions',
        icon: 'fa fa-hand-pointer-o',
        id: 'ITEM_CUSTOM_ACTIONS',
        quickAccess: true,
        to: '/custom-actions#/custom-actions',
    },
    ITEM_DASHBOARD: {
        display: 'Dashboard',
        icon: 'fa fa-home',
        id: 'ITEM_DASHBOARD',
        to: `/site-overview#/dashboard`,
    },
    ITEM_DISPLAYS: {
        category: 'STATISTICS',
        display: 'Displays',
        icon: 'fa fa-table',
        id: 'ITEM_DISPLAYS',
        quickAccess: true,
        to: `/insight/display-stats`,
    },
    ITEM_EMAIL_RECOMMENDATIONS: {
        category: 'FEATURES',
        display: 'Email Recommendations',
        icon: 'fa fa-envelope',
        id: 'ITEM_EMAIL_RECOMMENDATIONS',
        quickAccess: true,
        requiredSettings: {
            addons: 'email_campaign',
        },
```

```
        to: `/sites/${siteId}/email-recommendations`,
    },

    ITEM_GOOGLE_ANALYTICS: {
        category: 'DATA_TRACKING',
        display: 'Google Analytics',
        icon: 'fa fa-google',
        id: 'ITEM_GOOGLE_ANALYTICS',
        quickAccess: true,
        requiredSettings: {
            addons: 'google_analytics',
        },
        to: '/settings/integration',
    },
    ITEM_MODIFICATIONS: {
        display: 'Modifications',
        icon: 'fa fa-pencil-square-o',
        id: 'ITEM_MODIFICATIONS',
        quickAccess: false,
        subNav: [
            'ITEM_MODIFICATIONS_OVERVIEW',
            'ITEM_PLACEMENTS',
            'ITEM_TEMPLATES',
        ],
        to: `/mahi/message#/list`,
    },
    ITEM_MODIFICATIONS_OVERVIEW: {
        display: 'Overview',
        icon: 'fa fa-list',
        id: 'ITEM_MODIFICATIONS_OVERVIEW',
        quickAccess: false,
        to: `/mahi/message#/list`,
    },
    ITEM_MORE: {
        display: 'More',
        icon: 'fa fa-ellipsis-h',
        id: 'ITEM_MORE',
        quickAccess: false,
        to: ``,
```

```
    },
    ITEM_PLACEMENTS: {
        display: 'Placements',
        icon: 'fa fa-map-marker',
        id: 'ITEM_PLACEMENTS',
        quickAccess: true,
        to: `/mahi/message/positions#/list`,
    },
    ITEM_PRODUCTS: {
        category: 'DATA_TRACKING',
        display: 'Products',
        icon: 'fa fa-shopping-bag',
        id: 'ITEM_PRODUCTS',
        quickAccess: true,
        to: `/sites/${siteId}/products`,
    },
    ITEM_RECOMMENDATIONS_STRATEGIES: {
        category: 'FEATURES',
        display: 'Recommendations',
        icon: 'fa fa-star',
        id: 'ITEM_RECOMMENDATIONS_STRATEGIES',
        quickAccess: true,
        requiredSettings: {
            addons: ['or', 'recommendations', 'email_campaign'],
        },
        subNav: ['ITEM_STRATEGIES', 'ITEM_CONFIGURATIONS'],
        to: `/sites/${siteId}/recommendation-strategies`,
    },
    ITEM_RETENTION: {
        category: 'STATISTICS',
        display: 'Retention',
        icon: 'fa fa-bar-chart',
        id: 'ITEM_RETENTION',
        quickAccess: true,
        to: `/insight/retention`,
    },
    ITEM_SEGMENTS_SUMMARY: {
        category: 'STATISTICS',
        display: 'Segments',
```

