OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

PixelFlow

BSc Computer Science & Information Technology
CT413 Final Year Project

David Bohan - 20436904 - March 2024

Table of Contents

Acknowledgements

I would like to express my gratitude to my project supervisor, Effirul Ramlan, for his support and guidance for the duration of my time creating PixelFlow. His openness to discussions and meetings, his prompt email responses to queries, and willingness to dedicate his time are truly appreciated. This was all while maintaining a great sense of humour, which made the process much more enjoyable.

I am thankful to my friends and family, who have encouraged me at every stage and participated in numerous user tests. Additionally, I extend my appreciation to the faculty in the College of Computer Science at the University of Galway. Their dedication to teaching has significantly contributed to my academic and personal growth over the last four years of study.

Lastly, my gratitude goes to the examiners who are taking the time to review my project. Your effort in evaluating my work is greatly appreciated.

Section One - Introduction

1.1 Document Outline

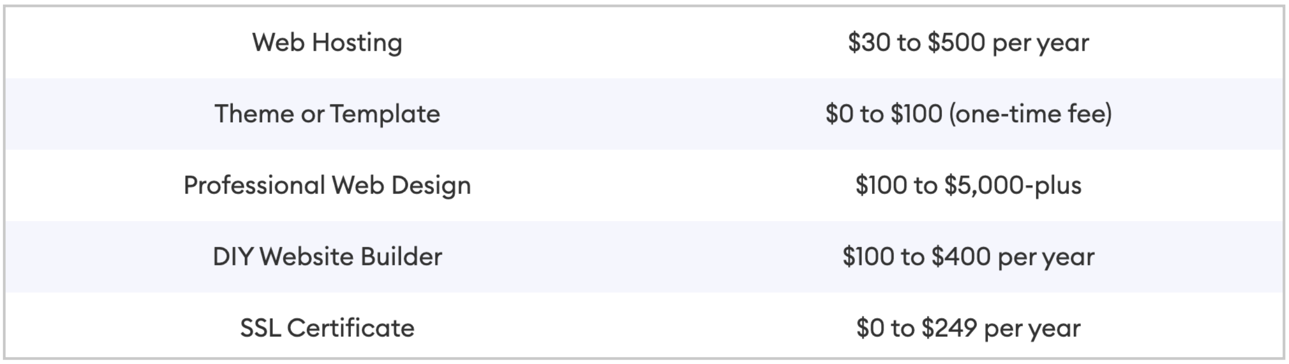
This document outlines the CT413 Final Year Project, PixelFlow, undertaken as a requirement for the Bachelor of Science in Computer Science & Information Technology at the University of Galway, Ireland. It serves as a formal report, offering a comprehensive overview of the software development cycle from inception to completion.

The goal of this document is to provide detailed accounts of the project's background, the technologies used, and software implementation. Upon completing the report, the reader will possess an in-depth understanding of the project's purpose and functionality, alongside reflections on personal insights and directions for future work.

1.2 Project Background

The idea for this project emerges from a personal interest in freelance web design and development. Having previously worked with many local Irish businesses on their websites, I have directly observed the challenges these businesses face in establishing an online presence. There are multiple components that go into the creation of a successful website. There are typically two primary options for businesses;

i). Hire Professionals: Engaging a professional team of designers, copywriters and developers will ensure a polished final result. However, this approach can be expensive and demands substantial communication between both parties. These professionals can typically charge anywhere from $40 to $160 an hour for their services according to Zippia [1].



Web Hosting	$30 to $500 per year
Theme or Template	$0 to $100 (one-time fee)
Professional Web Design	$100 to $5,000-plus
DIY Website Builder	$100 to $400 per year
SSL Certificate	$0 to $249 per year

Figure 1: Website Cost According to Forbes [2]

ii). DIY Approach: The process of building a website yourself can be time-consuming and lack a professional touch, with the resulting website not being mobile-responsive, SEO-friendly, or optimised for media, and may miss key elements like security and accessibility. According to Wix [3], building a website can take from around one to six months, depending on the goals and resources available.

Boston Consulting Group [4] has found "almost a quarter of all URLs have at least one website issue—bloated pages that load sluggishly, broken pages that don't load at all, pages that are not linked internally—and on average, nearly 40% of these issues are critical in nature. Lost visitors are eroding the effectiveness of marketing expenditure by as much as 20%, and companies are squandering thousands of dollars a month."

Due to the high costs and/or steep learning curve involved, many business decide to not maintain a website at all. This decision can negatively impact their brand and ultimately lead to significant losses. According to a report conducted by Harris Poll on behalf of Salesforce [5], 79% of shoppers research products online prior to purchasing in-store. The goal of PixelFlow is to provide a new-age marketing tool, designed to address the problems presented by both the professional and DIY development paradigms.

To alleviate these issues, thus to allow for an easier and more streamlined experience for the user, PixelFlow provides a robust website development platform, augmented with generative AI capabilities. The PixelFlow AI website builder can create a website entirely using simple text prompts, based on the business description and the services to be listed on the site. The AI will then create a website, including the initial copy, images and core pages needed to support the business. This method can shave off weeks—if not months—from the typical development process and present a fully functional website in minutes.

1.3 Feasibility Study

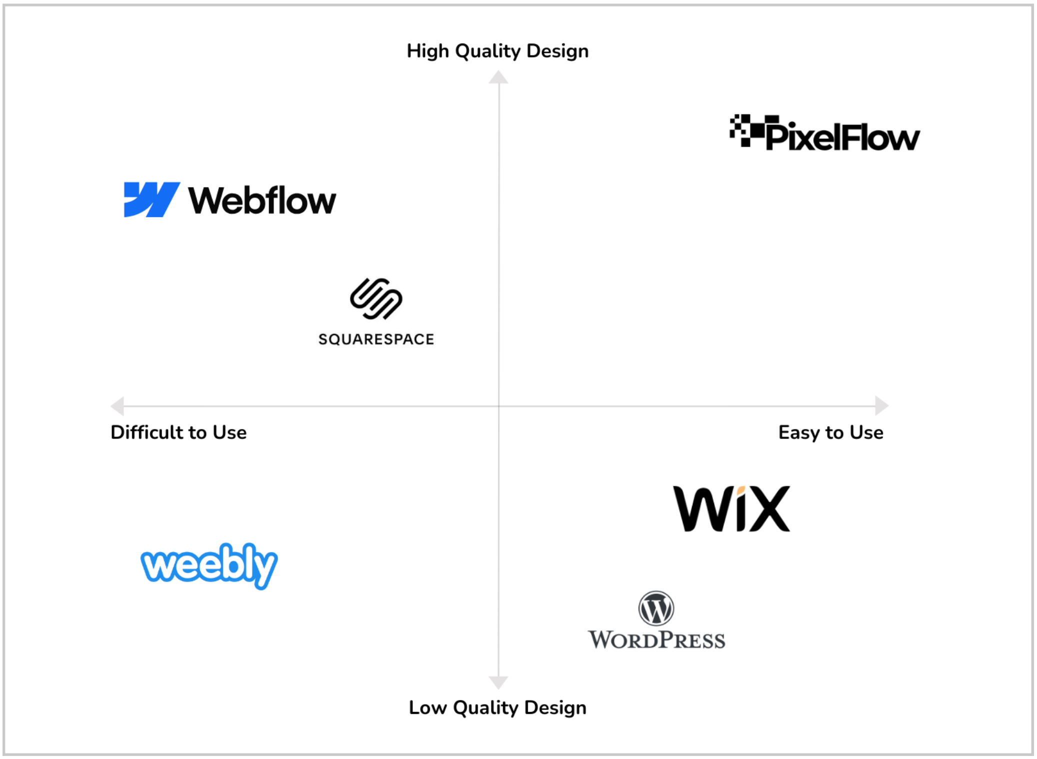


Figure 2: Competitive Landscape

PixelFlow exists at the intersection of high-quality design and user-friendliness. It is designed to address the key issues faced by users of other platforms. It offers the high-quality design that users might find in Webflow [6] or Squarespace [7], without the associated complexity. By offering a straightforward yet powerful design experience, PixelFlow offers a solution for those who find Wix [8], Wordpress [9] or Weebly's [10] capabilities too limiting. In essence, PixelFlow is carving out a niche by combining professional design quality with an ease of use that other platforms have yet to fully achieve.

1.4 Stakeholders

The primary stakeholders of this project are owner-operated businesses, for example BER Assessors, Gas Installers or Construction Contractors, lacking a strong online presence due to the costs and complexity associated with traditional website development options. Additionally, the project implicates their end-users—customers of these businesses—who will interact with the website.

Having first-hand experience with these stakeholders allowed me to prioritise specific features that would be seen as necessary requirements. Leveraging my past experiences and gathering user feedback has been key to enhancing the usability and accessibility of the project. This approach has allowed the software to evolve according to real-world user needs, resulting in a more efficient and user-friendly application.

1.5 Project Management

Setting a realistic project scope was vital for the success of this project. With guidance from my project supervisor, we decided on achievable goals, identified the essential features to be built and how to prioritise time. The combination of generative AI integration, a multi-tenant architecture, content management system and a drag-and-drop editor, posed a great technical challenge.

