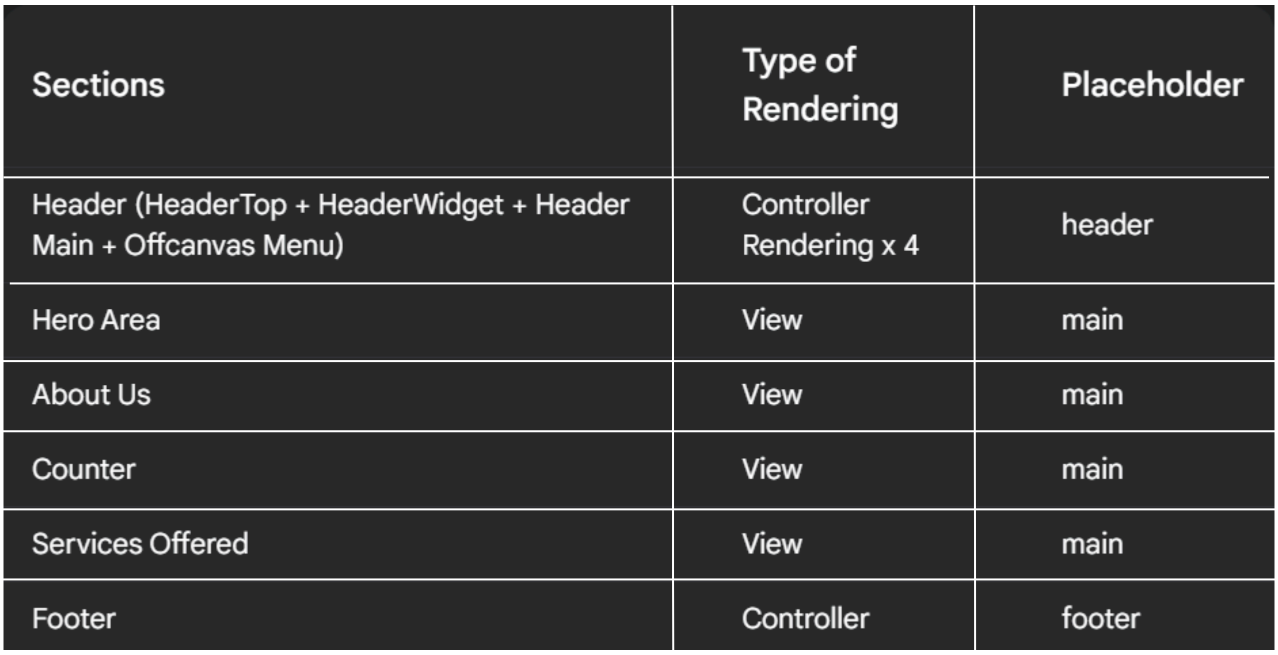
Sitecore Assessment Steps

Yash Gupta
ALT/IND/8171

Step 1) The first stage after raw coding the website was to analyse the website and individual components/renderings to determine the base data templates that can be reused across various site components/renderings, as well as the renderings (View/Controller) that will be used in Sitecore.

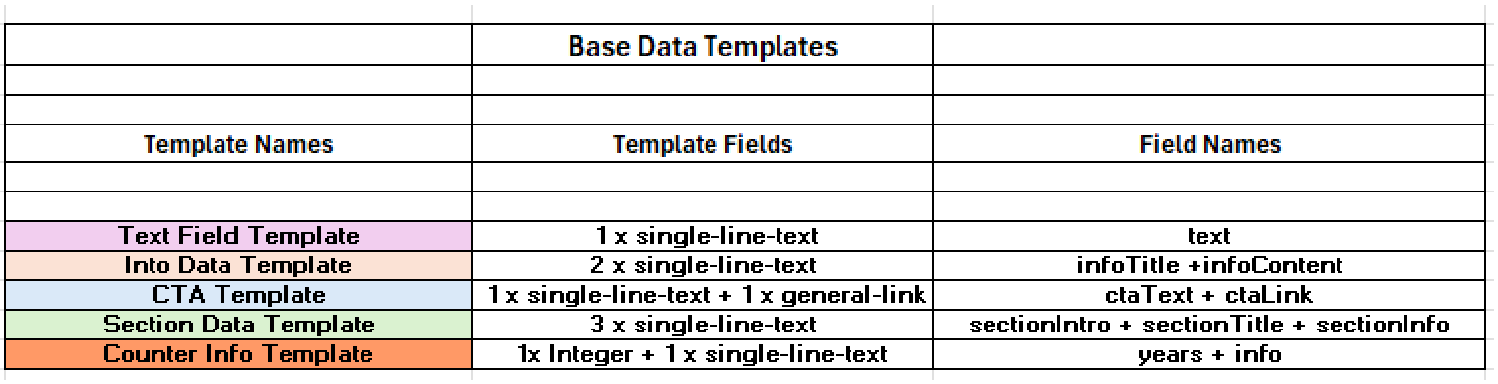
1.1) To effectively analyse it, I created an excel file (which is included in this repo) with a list of all the components to be created

1.2) Divided the components into individual fields and analysed them to identify the base data templates that can be reused.



Sections	Type of Rendering	Placeholder
Header (HeaderTop + HeaderWidget + Header Main + Offcanvas Menu)	Controller Rendering x 4	header
Hero Area	View	main
About Us	View	main
Counter	View	main
Services Offered	View	main
Footer	Controller	footer

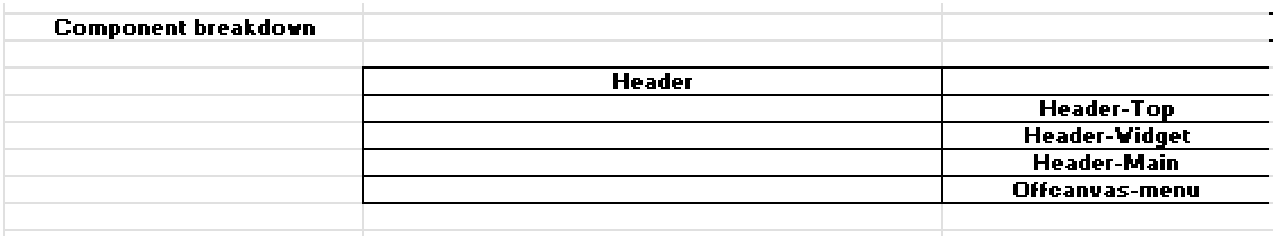
Controller/View Renderings Table



	Base Data Templates	
Template Names	Template Fields	Field Names
Text Field Template	1 x single-line-text	text
Into Data Template	2 x single-line-text	infoTitle +infoContent
CTA Template	1 x single-line-text + 1 x general-link	ctaText + ctaLink
Section Data Template	3 x single-line-text	sectionIntro + sectionTitle + sectionInfo
Counter Info Template	1x Integer + 1 x single-line-text	years + info