```
    icon: 'fa fa-user-plus',
    id: 'ITEM_SEGMENTS_SUMMARY',
    quickAccess: true,
    to: `/insight/segment-summary`,
  },
  ITEM_SHARED_CODE: {
    category: 'FEATURES',
    display: 'Shared Code',
    icon: 'fa fa-code',
    id: 'ITEM_SHARED_CODE',
    quickAccess: true,
    to: `/sites/${siteId}/shared code`,
  },
  ITEM_SOURCES: {
    category: 'STATISTICS',
    display: 'Sources',
    icon: 'fa fa-bar-chart',
    id: 'ITEM_SOURCES',
    quickAccess: true,
    to: `/insight/sources`,
  },
  ITEM_STRATEGIES: {
    display: 'Strategies',
    icon: 'fa fa-lightbulb-o',
    id: 'ITEM_STRATEGIES',
    quickAccess: false,
    to: `/sites/${siteId}/recommendation-strategies`,
  },
  ITEM_TEMPLATES: {
    display: 'Templates',
    icon: 'fa fa-book',
    id: 'ITEM_TEMPLATES',
    quickAccess: true,
    to: `/mahi/template#/list`,
  },
  ITEM_TRAFFIC: {
    category: 'STATISTICS',
    display: 'Traffic',
    icon: 'fa fa-line-chart',
```

```
        id: 'ITEM_TRAFFIC',
        quickAccess: true,
        to: '/insight/users',
    },
    ITEM_TRENDS: {
        category: 'STATISTICS',
        display: 'Trends',
        icon: 'fa fa-bar-chart',
        id: 'ITEM_TRENDS',
        quickAccess: true,
        to: `/insight/segment-changes`,
    },
    ITEM_TRIGGERS: {
        category: 'FEATURES',
        display: 'Triggers',
        icon: 'fa fa-mouse-pointer',
        id: 'ITEM_TRIGGERS',
        quickAccess: true,
        to: '/settings/triggers#/list',
    },
    ITEM_WORKSPACES: {
        category: 'TOOLS',
        display: 'Workspaces',
        icon: 'fa fa-shield',
        id: 'ITEM_WORKSPACES',
        quickAccess: true,
        requiredSettings: {
            addons: 'workspaces',
            condition: CONDITION.OR,
            siteSettings: userHasExtPermission,
        },
        to: '/workspaces#/workspaces',
    },
};
```

Appendix 3. Handle style function

```
const handleStyle = () => {
```

```
    let width: any;
    let left: any;
    let footerMarginLeft: any;
    if (isSideBarMenuOpen && isSubMenuOpen) {
        width = '65%';
        left = '475px';
        footerMarginLeft = '460px';
    } else if (!isSideBarMenuOpen && isSubMenuOpen) {
        width = '75%';
        left = '330px';
        footerMarginLeft = '325px';
    } else if (isSideBarMenuOpen && !isSubMenuOpen) {
        width = '80%';
        left = '215px';
        footerMarginLeft = '220px';
    } else {
        width = '90%';
        left = '100px';
        footerMarginLeft = '85px';
    }
    if (wrapper) {
        wrapper.style.backgroundColor = 'white';
    }
    if (window.innerWidth < 1300) {
        left = '110px';
        footerMarginLeft = '85px';
    }
    if (window.innerWidth < 1300 && isSubMenuOpen) {
        left = '330px';
        footerMarginLeft = '325px';
    }
    // Min-height and max-height could be changed
    // when the margin-top: 7px of ul elements is taken away.
    // Position relative works but when the contents width is
    // not longer than display's height then footer doesn't stay at the                bottom.
    if (!styleEl) {
        styleEl = document.createElement('style');
        styleEl.id = 'nonreact-content-style';
        document.head.appendChild(styleEl);
```

```
    }
    styleEl.innerHTML = `
      #nonreact-content {
        left: ${left};
        width: ${width};
        position: relative;
        top: 10px;
        min-height: 88vh;
        }
      #footer {
        margin-left: ${footerMarginLeft} ! important;
        position: relative;
        min-height:60px;
        max-height: 75px;
        width: auto;
        bottom:0;
        border-top: none !important;
        }
    `;
};
```