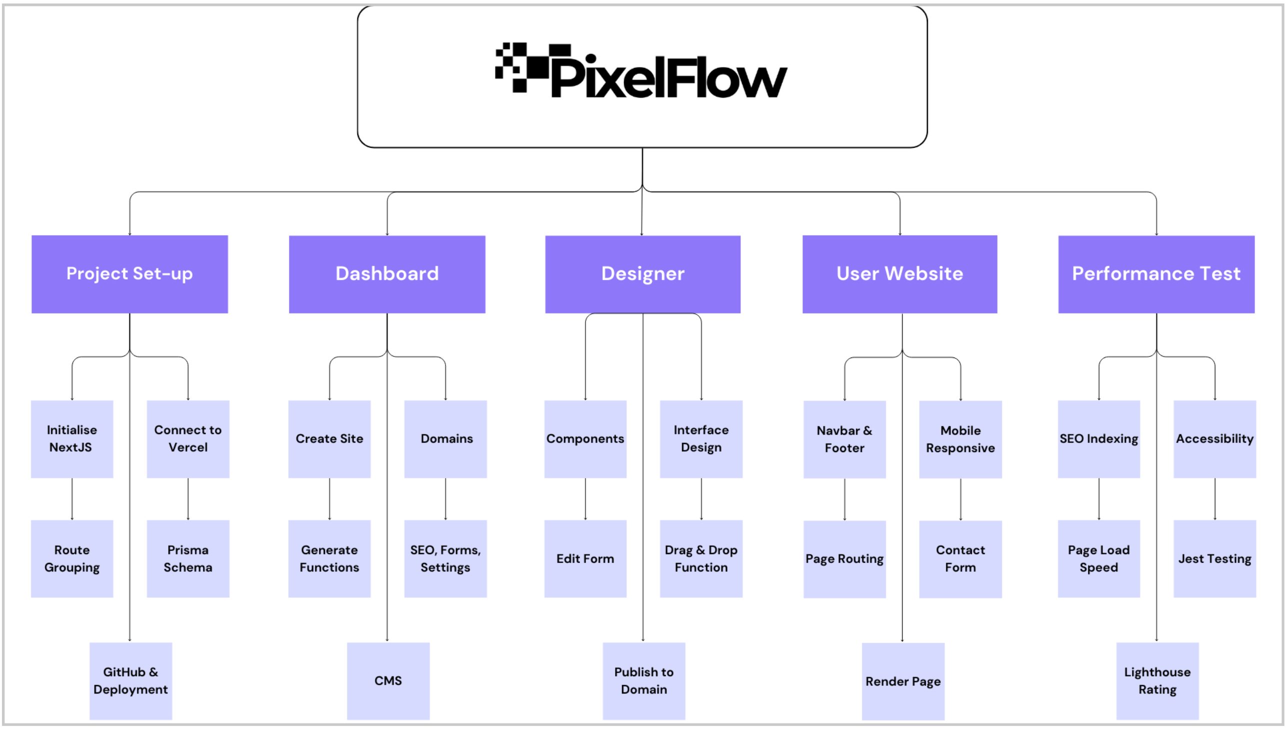


Figure 3: Compartmentalisation of Components

Git and GitHub [11]

For version control, my project relied heavily on Git and GitHub. The use of branching allowed for the development of features or bug fixes in isolation, which ensured the master codebase remained functional and rollbacks were possible if required. Clear commit messages were used to outline the update being pushed. Visual Studio Code (VSCode) was used as the Integrated Development Environment (IDE).

Notion [12]

Notion was used in project documentation and task management. It served as a central hub for organising tasks, notes, resources, code explanations and more. Complex code files were documented and explained for future reference, essentially creating a record of the project's architecture and design choices. I was able to share Notion environments with my project supervisor, providing a transparent view of the project's progress.

Agile Methodology

The project originally adopted an Agile methodology, specifically the use of sprints for structured development. Before beginning development on a given day, I would hold a "self-standup" meeting, where I would prioritise tasks to be completed. It was difficult to determine the required time necessary allocate to specific tasks, due to their complexity. The project was ultimately delivered without compromising on quality or scope.

Below is a high-level overview of the major tasks which were to be completed. Each of these were further broken down into sub-tasks for completion and tracked in Notion.

Date	Task/Milestone	Details
1st Nov	**Project Set-up**	Set up project repository. Initialise NextJS setup.
7th Nov	**Prisma**	Initialise Prisma schema with connection to Vercel.
14th Nov	**Dashboard**	Implement Clerk Auth and begin dashboard layout.
1st Dec	**Create Site**	Using a modal, create a new site associated with the user.
21st Dec	**Pages**	Site can now be created and pages successfully rendered.
7th Jan	**Drag/Drop**	Site now fully editable and DnD Kit implemented.
14th Jan	**CMS**	Add facility for adding "page collections".
1st Feb	**Domains**	Integrate custom domain functionality.
14th Feb	**GPT Integration**	Create an automated method of creating sites.
1st March	**Finalise**	Bug or UI fixes in preparation for project completion.

** Testing was continuously carried out during the development process.*

1.6 Technology Choices

Next JS [13]

Next.js offers a powerful route grouping system that enables the organisation of an application's routes into distinct categories. By using the file-system based routing mechanism, it is simple to create nested routes and dynamic paths. With Next.js, when a request targets a specific route group, only the relevant route is loaded. This selective loading will ensure that as the user base grows, the application's loading performance remains efficient, a primary reason why Next.JS was used for this project. This approach differs from the conventional full-application loading strategy employed by a standard Create React App setup.

Vercel [14]

Vercel Postgres is a serverless, fully-managed relational database built on open standards and designed to work with frontend frameworks, such as NextJs. It is built to scale automatically as the needs of the applications change, which ensures a high performance even when a large load is placed on the system. Vercel's tooling is intuitive and straightforward to use. It makes setting up and managing databases very easy. Vercel Blob was also used for media uploads and storage.

Prisma [15]

Prisma provides a seamless experience when interacting with the database. It handles connection pooling, caching and querying. It is used to simplify database workflows by turning database queries into simple function calls, thus leading to more secure and scalable code. It integrates with TypeScript to ensure that compile-time checks are passed for database queries. The Prisma schema is used to define the application's models in a way that makes their relationships clear and easily visualised.

Tailwind CSS [16]

Tailwind CSS offers an assortment of utility classes designed for efficiency and consistency when creating visually appealing websites. Unlike frameworks such as Bootstrap and Material UI, which offer styles for high-level components (like buttons, drop-downs, and forms), Tailwind adopts a more functional strategy by supplying utility classes for component construction. ShadCn-UI's open-source component library was also used.

Clerk Authentication [17]

Clerk provides comprehensive user authentication management capabilities through its suite of inbuilt functionalities, including user-specific account buttons and dedicated sign-in/sign-up pages. By wrapping protected routes with the ClerkProvider from @clerk/nextjs, secure user session management is ensured across the platform. This allows for precise control over access permissions. Only verified, authenticated users can interact with their respective site data, safeguarding operations like creating, updating, and deleting of database records.

Integrations

SaaSCustomDomains [18] provides a simple API for integrating custom domains into applications, regardless of underlying technology stack. Resend API [19] offers an easy way to programmatically send emails, and is used in contact forms on the sites. OpenAi API [20] is used in the text generation for websites. Unsplash API [21] is used to add royalty-free stock images to the sites.

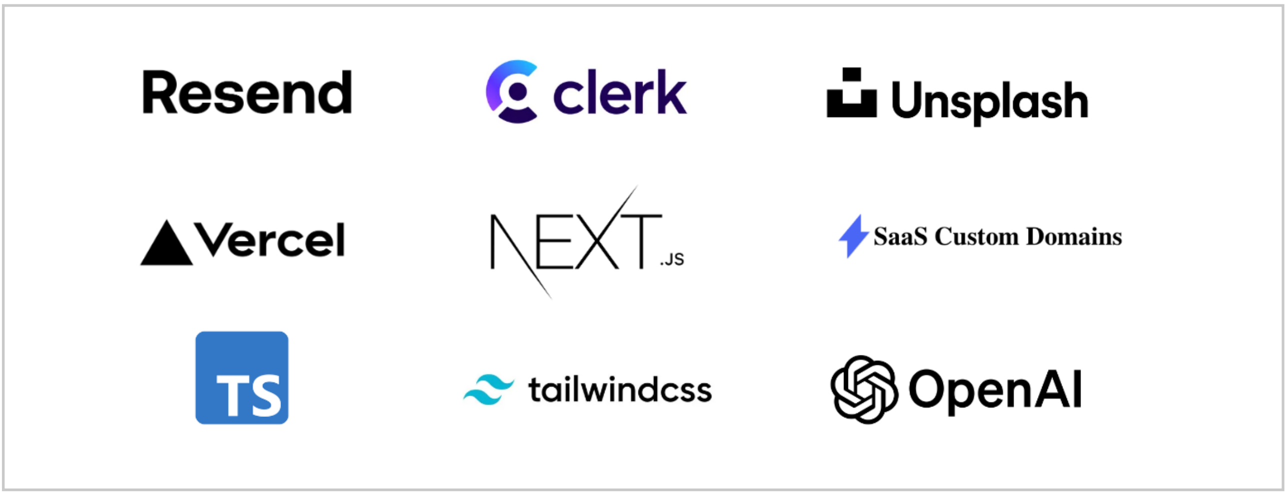


Figure 4: Technology Stack

Section Two - Requirements

2.1 Introduction

This section outlines the core fundamental needs and expectations for the PixelFlow platform. It details the functional and non-functional requirements, ensuring a comprehensive understanding of what the software must accomplish and how it should perform. Clearly defined requirements are used in industry to enable developers and stakeholders to align their goals regarding the software's design, development, and implementation phases.

2.2 Functional Requirements

Functional requirements define what a system is supposed to do. In this section, we are defining the specific behaviours and functions required for the successful completion of the project. These requirements focus on the interaction between the software platform and the end-users.

Landing Page: Acts as the entry point into the application. It allows the user to create a new account or log into an existing one. User credentials and session data is effectively managed to maintain security and confidentiality.

Dashboard: Create a website by specifying basic information such as name, description, about and services. Input SEO-related information such as title tags, descriptions, favicon, open graph to improve their website's visibility in search engine results. Allow users to associate custom domain names with their website, view contact form submissions and upload media assets.

Designer: The Designer provides a visual interface for editing pages or adding new components. This allows users to easily update the layout and content of their website without needing technical coding knowledge. Edit the content and design of individual pages, with Drag and Drop components. Add, arrange, and remove components on a page using drag-and-drop functionality.

User Website: The end user’s website will be functional, mobile responsive, always accessible to customers and quick to load. It will effectively showcase and act as a strong representation of the business.

2.3 Non-Functional Requirements

Non-functional requirements detail how the system performs tasks, rather than the tasks themselves, instead focusing on the user experience and system attributes such as reliability, efficiency, and quality.

Usability: The interface must be intuitive and comprehensive. A lot of consideration was given to the application design in order to facilitate an easy and efficient user experience, ensuring stakeholders are not left confused or frustrated. Significant care was given to correctly handling application errors and displaying these to the user.

Performance: It is important for modern applications to optimise for load speed and responsiveness. Steps for ensuring minimal latency, suspense and appropriate fallbacks are further outlined in the following sections.

Reliability: The system must perform under different conditions, including accounting for unique edge cases. This is done to maintain availability, ensuring users can depend on the application and they will not miss out on business due to an unresponsive website.