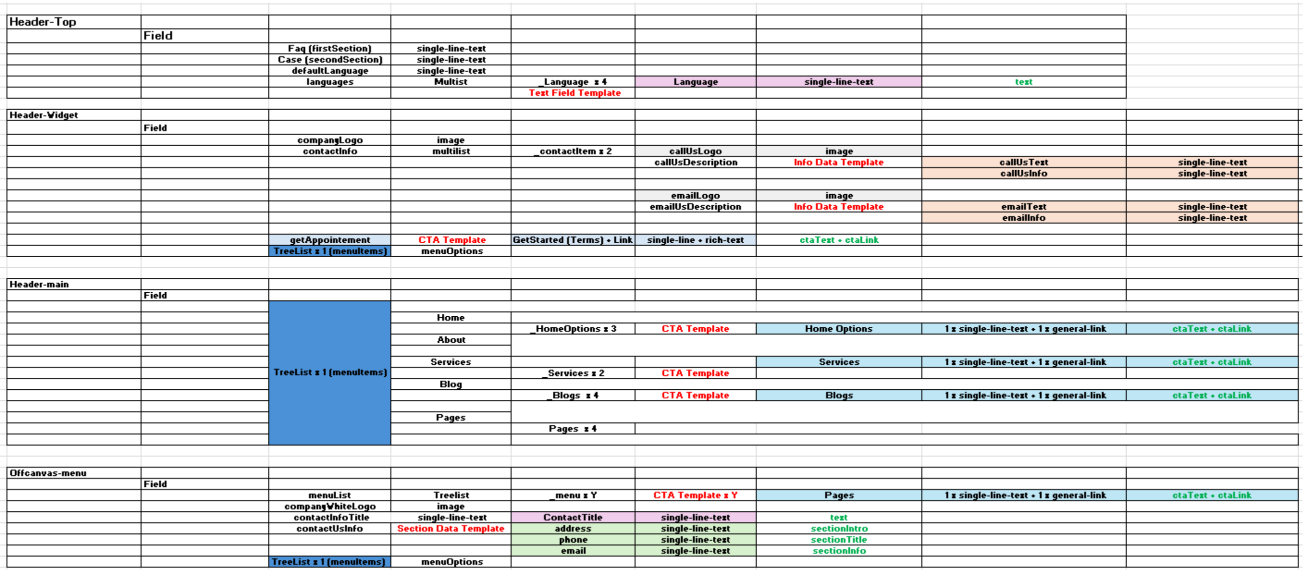
Base Data Templates Table

The attached excel file contains the detailed component breakdown of each website component/rendering



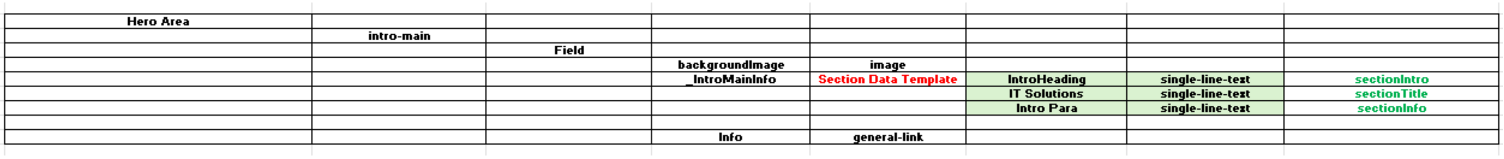
Component breakdown		
	Header	
		Header-Top
		Header-Widget
		Header-Main
		Offcanvas-menu

Header Further Breakdown into different components



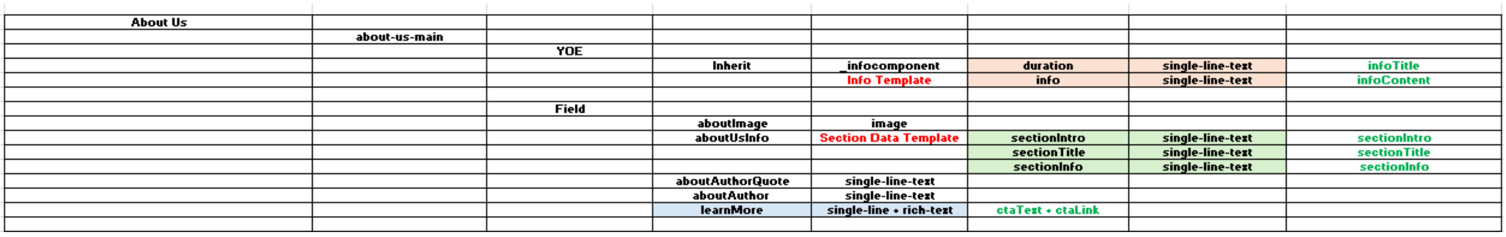
Header-Top								
	Field							
		Faq (firstSection)	single-line-text					
		Case (secondSection)	single-line-text					
		defaultLanguage	single-line-text					
		languages	Multist	_Language x 4	Language	single-line-text	text	
				Text Field Template				
Header-Widget								
	Field							
		companyLogo	image					
		contactInfo	multilist	_contactItem x 2	callUsLogo	image		
					callUsDescription	Info Data Template	callUsText	single-line-text
							callUsInfo	single-line-text
					emailLogo	image		
					emailUsDescription	Info Data Template	emailText	single-line-text
							emailInfo	single-line-text
		getAppointement	CTA Template	GetStarted (Terms) + Link	single-line + rich-text	ctaText + ctaLink		
		TreeList x 1 (menuItems)	menuOptions					
Header-main								
	Field							
		TreeList x 1 (menuItems)	Home					
				_HomeOptions x 3	CTA Template	Home Options	1 x single-line-text + 1 x general-link	ctaText + ctaLink
			About					
			Services			Services	1 x single-line-text + 1 x general-link	ctaText + ctaLink
				_Services x 2	CTA Template			
			Blog					
				_Blogs x 4	CTA Template	Blogs	1 x single-line-text + 1 x general-link	ctaText + ctaLink
			Pages					
				Pages x 4				
Offcanvas-menu								
	Field							
		menuList	Treelist	_menu x Y	CTA Template x Y	Pages	1 x single-line-text + 1 x general-link	ctaText + ctaLink
		companyWhiteLogo	image					
		contactInfoTitle	single-line-text	ContactTitle	single-line-text	text		
		contactUsInfo	Section Data Template	address	single-line-text	sectionIntro		
				phone	single-line-text	sectionTitle		
				email	single-line-text	sectionInfo		
		TreeList x 1 (menuItems)	menuOptions					

In Detail Breakdown of header components



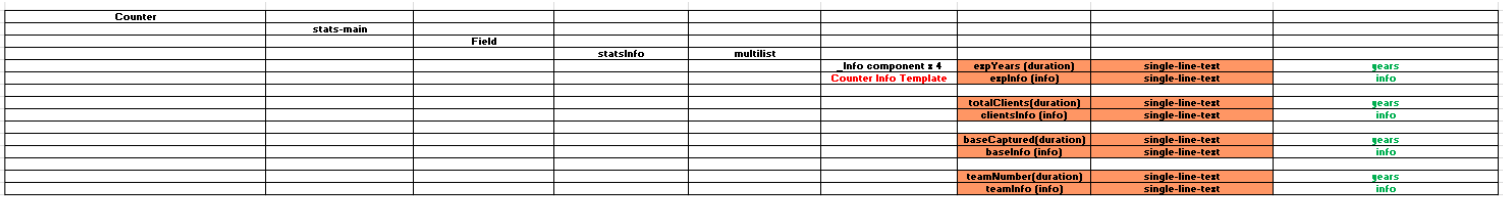
Hero Area							
	intro-main						
		Field					
			backgroundImage	image			
			_IntroMainInfo	Section Data Template	IntroHeading	single-line-text	sectionIntro
					IT Solutions	single-line-text	sectionTitle
					Intro Para	single-line-text	sectionInfo
			Info	general-link			

Hero Area Component Breakdown



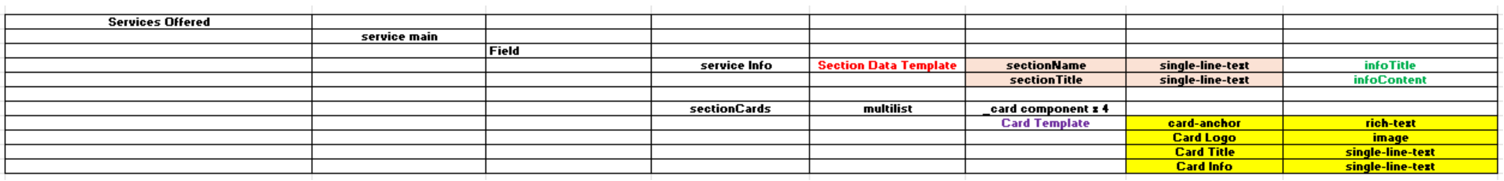
About Us							
	about-us-main						
		YOE					
			Inherit	_infocomponent	duration	single-line-text	infoTitle
				Info Template	info	single-line-text	infoContent
		Field					
			aboutImage	image			
			aboutUsInfo	Section Data Template	sectionIntro	single-line-text	sectionIntro
					sectionTitle	single-line-text	sectionTitle
					sectionInfo	single-line-text	sectionInfo
			aboutAuthorQuote	single-line-text			
			aboutAuthor	single-line-text			
			learnMore	single-line + rich-text	ctaText + ctaLink		

About Us Area Component Breakdown



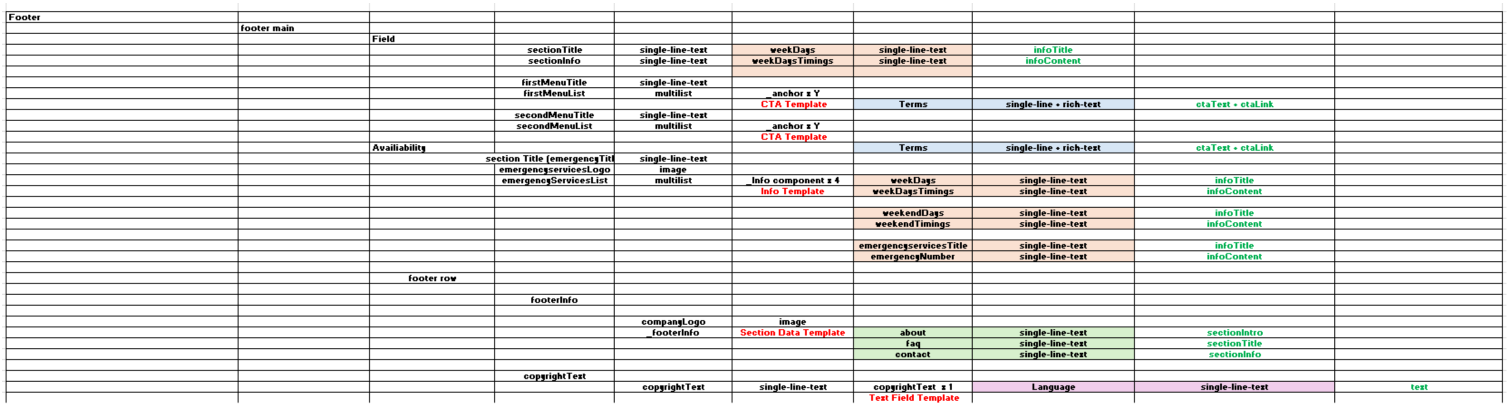
Counter								
	stats-main							
		Field						
			statsInfo	multilist				
					_Info component x 4	expYears (duration)	single-line-text	years
					Counter Info Template	expInfo (info)	single-line-text	info
						totalClients(duration)	single-line-text	years
						clientsInfo (info)	single-line-text	info
						baseCaptured(duration)	single-line-text	years
						baseInfo (info)	single-line-text	info
						teamNumber(duration)	single-line-text	years
						teamInfo (info)	single-line-text	info

Counter Area Component Breakdown



Services Offered							
	service main						
		Field					
			service Info	Section Data Template	sectionName	single-line-text	infoTitle
					sectionTitle	single-line-text	infoContent
			sectionCards	multilist	_card component x 4		
					Card Template	card-anchor	rich-text
						Card Logo	image
						Card Title	single-line-text
						Card Info	single-line-text

Service Area Component Breakdown



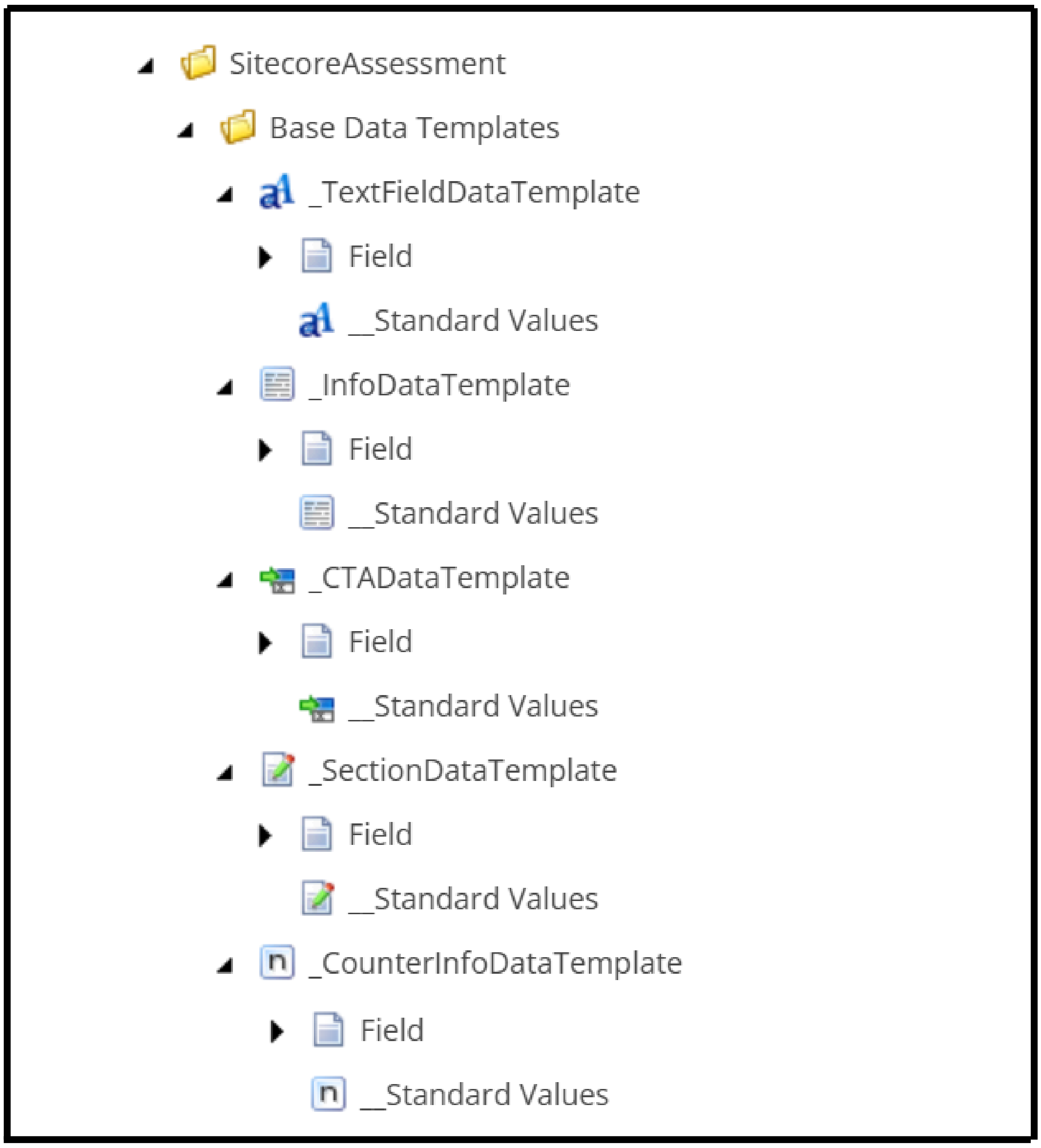
Footer									
	footer main								
		Field							
			sectionTitle	single-line-text	weekDays	single-line-text	infoTitle		
			sectionInfo	single-line-text	weekDaysTimings	single-line-text	infoContent		
			firstMenuTitle	single-line-text					
			firstMenuList	multilist	_anchor x Y				
					CTA Template	Terms	single-line + rich-text	ctaText + ctaLink	
			secondMenuTitle	single-line-text					
			secondMenuList	multilist	_anchor x Y				
					CTA Template				
		Availiability				Terms	single-line + rich-text	ctaText + ctaLink	
			section Title (emergencyTitl	single-line-text					
			emergencyservicesLogo	image					
			emergencyServicesList	multilist	_Info component x 4	weekDays	single-line-text	infoTitle	
					Info Template	weekDaysTimings	single-line-text	infoContent	
						weekendDays	single-line-text	infoTitle	
						weekendTimings	single-line-text	infoContent	
						emergencyservicesTitle	single-line-text	infoTitle	
						emergencyNumber	single-line-text	infoContent	
		footer row							
			footerInfo						
				companyLogo	image				
				_footerInfo	Section Data Template	about	single-line-text	sectionIntro	
						faq	single-line-text	sectionTitle	
						contact	single-line-text	sectionInfo	
			copyrightText						
				copyrightText	single-line-text	copyrightText x 1	Language	single-line-text	text
						Text Field Template			