Discoverability: The application must be designed to be easily found by search engines., an important aspect of any well-performing website. This was achieved through using metadata best practices and other SEO techniques.

Security: It is vital we protect user data and prevent unauthorised access. This was a primary factor in deciding to use Clerk for authentication and enforcing privacy controls for sensitive information. Users should have trust in the application and be confident their data is kept safe from bad actors.

Scalability: The architecture should support growth in user numbers and geographical expansion without performance bottlenecks.

Maintainability: All code should be well-documented, separated by function and robustly version controlled. This allows for easy bug fixes, rollbacks and feature additions.

2.4 UX/UI

Being a website design and development platform, it was important from the start to ensure the application maintained a clear UI/UX. Application design is not only about the software's outward appearance, it also has a rational basis in influencing a user's trust and confidence in the product.

Prototyping

The prototype phase began by firstly defining the features and functionality scope to be achieved. A list of the functionality required was made and ordered by priority. Paper sketched wireframes of how the site would look and be laid out were created. This helped to form a clear understanding of how the application's structure would flow and how different components would work together to achieve the key functionality. After sketching the overall layout of the application, a workable prototype was created using Webflow. This gave an idea of what the site would look like in a live browser context.

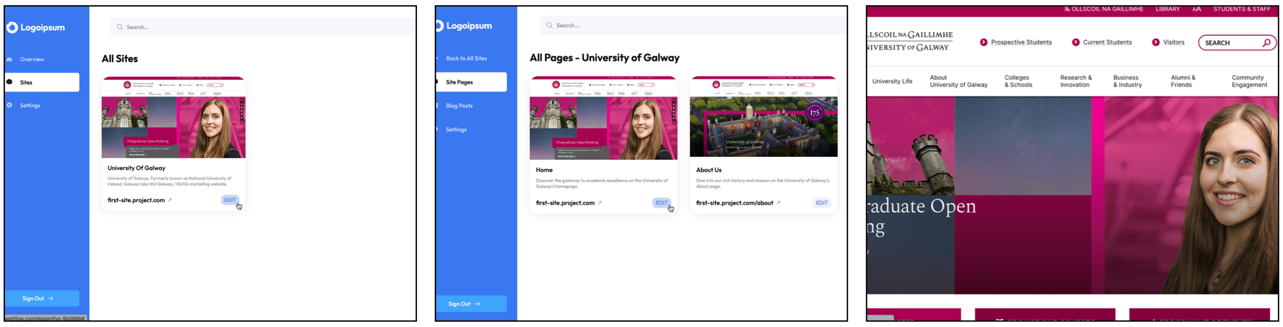


Figure 5: Original Prototype from Webflow

User Testing and Design Choices

Throughout the development of this application, considerable attention was paid to iterating based on user feedback, modern design standards and the principles of visual brand identity. The goal was to create an aesthetic and functional experience for the end user. The brand strategy involved the development of a PixelFlow logo, a consistent colour palette across the application, and a set of reusable components to maintain coherence. The font was chosen because of its readability and modernity. Graphics are used in the application not just for decoration but also as components to help in the comprehension of the textual content, for example, when a website has not been created yet, a graphic of someone building a website is shown.

2.5 Development Considerations

Optimisations

When attempting to optimise performance and latency of the application, it was important to consider the implications of caching. By caching frequently accessed data, it is possible to significantly reduce the computational load on the server and ensure the platform remains responsive. Content Delivery Networks (CDNs) are provided by Vercel to cache static content at different data centres across the world, typically in closer geographic proximity to the end-client. The revalidateTag function allows the application to purge cached data on-demand for a specific cache tag. In the context of a website development platform this is very important, so that users can see their changes automatically without having to clear their cache first. Vercel also provides Font and Image optimisation as standard. This removes external network requests for improved privacy and performance, and ensures faster load times for end users.

Metadata

Another important consideration with be the handling of Metadata across the platform. In the context of web development, optimised Metadata plays an important role in how a website is perceived by search engines and other external platforms. Each website's Metadata is used to define information about the underlying structure and content of the site. These include items such as; title tags, open graph images, favicon and more. Each website automatically constructs a sitemap.xml file, which can then be read by search engines crawlers for indexing.

Suspense

Another important consideration was suspense and loading states. An instant loading state is fallback UI that is shown immediately upon navigation. It indicates to the user that the application is working and loading, which increases engagement. The loading states in this application include placeholder cards, loading dots and image blur hashes, all designed to maintain a smooth and engaging user experience while minimising the perception of wait times.

2.6 Constraints

In progressing through the development of PixelFlow, several constraints emerged, directly impacting the scope and execution. Strategic decisions were made in order to deliver a fully-functional minimal viable product.

Scope

Defining the project's scope was a critical step to ensure overall focus and feasibility. The project began with a vast array of additional features to be implemented. However, looking at the project timeline, skillset available, and resources led to a more refined feature set. Core functionalities that directly addressed the needs of the target users were prioritised, which ensured a solid foundation to which additional features can be added at a later time. The combination of generative AI integration, a multi-tenant architecture, content management system and a drag-and-drop editor, posed a great technical challenge in itself.

Quality

Ensuring a high-quality experience for the end user in both the user interface (UI) and user experience (UX) was a primary concern, given the project's aim. The need to focus on seamless front-end design and back-end functionality posed a significant challenge. Considerable attention was spent on making the application behave like a real-world software application, extending to branding, font selection, colour palette and more.

Time

Being a single-person project, time was the biggest constraint faced. Without prior experience working directly with the technologies involved, it was hard to provision the correct amount of time that an individual feature would take or how technically difficult it would be. This was coupled was an intense assignment workload from other course modules. A lot of time was dedicated over Christmas break to getting the project functional.

Section Three - System Design

3.1 Introduction

In this section, we will discuss the intricate details of the software system design. Through this comprehensive overview, the reader will gain an understanding for how the components, data and structural elements of the platform come together to meet the end-user's requirements.

3.2 Overview

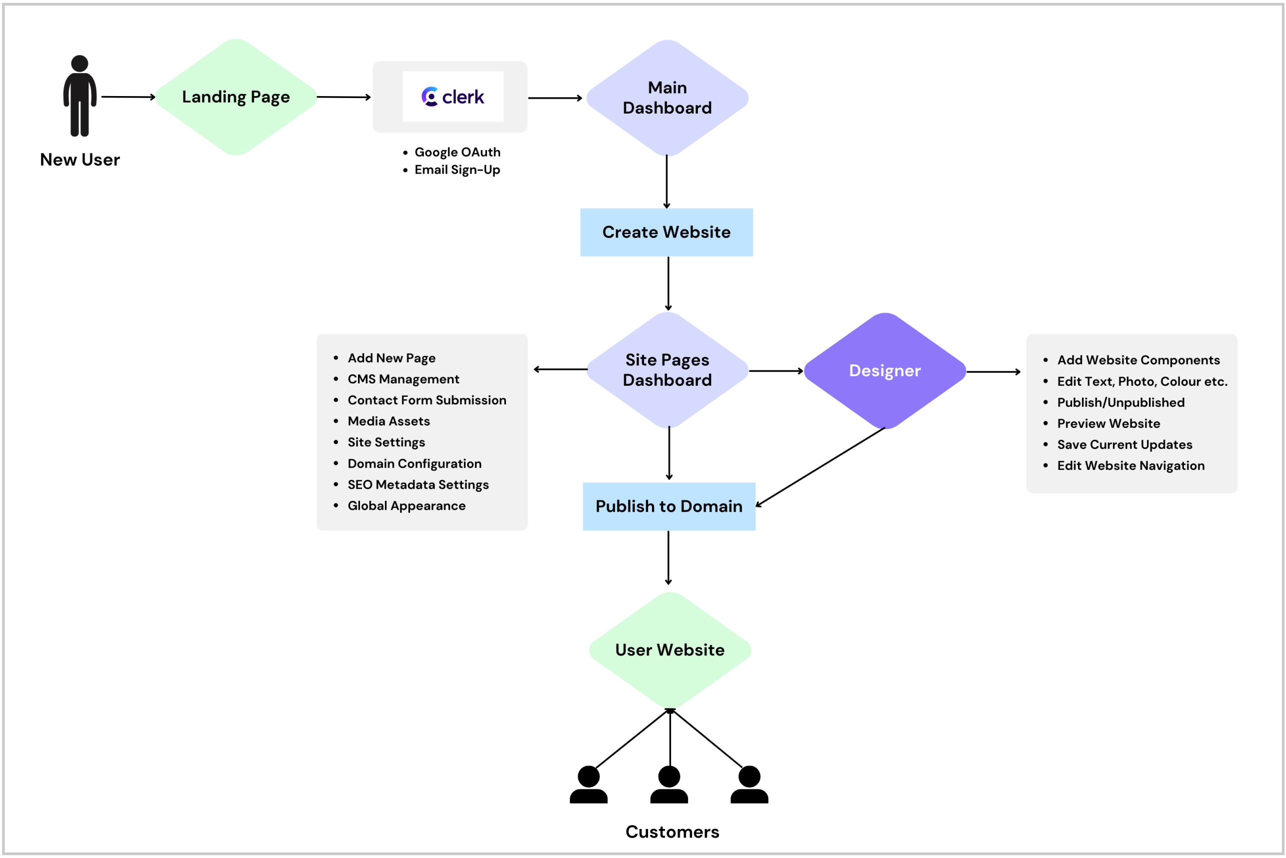


Figure 6: PixelFlow Framework

The PixelFlow project is structured into four main categories for organising its routes: i). Home, This category is dedicated to the website's landing page. ii). Auth - This section manages the routes related to user authentication. iii). App - This area handles the internal routes of the application, which includes the dashboard and page designer features. iv). Domain - This category is responsible for displaying the client's website on the internet.

Database Structure

The project's Prisma schema defines multiple models: Site, PageType, SitePage, ContactForm, FormSubmission, and Asset, each used for different aspects of the application's data.

The Site Model is the primary model of the application and is used for controlling the majority of the data throughout the project. All other models reference back to the singular Site model, either in a many-to-one relationship or a one-to-one relationship. It is also responsible for holding data such as the custom domain identifier, SEO metadata and any other data that is used across all aspects of the client website.

The SitePage Model is responsible for individual pages within the website. These pages can be grouped into different PageTypes, which allows for a content management system (CMS) to be used.