Footer Component Breakdown

Click to download the file - Download

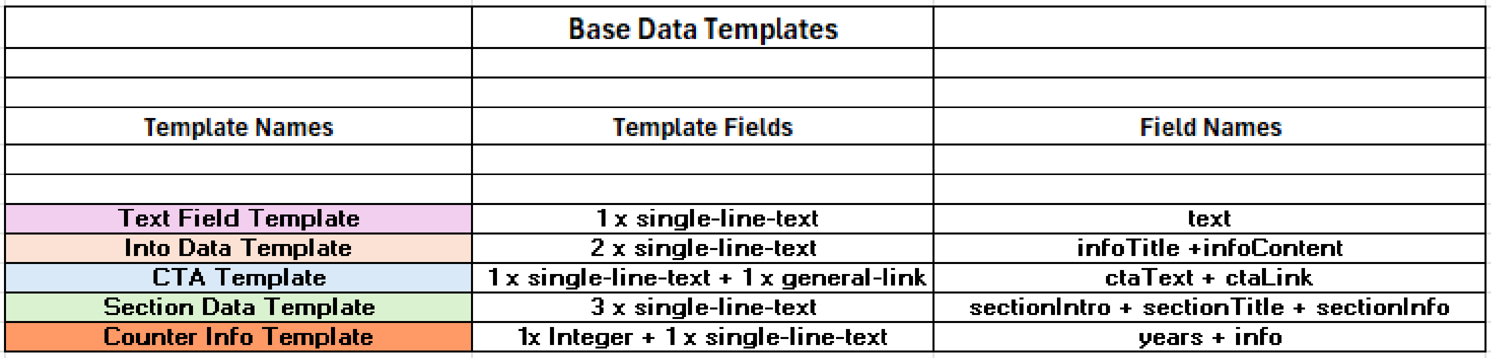
Step 2) Now that the overall structure of the website components is clear, I began by creating the necessary templates, which included the branch template (renderings/components), base data templates, data templates inheriting the base data template, and page templates in a structured folder, along with the standard values assigned to the base data templates and data templates.

2.1) Created base data templates to reuse across various website components/renderings, with appropriate field names and standard values.



- SitecoreAssessment
  - Base Data Templates
    - _TextFieldDataTemplate
      - Field
      - __Standard Values
    - _InfoDataTemplate
      - Field
      - __Standard Values
    - _CTADataTemplate
      - Field
      - __Standard Values
    - _SectionDataTemplate
      - Field
      - __Standard Values
    - _CounterInfoDataTemplate
      - Field
      - __Standard Values

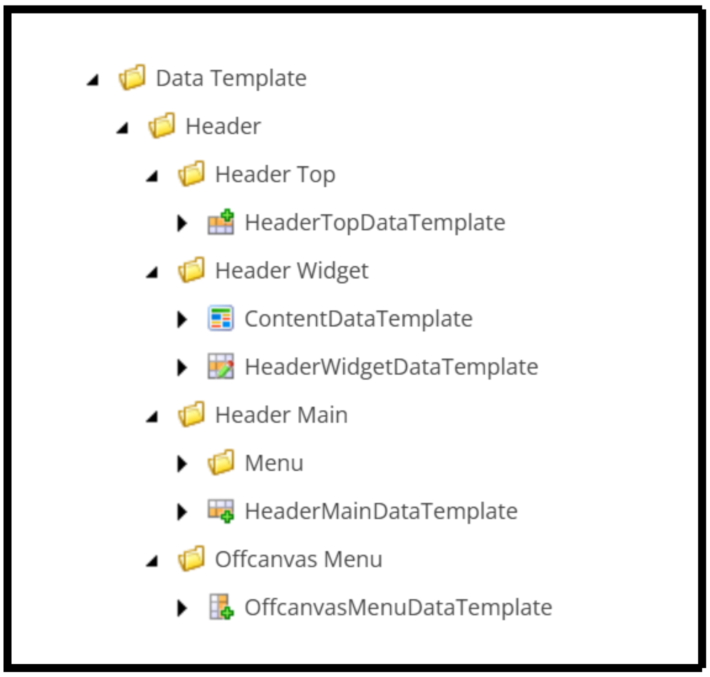
Base Data Templates



	Base Data Templates	
Template Names	**Template Fields**	**Field Names**
Text Field Template	1 x single-line-text	text
Into Data Template	2 x single-line-text	infoTitle +infoContent
CTA Template	1 x single-line-text + 1 x general-link	ctaText + ctaLink
Section Data Template	3 x single-line-text	sectionIntro + sectionTitle + sectionInfo
Counter Info Template	1x Integer + 1 x single-line-text	years + info

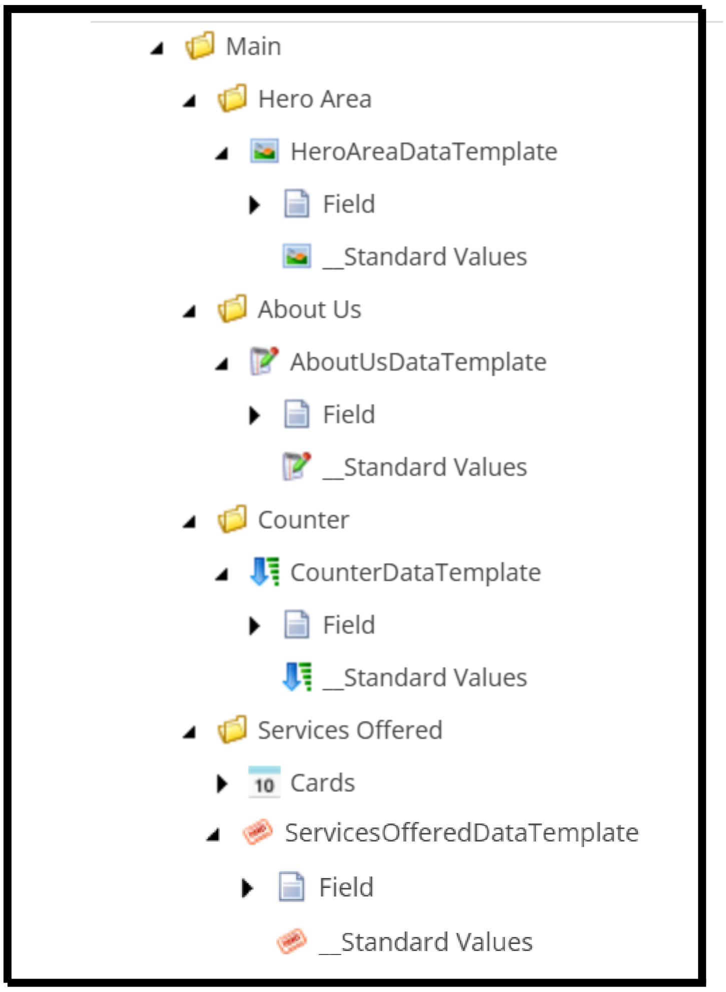
Base Data Templates Tables

2.2) Generated data templates with appropriate base data templates, field names, and standard values.



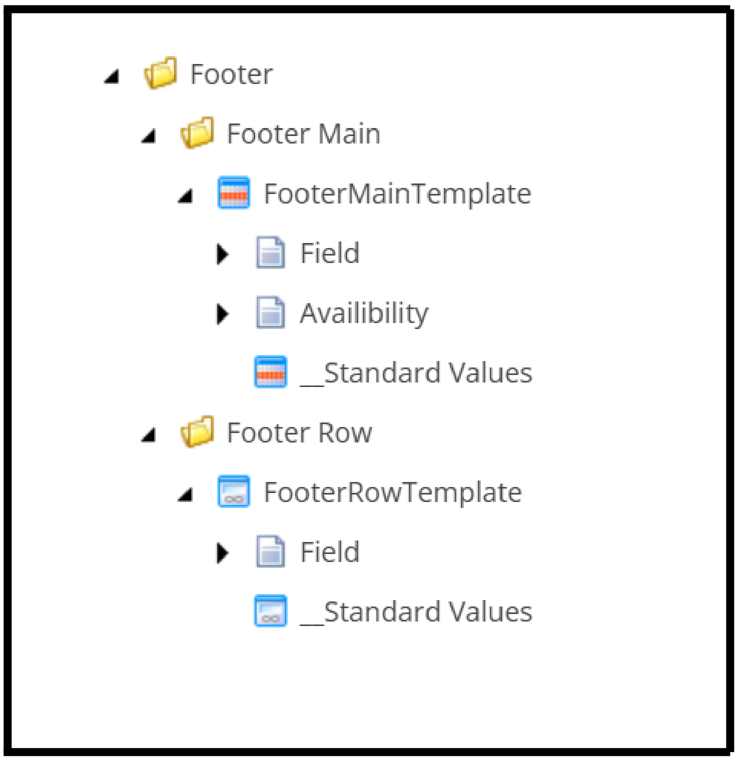
- Data Template
  - Header
    - Header Top
      - HeaderTopDataTemplate
    - Header Widget
      - ContentDataTemplate
      - HeaderWidgetDataTemplate
    - Header Main
      - Menu
      - HeaderMainDataTemplate
    - Offcanvas Menu
      - OffcanvasMenuDataTemplate

Header Data Templates



- Main
  - Hero Area
    - HeroAreaDataTemplate
      - Field
      - __Standard Values
  - About Us
    - AboutUsDataTemplate
      - Field
      - __Standard Values
  - Counter
    - CounterDataTemplate
      - Field
      - __Standard Values
  - Services Offered
    - Cards
    - ServicesOfferedDataTemplate
      - Field
      - __Standard Values

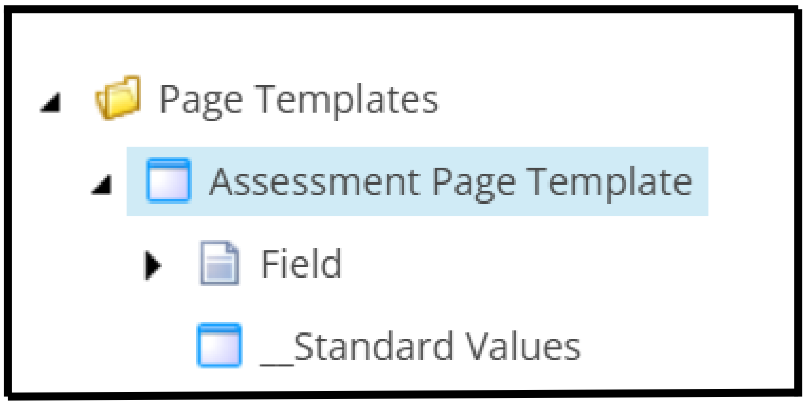
Main section Data Templates (Hero Area + About Us + Counter + Services Offered)



Footer
Footer Main
FooterMainTemplate
Field
Availibility
__Standard Values
Footer Row
FooterRowTemplate
Field
__Standard Values

Footer Data Templates

2.3) Created the page template to be utilised.



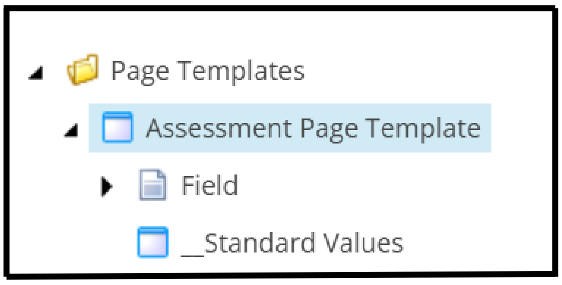
Page Templates
Assessment Page Template
Field
__Standard Values

Page Template

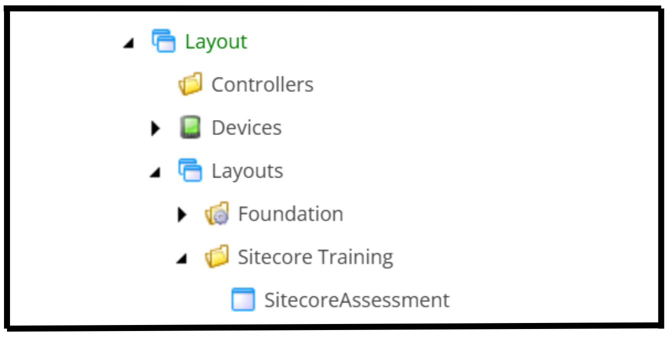
Step 3) The following step was to construct a layout in the layouts folder that is linked with the website raw code and add it to the page template.

3.1) Made a layout and added the source for it.

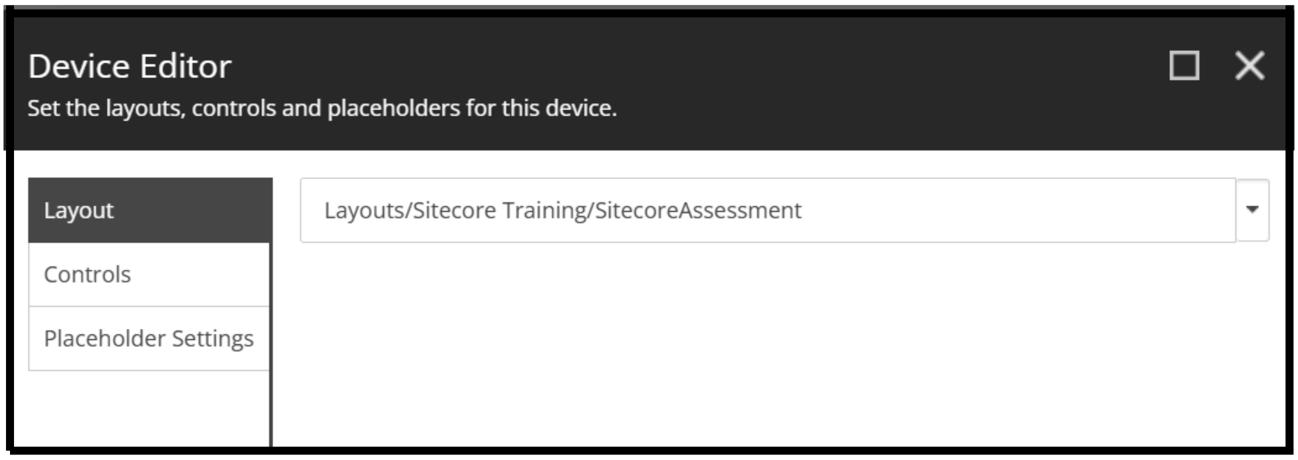
3.2) Added the layout to the presentation details of the produced page template.