The ContactForm, FormSubmission and Asset models are relatively straightforward. A user can have a single contact form and receive multiple form submissions. The Assets model stores the various media used on the site, ie. images, video etc.

Middleware

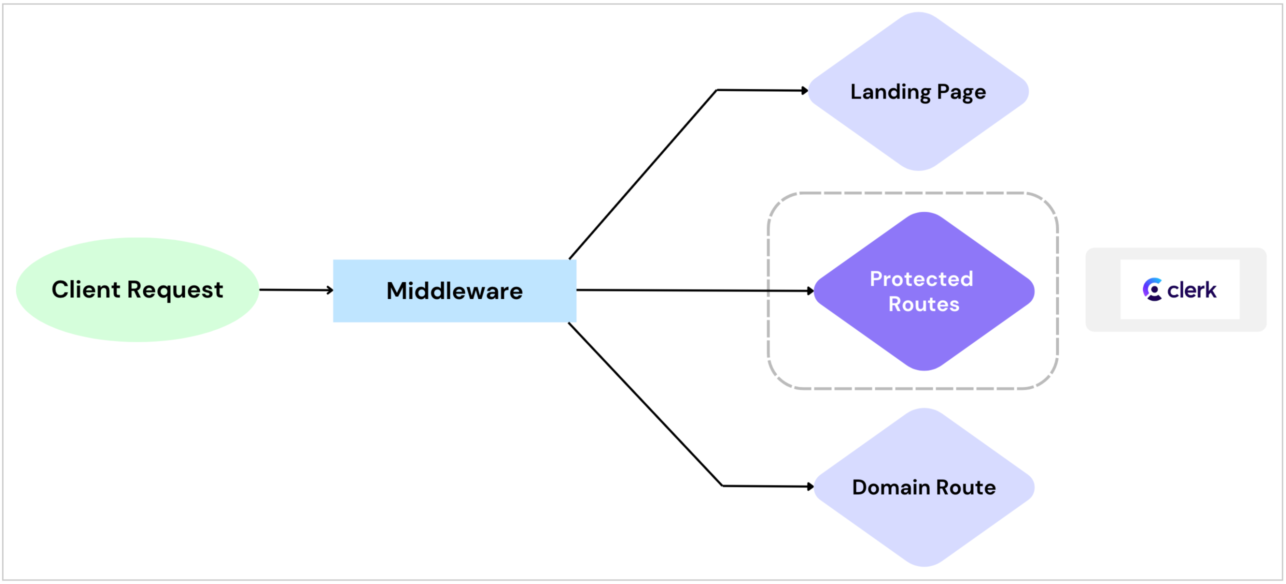


Figure 7: Middleware Workflow

The middleware file is used to control all requests directed at the application. It is crucial in how the application routing and structure works. It is used in allowing for custom processing steps like authentication, redirections, or modifying requests or responses. In this project, we are using Clerk for user authentication and management. By using the Clerk middleware, the project can securely manage auth states and protect routes, while still allowing for public routes to be defined and visited.

The middleware function takes in a request and event. The URL is extracted from the request, as well as the Headers associated with that request. The X-Served-For header is added by SaaSCustomDomains and allows us to identify when the request is coming from a custom website domain, rather than from within the application. We extract the path or slug associated with the request to be able to correctly rewrite the URL for the request made by the user.

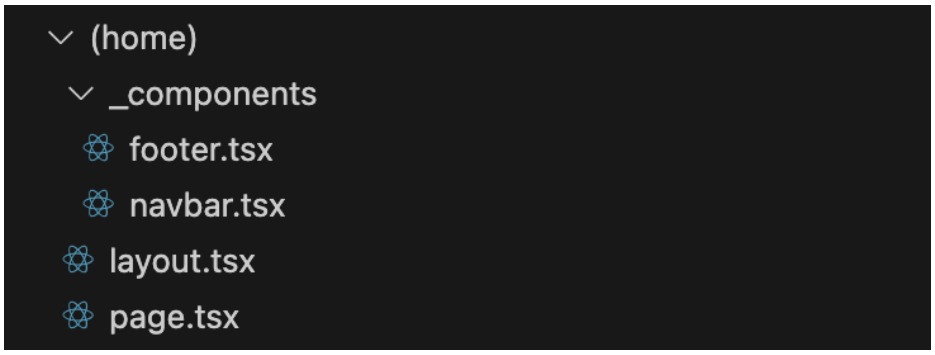
If the request comes from the application itself, such as from localhost or from the environment variable defined NEXT_PUBLIC_ROOT_DOMAIN, then we know that the user is not coming from a custom domain but rather trying to access the application.

Public Routes are used to define routes in the application which can be accessed without being authenticated. These are used in our landing page and domain routes for the application and allows it to be public-facing while remaining secure.

<u>Root Layout File</u>

The Root Layout file wraps the application, which is important for allowing providers to have the correct access project-wide. In this project, we need to wrap the application with a ClerkProvider, Modal Provider and Toaster Provider. This is important to avoid hydration errors, which occur when the client and the server of the application are out of sync. Important project Metadata is also defined in this file, including title tag, description, favicon, and open graph image.

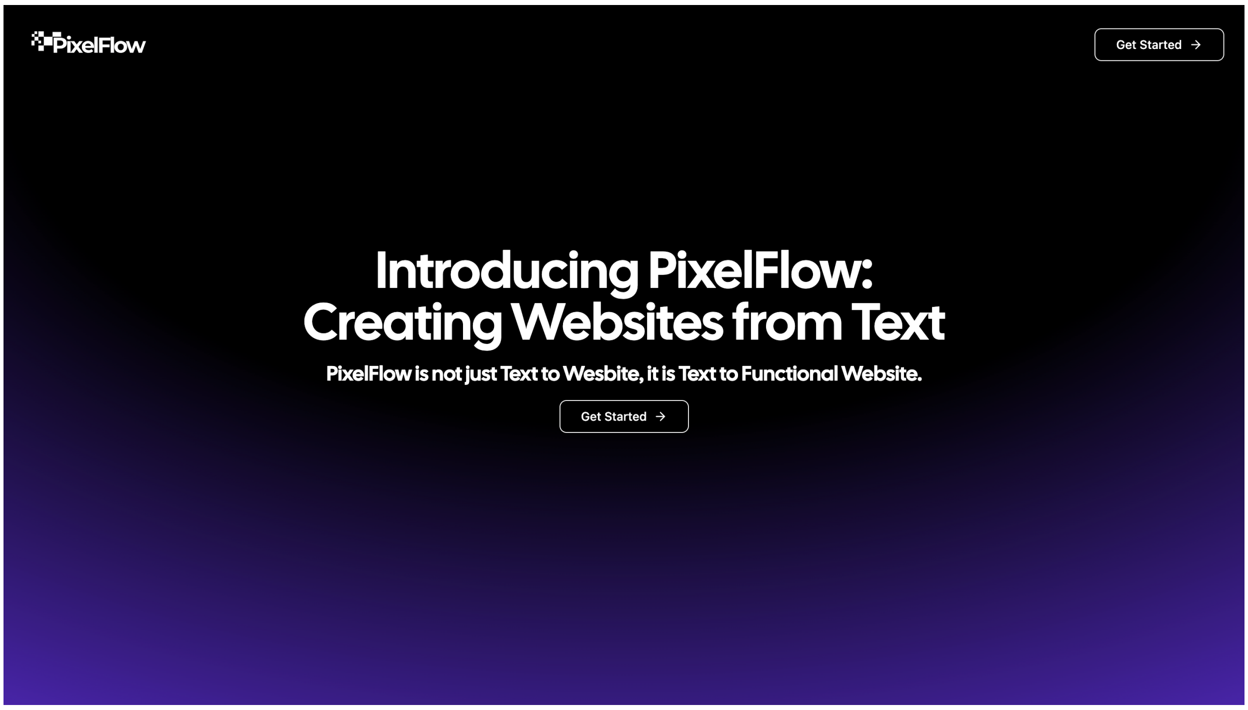
3.3 Feature One - Landing Page



```
(home)
  _components
    footer.tsx
    navbar.tsx
  layout.tsx
  page.tsx
```

Figure 8: Home Route Group

The home route group is used for the marketing or public-facing website associated with the project. In this case, it is just a landing page, but it could be built out to have many different pages and routing between them, for example an "About Us" page could be added by adding a new folder called about, with a page.tsx file. The code in these pages is straightforward, mainly consisting of simple Tailwind CSS classes with little functionality.

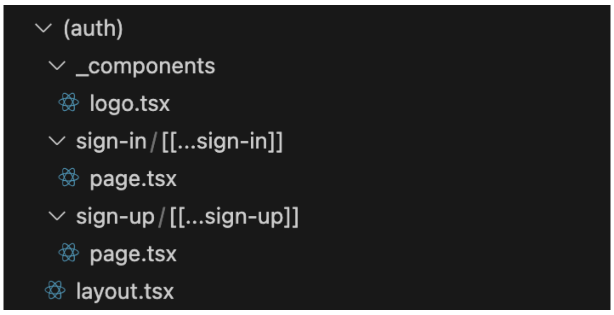


Introducing PixelFlow: Creating Websites from Text

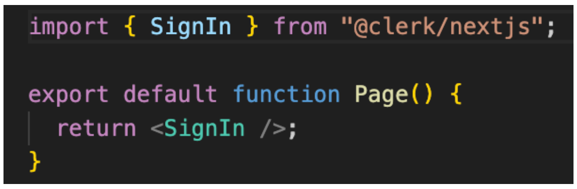
PixelFlow is not just Text to Wesbite, it is Text to Functional Website.

Figure 9: Acts as the entry point into the application.

3.4 Feature Two - Authentication

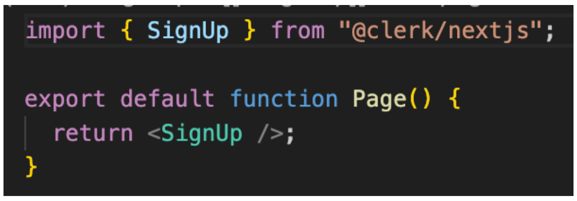


```
(auth)
  _components
    logo.tsx
  sign-in / [[...sign-in]]
    page.tsx
  sign-up / [[...sign-up]]
    page.tsx
  layout.tsx
```



```tsx
import { SignIn } from "@clerk/nextjs";

export default function Page() {
  return <SignIn />;
}
```



```tsx
import { SignUp } from "@clerk/nextjs";

export default function Page() {
  return <SignUp />;
}
```

Figure 10: Auth Route Group - SignIn and SignUp Components

The ClerkProvider is used to provide authentication management across all different pages of the project. The Clerk appearance property is set to dark, which matches with the branding of the application. Clerk provides built-in authentication pages for sign-in and sign-up which can easily be imported using import { SignUp, SignIn } from "@clerk/nextjs". This simple structure within the application allows for authentication to be handled very easily. The layout.tsx instructs the children pages to take up the full screen.