Page Template



Layout



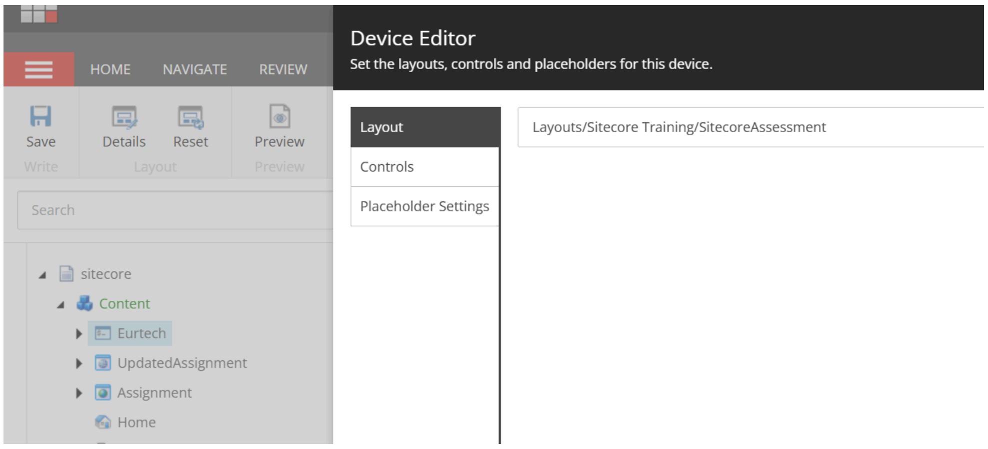
Device Editor

Set the layouts, controls and placeholders for this device.

Layout: Layouts/Sitecore Training/SitecoreAssessment

Adding the page Template in the layout

Step 4) Created an item inheriting the page base template in the sitecore/content to construct the assignment item and to examine the raw html/css/jss working of the site, to confirm the layout is working properly.



Device Editor

Set the layouts, controls and placeholders for this device.

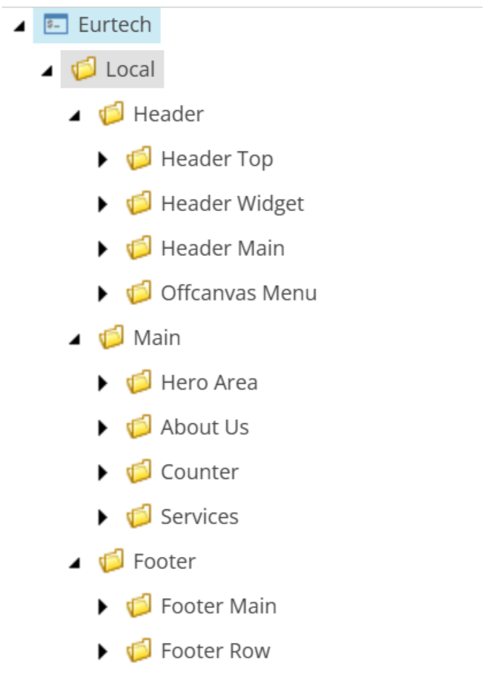
Layout: Layouts/Sitecore Training/SitecoreAssessment

Controls

Placeholder Settings

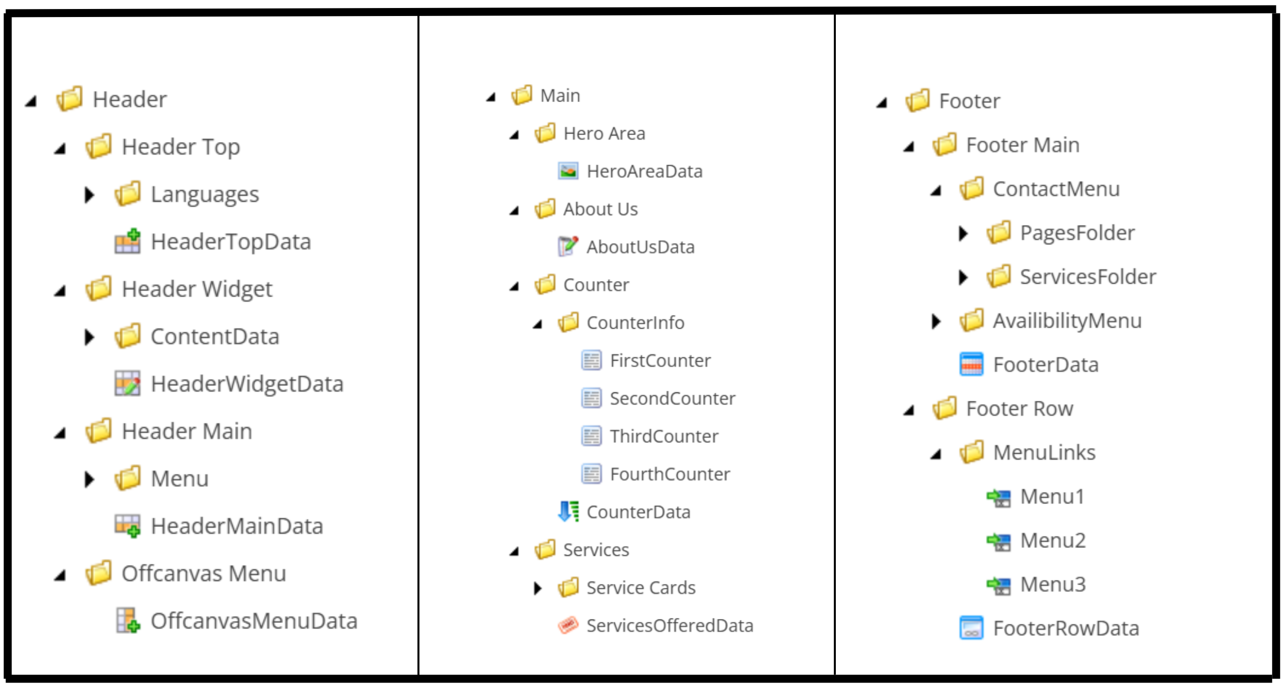
Adding the page Template in the layout

Step 5) The next step was to establish a folder and inherit the branch template to create the appropriate data templates matching to the components folder in the component.



Data Source Template Structure inherited from Branch Template

Step 6) The next step is to inherit the necessary data templates and put the data/data source into them.



- Header
  - Header Top
    - Languages
    - HeaderTopData
  - Header Widget
    - ContentData
    - HeaderWidgetData
  - Header Main
    - Menu
    - HeaderMainData
  - Offcanvas Menu
    - OffcanvasMenuData

- Main
  - Hero Area
    - HeroAreaData
  - About Us
    - AboutUsData
  - Counter
    - CounterInfo
      - FirstCounter
      - SecondCounter
      - ThirdCounter
      - FourthCounter
    - CounterData
  - Services
    - Service Cards
    - ServicesOfferedData

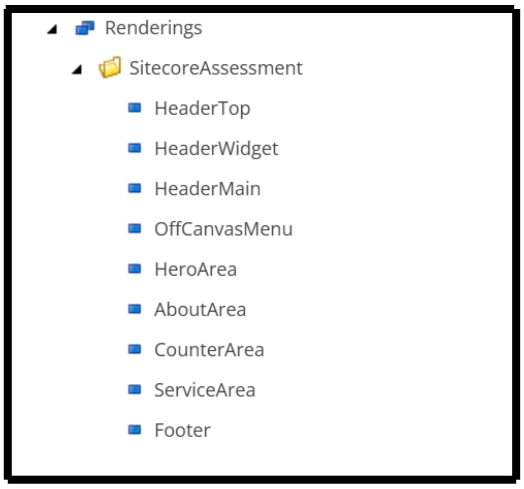
- Footer
  - Footer Main
    - ContactMenu
      - PagesFolder
      - ServicesFolder
    - AvailibilityMenu
    - FooterData
  - Footer Row
    - MenuLinks
      - Menu1
      - Menu2
      - Menu3
    - FooterRowData

Data templates

Step 7) After completing the data sources, the raw Html/CSS was separated into renderings, and the appropriate placeholders were added to the main code.