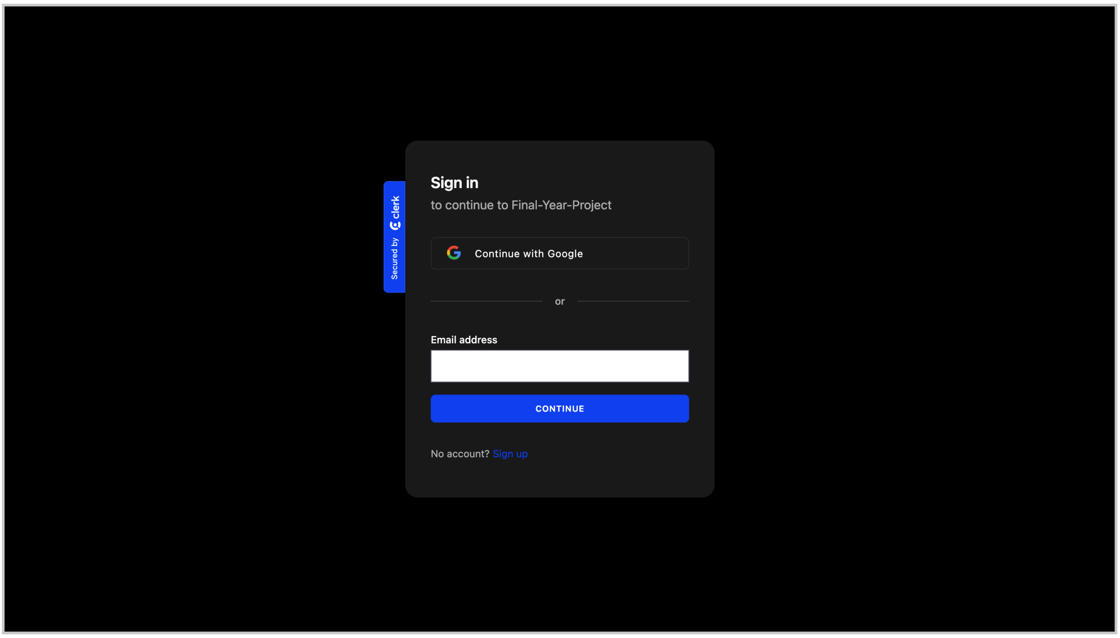
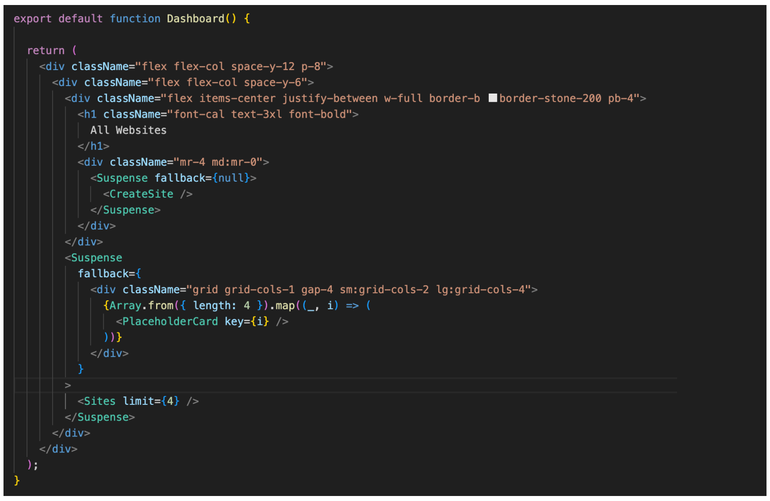


Figure 11: Clerk Authentication to manage user authentication and session data.

3.5 Feature Three - Dashboard

The main Dashboard allows the user to create a new site and/or lists any site that the user has created previously. As the websites are loading, placeholder cards are shown on the screen. These are simply created using Tailwind CSS classes and load very quickly, which indicates to the user that the application is responding.

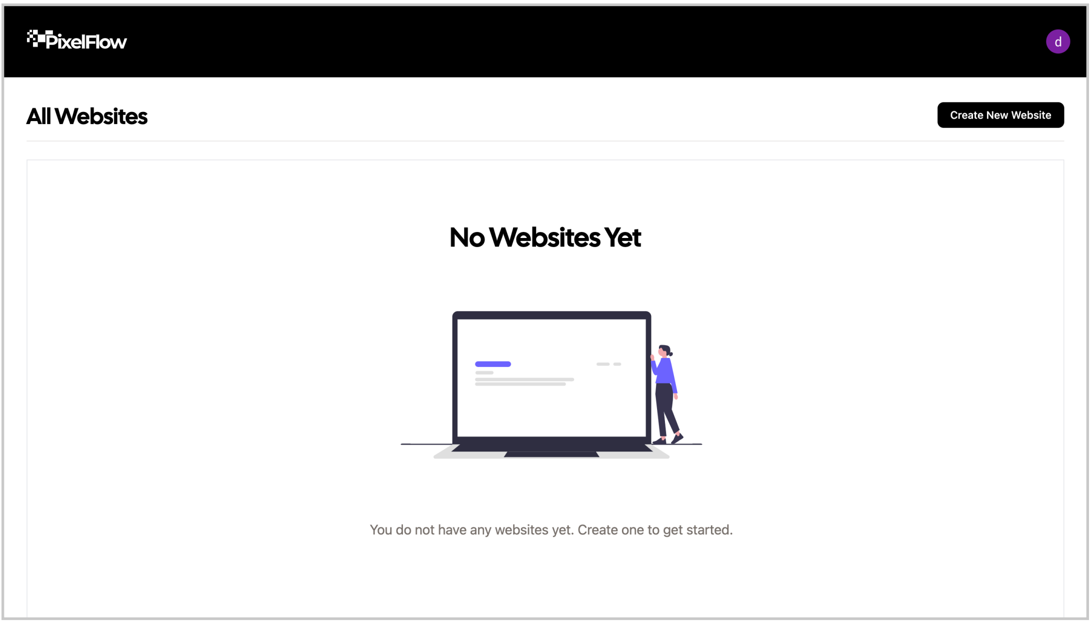
The <Sites /> component checks that the user is logged in, and then queries the database for all Sites associated with that given user's unique ID. If no Sites exist for that user, an image and text prompt are displayed telling them to create a new site.



```
export default function Dashboard() {

  return (
    <div className="flex flex-col space-y-12 p-8">
      <div className="flex flex-col space-y-6">
        <div className="flex items-center justify-between w-full border-b border-stone-200 pb-4">
          <h1 className="font-cal text-3xl font-bold">
            All Websites
          </h1>
          <div className="mr-4 md:mr-0">
            <Suspense fallback={null}>
              <CreateSite />
            </Suspense>
          </div>
        </div>
        <Suspense
          fallback={
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2 lg:grid-cols-4">
              {Array.from({ length: 4 }).map((_, i) => (
                <PlaceholderCard key={i} />
              ))}
            </div>
          }
        >
          <Sites limit={4} />
        </Suspense>
      </div>
    </div>
  );
}
```

Figure 12: Dashboard Page.tsx



You do not have any websites yet. Create one to get started.

Figure 13: Dashboard entry point as a new user.

3.6 Feature Four - Create Site

The <CreateSite> component passes the <CreateSiteModal> component as a child property to <CreateSiteButton> - in this way when the button is pressed the modal can be opened with the form and shown to the user. When the user submits the form, it passes the data object to createSite() and waits for a response. The createSite() function on the back-end contains external calls to OpenAi's API and Unsplash API to populate the site data. Currently, each generated site will have the same starting structure and these text prompts are hard-coded. This is an area of the site that could be improved over time to allow for multiple different templates, depending on the business type.