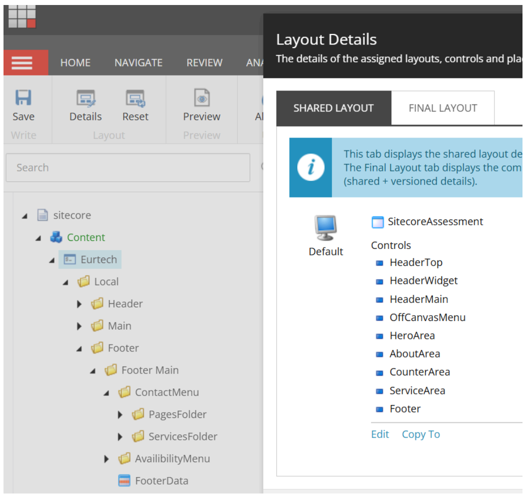
7.1) Created View/Controller renderings by attaching the various views, assigning the corresponding raw Html/css, and changing the main website html to provide the required placeholders.

7.2) The following step was to put the renderings in the main content tree site item along with the data sources produced previously and check whether the site is running correctly with renderings.



Renderings
- SitecoreAssessment
  - HeaderTop
  - HeaderWidget
  - HeaderMain
  - OffCanvasMenu
  - HeroArea
  - AboutArea
  - CounterArea
  - ServiceArea
  - Footer

Renderings



Layout Details

The details of the assigned layouts, controls and pla

SHARED LAYOUT | FINAL LAYOUT

This tab displays the shared layout de
The Final Layout tab displays the com
(shared + versioned details).

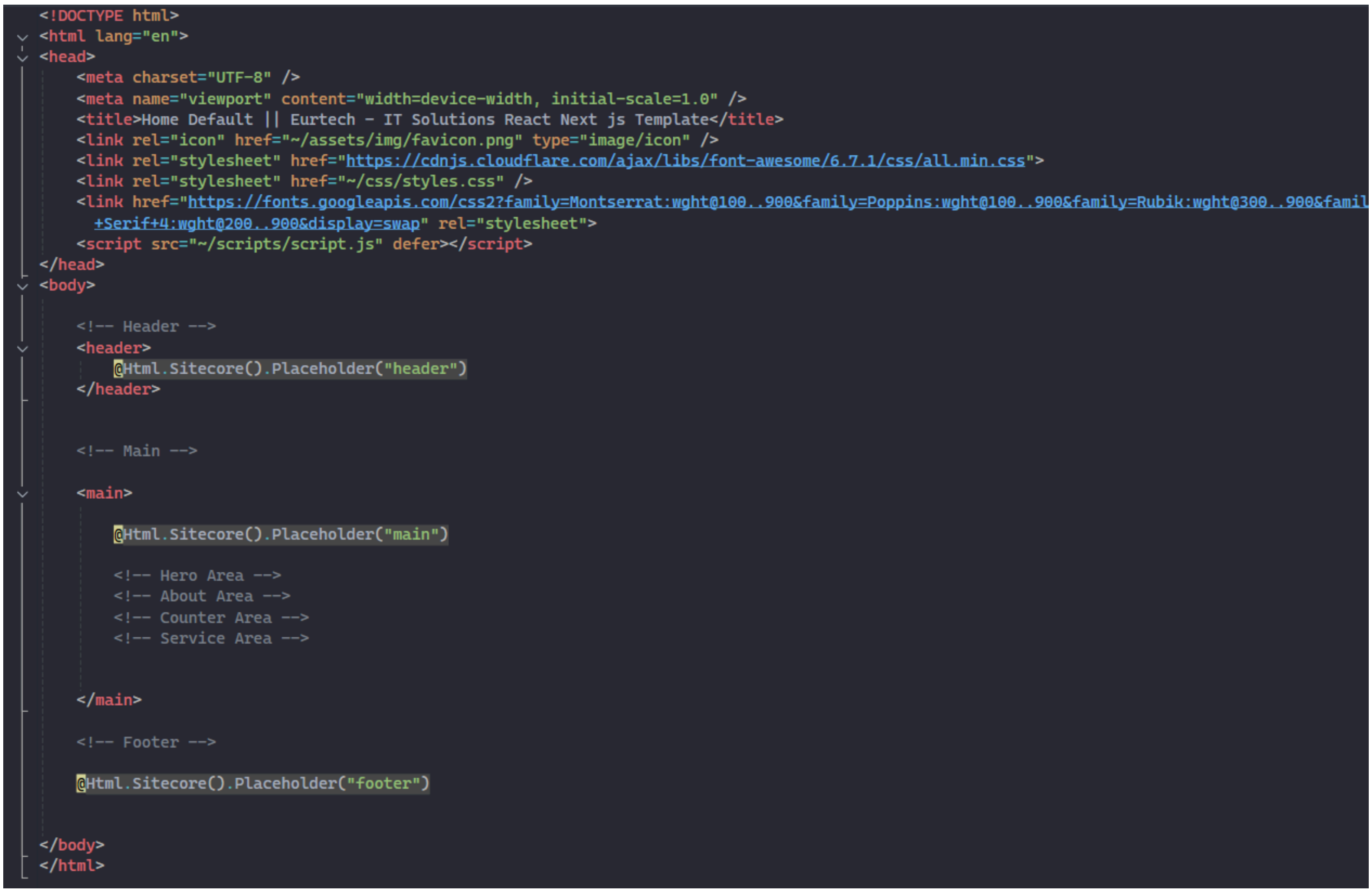
Default

SitecoreAssessment

Controls
- HeaderTop
- HeaderWidget
- HeaderMain
- OffCanvasMenu
- HeroArea
- AboutArea
- CounterArea
- ServiceArea
- Footer

Edit Copy To

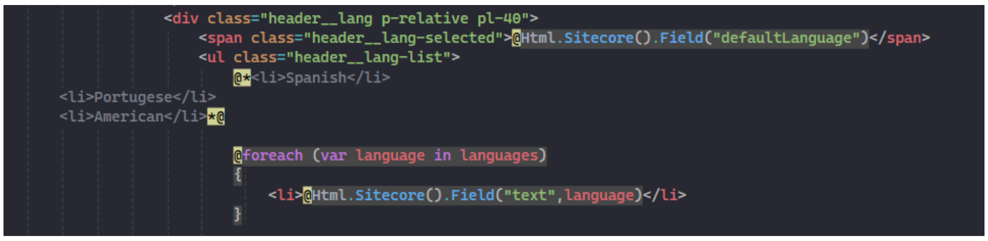
Added rendering to the Site item



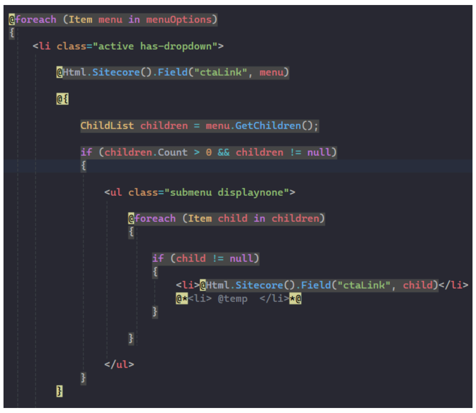
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Home Default || Eurtech - IT Solutions React Next js Template</title>
    <link rel="icon" href="~/assets/img/favicon.png" type="image/icon" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.1/css/all.min.css">
    <link rel="stylesheet" href="~/css/styles.css" />
    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@100..900&family=Poppins:wght@100..900&family=Rubik:wght@300..900&famil
      +Serif+4:wght@200..900&display=swap" rel="stylesheet">
    <script src="~/scripts/script.js" defer></script>
</head>
<body>

    <!-- Header -->
    <header>
        @Html.Sitecore().Placeholder("header")
    </header>


    <!-- Main -->

    <main>

        @Html.Sitecore().Placeholder("main")

        <!-- Hero Area -->
        <!-- About Area -->
        <!-- Counter Area -->
        <!-- Service Area -->


    </main>

    <!-- Footer -->

    @Html.Sitecore().Placeholder("footer")


</body>
</html>
```

Placeholders

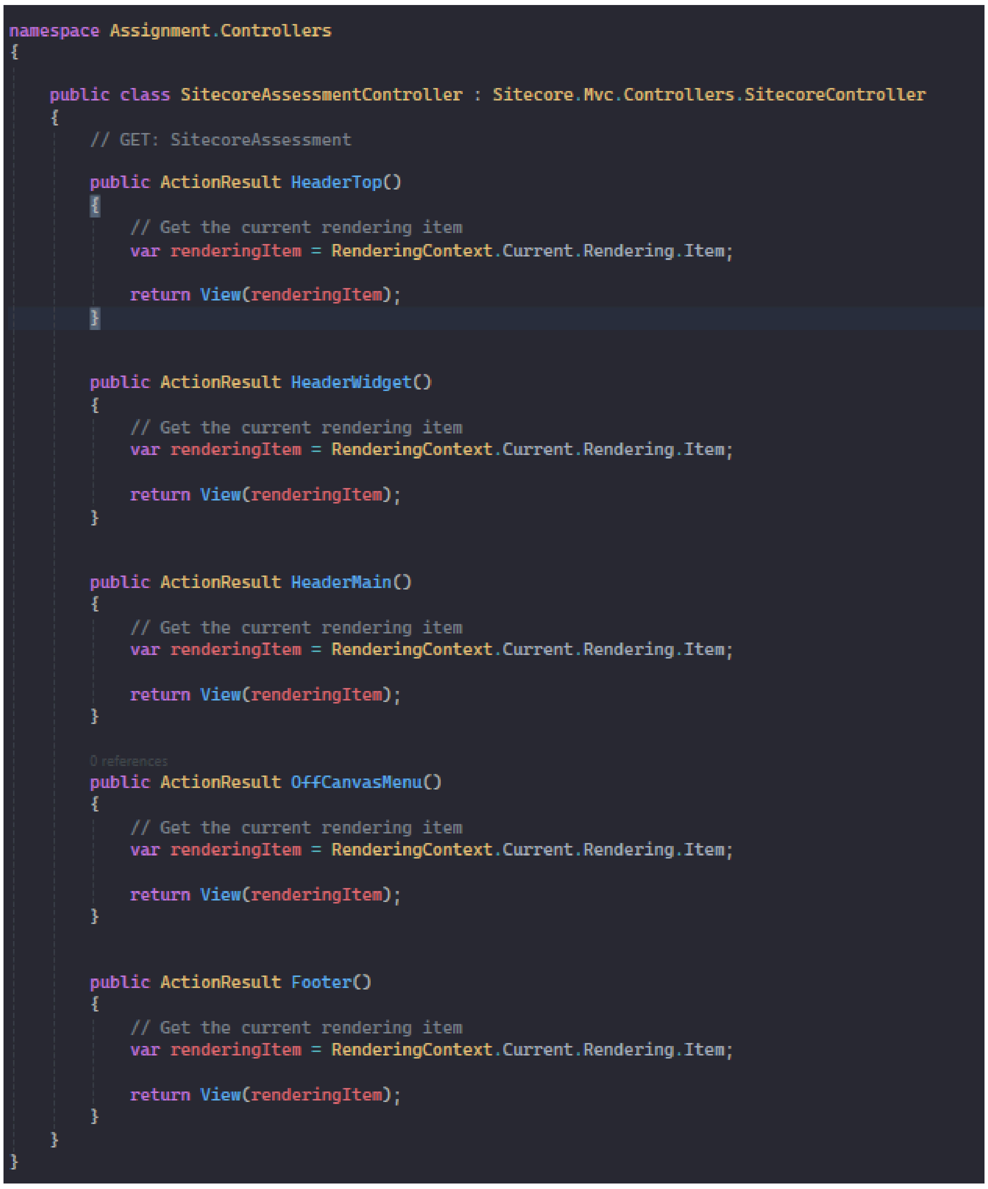
Step 8) The last step was to replace the raw html/css/js data with the applicable sitecore fields from the data source and using the necessary logic based on the rendering utilised.



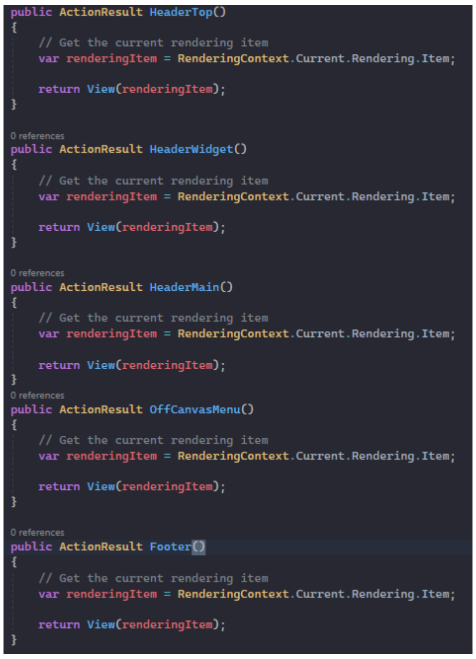
```
                <div class="header__lang p-relative pl-40">
                    <span class="header__lang-selected">@Html.Sitecore().Field("defaultLanguage")</span>
                    <ul class="header__lang-list">
                        @*<li>Spanish</li>
        <li>Portugese</li>
        <li>American</li>*@

                        @foreach (var language in languages)
                        {
                            <li>@Html.Sitecore().Field("text",language)</li>
                        }
```



```
@foreach (Item menu in menuOptions)
{
    <li class="active has-dropdown">

        @Html.Sitecore().Field("ctaLink", menu)

        @{

            ChildList children = menu.GetChildren();

            if (children.Count > 0 && children != null)
            {

                <ul class="submenu displaynone">

                    @foreach (Item child in children)
                    {

                        if (child != null)
                        {
                            <li>@Html.Sitecore().Field("ctaLink", child)</li>
                            @*<li> @temp  </li>*@
                        }

                    }

                </ul>
            }
        }
```



```
namespace Assignment.Controllers
{

    public class SitecoreAssessmentController : Sitecore.Mvc.Controllers.SitecoreController
    {
        // GET: SitecoreAssessment

        public ActionResult HeaderTop()
        {
            // Get the current rendering item
            var renderingItem = RenderingContext.Current.Rendering.Item;

            return View(renderingItem);
        }


        public ActionResult HeaderWidget()
        {
            // Get the current rendering item
            var renderingItem = RenderingContext.Current.Rendering.Item;

            return View(renderingItem);
        }


        public ActionResult HeaderMain()
        {
            // Get the current rendering item
            var renderingItem = RenderingContext.Current.Rendering.Item;

            return View(renderingItem);
        }

        0 references
        public ActionResult OffCanvasMenu()
        {
            // Get the current rendering item
            var renderingItem = RenderingContext.Current.Rendering.Item;

            return View(renderingItem);
        }


        public ActionResult Footer()
        {
            // Get the current rendering item
            var renderingItem = RenderingContext.Current.Rendering.Item;

            return View(renderingItem);
        }
    }
}
```



```
public ActionResult HeaderTop()
{
    // Get the current rendering item
    var renderingItem = RenderingContext.Current.Rendering.Item;

    return View(renderingItem);
}

0 references
public ActionResult HeaderWidget()
{
    // Get the current rendering item
    var renderingItem = RenderingContext.Current.Rendering.Item;

    return View(renderingItem);
}

0 references
public ActionResult HeaderMain()
{
    // Get the current rendering item
    var renderingItem = RenderingContext.Current.Rendering.Item;

    return View(renderingItem);
}
0 references
public ActionResult OffCanvasMenu()
{
    // Get the current rendering item
    var renderingItem = RenderingContext.Current.Rendering.Item;

    return View(renderingItem);
}

0 references
public ActionResult Footer()
{
    // Get the current rendering item
    var renderingItem = RenderingContext.Current.Rendering.Item;

    return View(renderingItem);
}
```

Step 9) The final step was to confirm the website was running fine after replacing the static material with the dynamic content from Sitecore and making the appropriate changes if needed.