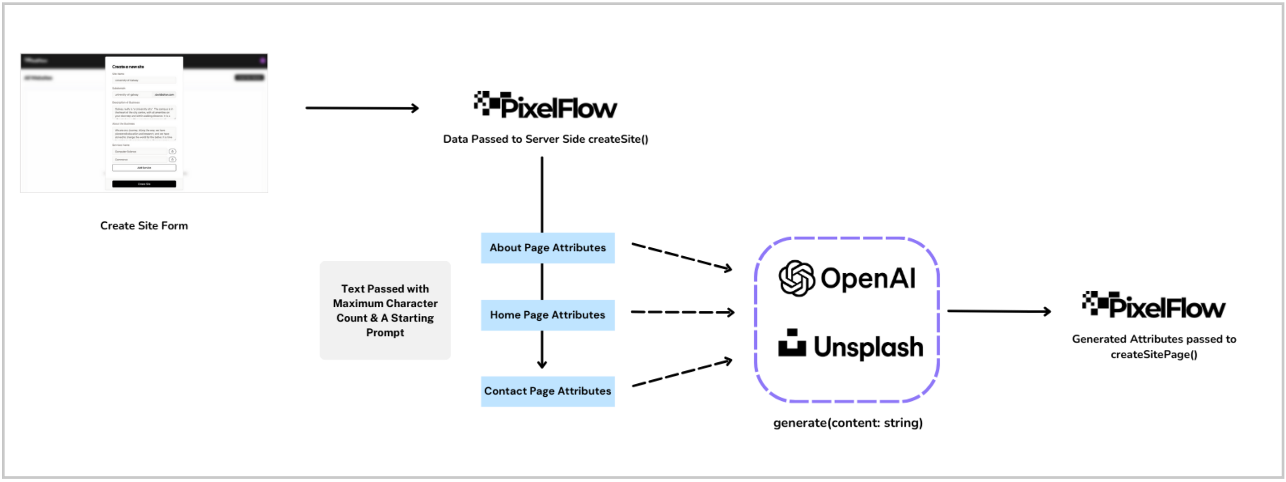
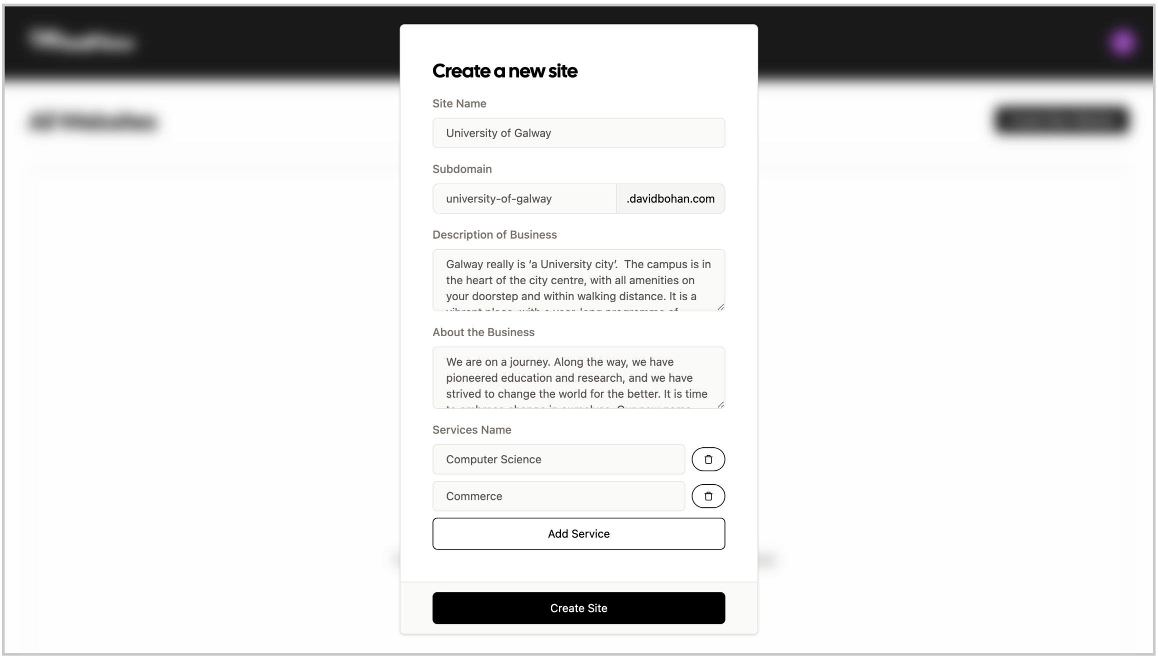

Create Site Form
PixelFlow
Data Passed to Server Side createSite()
Text Passed with Maximum Character Count & A Starting Prompt
About Page Attributes
Home Page Attributes
Contact Page Attributes
OpenAI
Unsplash
generate(content: string)
PixelFlow
Generated Attributes passed to createSitePage()

Figure 14: Generate Workflow



Create a new site

Site Name: University of Galway

Subdomain: university-of-galway .davidbohan.com

Description of Business: Galway really is 'a University city'. The campus is in the heart of the city centre, with all amenities on your doorstep and within walking distance. It is a

About the Business: We are on a journey. Along the way, we have pioneered education and research, and we have strived to change the world for the better. It is time

Services Name: Computer Science; Commerce

Figure 15: Create a website by specifying basic information such as name, description, about and services.

3.7 Feature Five - Individual Site Route

Once created, the individual Site route is accessed via /app/(dashboard)/site/[id]. This route contains the functionality used in maintaining that individual site. When this route is hit, the layout.tsx gets the site data by Param (id), and retrieves the SitePages and the different PageTypes associated with the Site. These are used for rendering the Main site pages and also grouping the CMS categories ie. Services, Blogs etc which are displayed in a Navbar. The grouped pages are displayed in app/(dashboard)/site/[id]/[slug]/page.tsx, where the slug is defined by the group type, ie. /services.

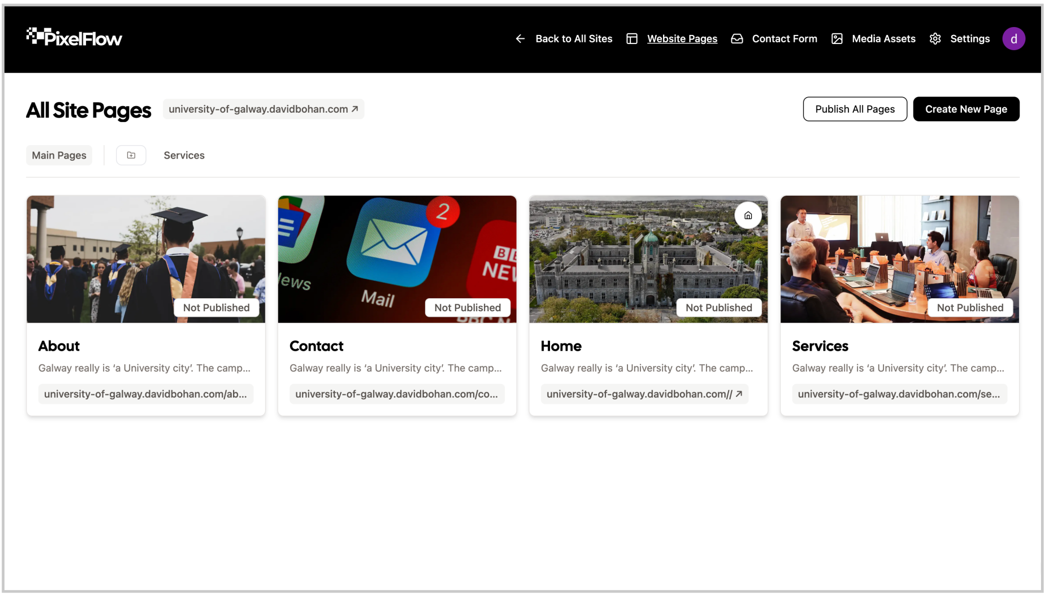

PixelFlow
Back to All Sites
Website Pages
Contact Form
Media Assets
Settings
d
All Site Pages
university-of-galway.davidbohan.com ↗
Publish All Pages
Create New Page
Main Pages
Services
Not Published
About
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com/ab...
Not Published
Contact
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com/co...
Not Published
Home
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com// ↗
Not Published
Services
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com/se...

Figure 16: The create site function initialises multiple starter pages.

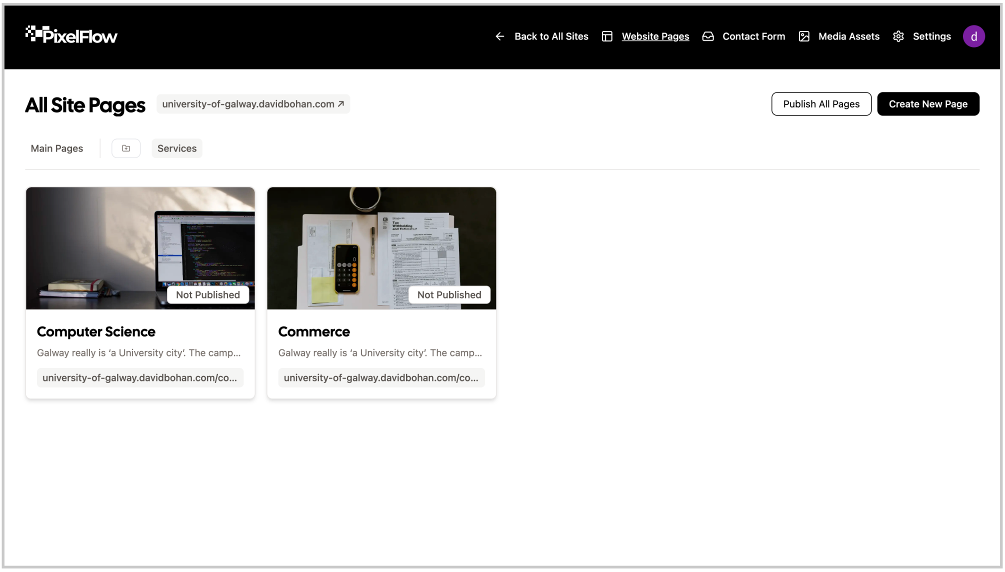
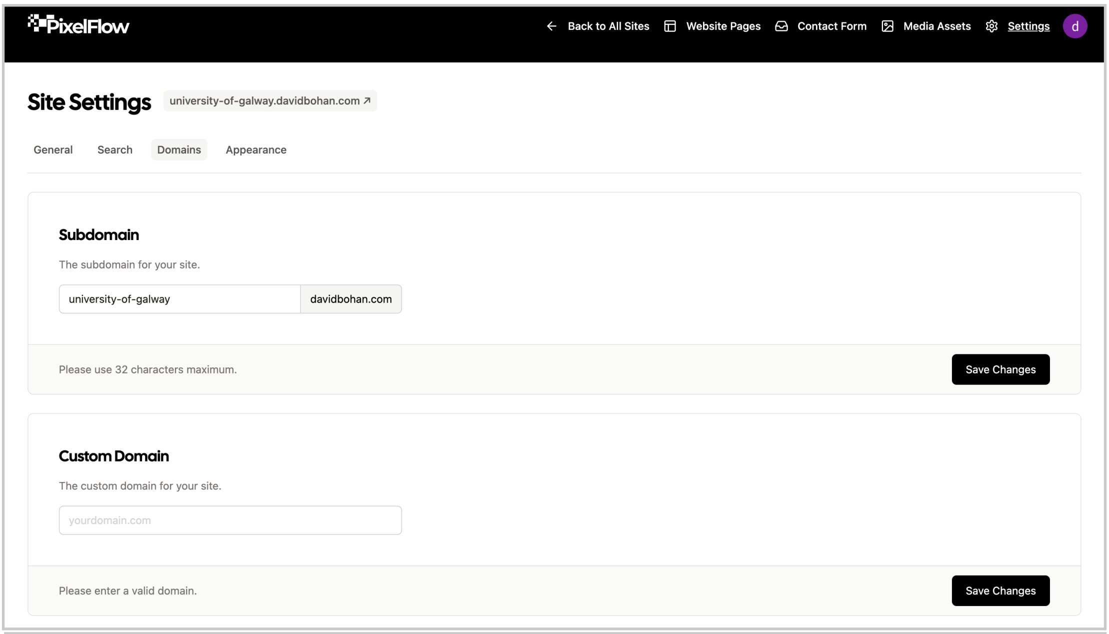

PixelFlow
Back to All Sites
Website Pages
Contact Form
Media Assets
Settings
d
All Site Pages
university-of-galway.davidbohan.com ↗
Publish All Pages
Create New Page
Main Pages
Services
Not Published
Computer Science
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com/co...
Not Published
Commerce
Galway really is 'a University city'. The camp...
university-of-galway.davidbohan.com/co...

Figure 17: Service collection also created, provides an improved navigation.

3.8 Feature Six - Site Settings



Site Settings university-of-galway.davidbohan.com ↗

General Search Domains Appearance

Subdomain

The subdomain for your site.

university-of-galway | davidbohan.com

Please use 32 characters maximum. Save Changes

Custom Domain

The custom domain for your site.

yourdomain.com

Please enter a valid domain. Save Changes

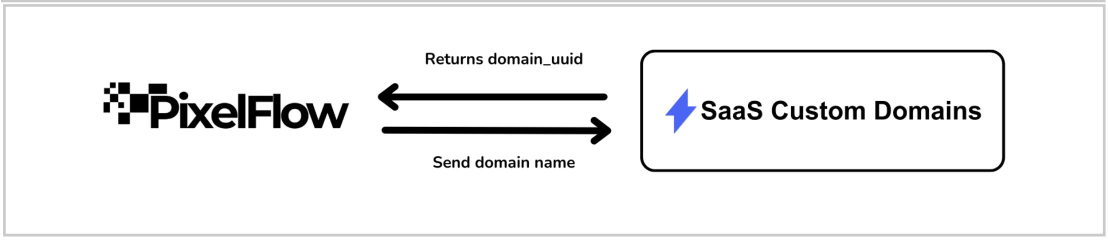
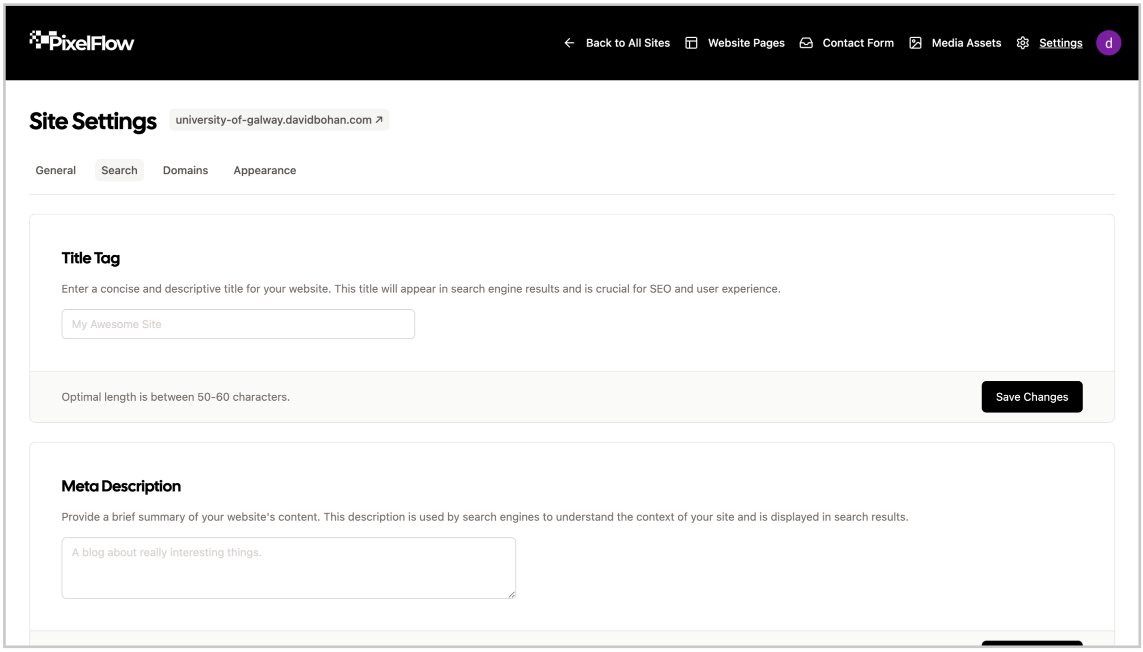


Figure 18: Allows users to associate custom domain names with their website.



Site Settings university-of-galway.davidbohan.com ↗

General Search Domains Appearance

Title Tag

Enter a concise and descriptive title for your website. This title will appear in search engine results and is crucial for SEO and user experience.

My Awesome Site

Optimal length is between 50-60 characters. Save Changes

Meta Description

Provide a brief summary of your website's content. This description is used by search engines to understand the context of your site and is displayed in search results.

A blog about really interesting things.

Figure 19: Input SEO-related information such as title tags, descriptions, favicon, open graph to improve their website's visibility in search engine results.

These settings actions all require the use of CRUD operations to make changes to the models in the database. This functionality was implemented by following the guidelines set-out by Vercel and Next Js. Vercel provides a robust form index.ts file, which can take in different parameters and perform different actions based on the HandleSubmit function passed to it. Based on the data type passed and its id, the index.ts file can correctly execute the correct server side function.

3.9 Feature Seven - Contact Forms

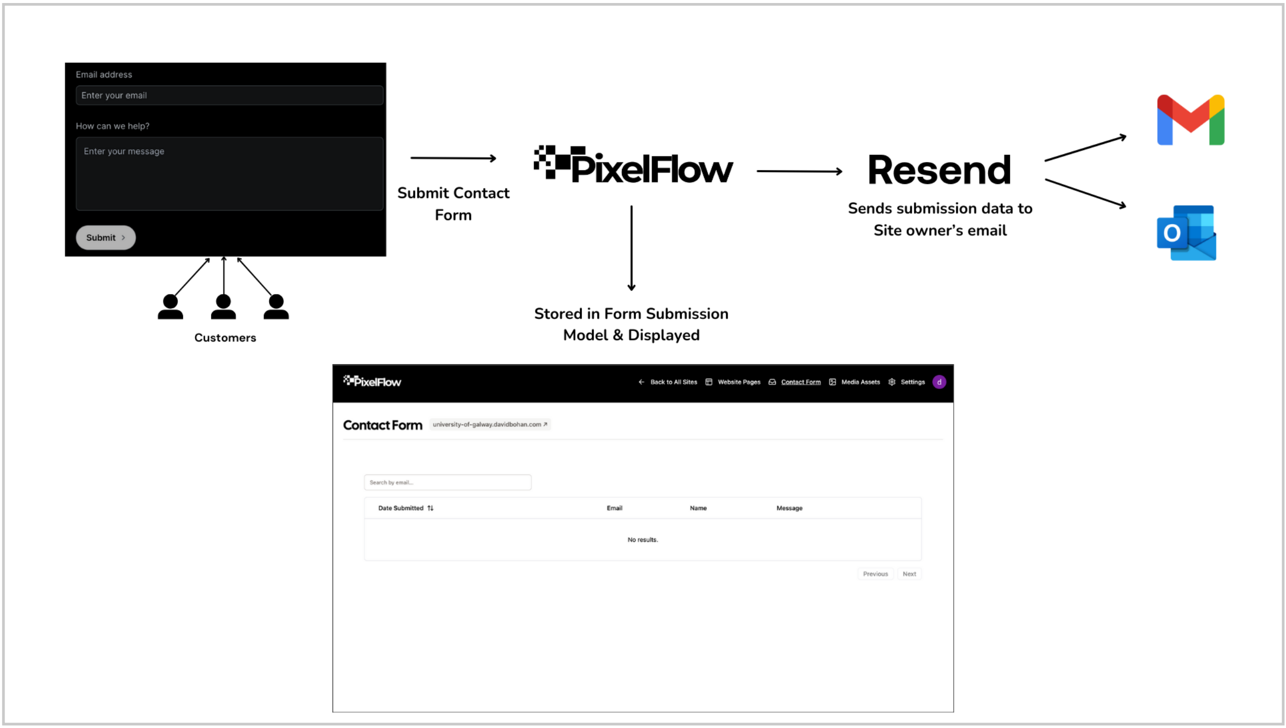


Figure 20: Contact Form Workflow

The Resend API is designed to streamline the process of email sending for developers, offering a straightforward and developer-friendly interface for sending emails. With Resend, developers can easily integrate email functionalities into their applications, including sending emails with attachments, using custom email headers, and tagging emails for organised tracking. The API also supports the sending of emails using React components, providing a unique approach to email template design. The API also supports the sending of emails using React components, providing a unique approach to email template design.

3.10 Feature Eight - Media Uploads

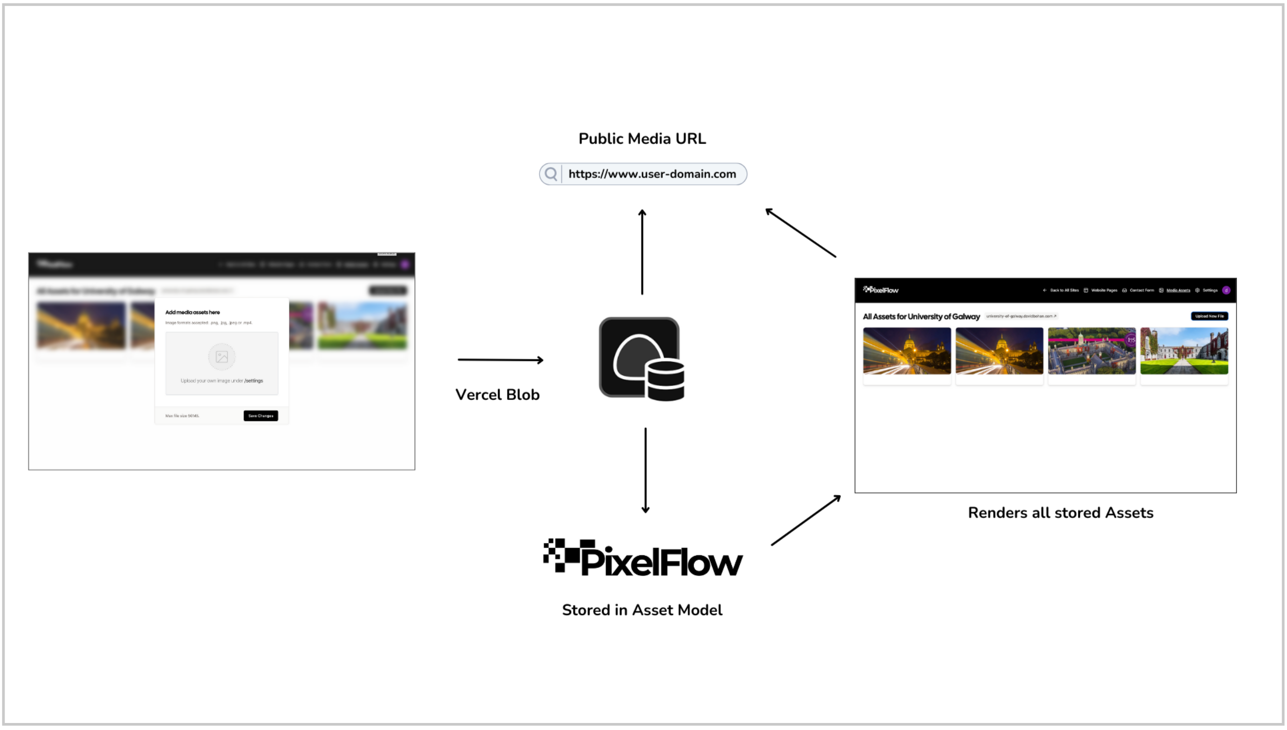


Figure 21: Media Assets Workflow

Vercel Blob is a cloud storage solution specifically designed to make storing and accessing files simple. It is particularly tailored for developers working with modern web frameworks, allowing the storage of any file type—ranging from images and videos to audio files—via an intuitive, promise-based API. This service integrates seamlessly with Vercel's compute products for both low-latency reads and high-throughput writes, offering developers a highly available storage option with minimal setup requirements. By providing immutable URLs for stored files, Vercel Blob ensures that developers can easily share and access their data without worrying about changes or deletions affecting their availability.

3.11 Feature Nine - Designer

The Designer provides a visual interface for editing pages or adding new components. This allows users to easily update the layout and content of their website without needing technical coding knowledge.

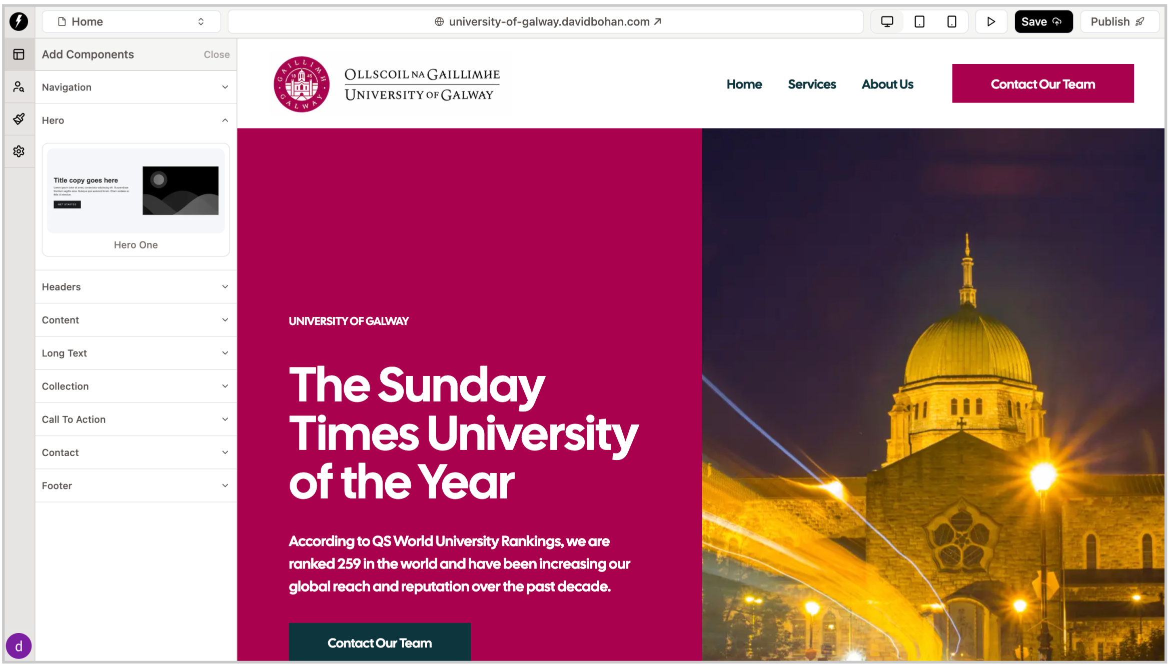


Figure 22: Edit the content and design of individual pages, with Drag and Drop components. Add, arrange, and remove components on a page using drag-and-drop functionality.

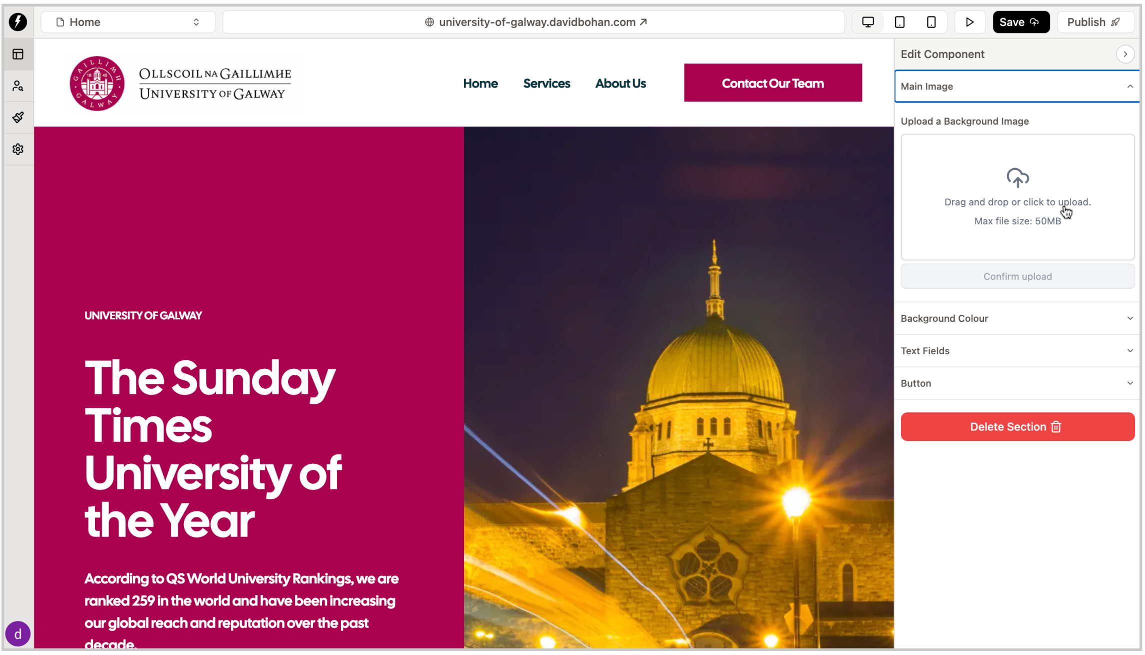
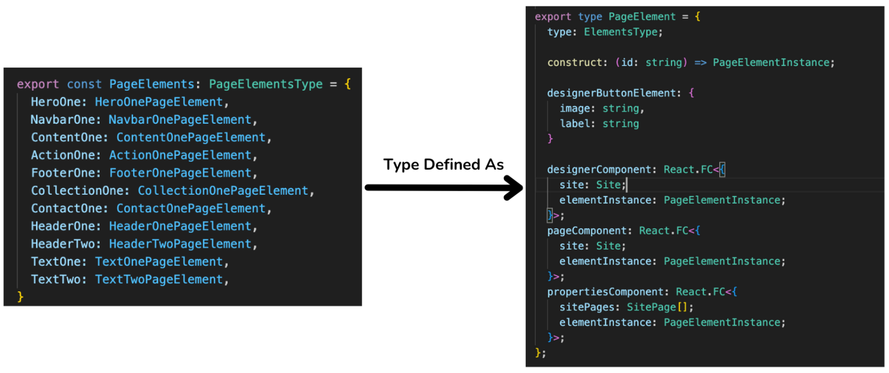

Home
university-of-galway.davidbohan.com
Save
Publish
OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY
Home
Services
About Us
Contact Our Team
UNIVERSITY OF GALWAY
The Sunday Times University of the Year
According to QS World University Rankings, we are ranked 259 in the world and have been increasing our global reach and reputation over the past decade.
Edit Component
Main Image
Upload a Background Image
Drag and drop or click to upload.
Max file size: 50MB
Confirm upload
Background Colour
Text Fields
Button
Delete Section

Figure 23: The Component Editor enables detailed customisation of individual website components. Can also view changes on different screen sizes, preview changes, save changes and publish changes.

PageElement Components

PageElements make up the building blocks of the user websites. The page-elements.tsx file the defines different types of page elements. It types all PageElements in the same structure, which includes an initialisation function, designerButtonElement for dragging into the editor, a designerComponent for showing the component in the designer, a properties Component for editing the attributes of the component, and a page Component to render on the actual website. The PageElementInstance is used to define a single instance of that element, with an ID, type and the extraAttributes associated with it. PageElements maps each ElementsType to its corresponding configuration (index.tsx file), which allows for easy creation, deletion, editing and rendering of components. This structure allows for a scalable, efficient solution to website building.



```
export const PageElements: PageElementsType = {
  HeroOne: HeroOnePageElement,
  NavbarOne: NavbarOnePageElement,
  ContentOne: ContentOnePageElement,
  ActionOne: ActionOnePageElement,
  FooterOne: FooterOnePageElement,
  CollectionOne: CollectionOnePageElement,
  ContactOne: ContactOnePageElement,
  HeaderOne: HeaderOnePageElement,
  HeaderTwo: HeaderTwoPageElement,
  TextOne: TextOnePageElement,
  TextTwo: TextTwoPageElement,
}
```

```
export type PageElement = {
  type: ElementsType;

  construct: (id: string) => PageElementInstance;

  designerButtonElement: {
    image: string,
    label: string
  }

  designerComponent: React.FC<{
    site: Site;
    elementInstance: PageElementInstance;
  }>;
  pageComponent: React.FC<{
    site: Site;
    elementInstance: PageElementInstance;
  }>;
  propertiesComponent: React.FC<{
    sitePages: SitePage[];
    elementInstance: PageElementInstance;
  }>;
};
```

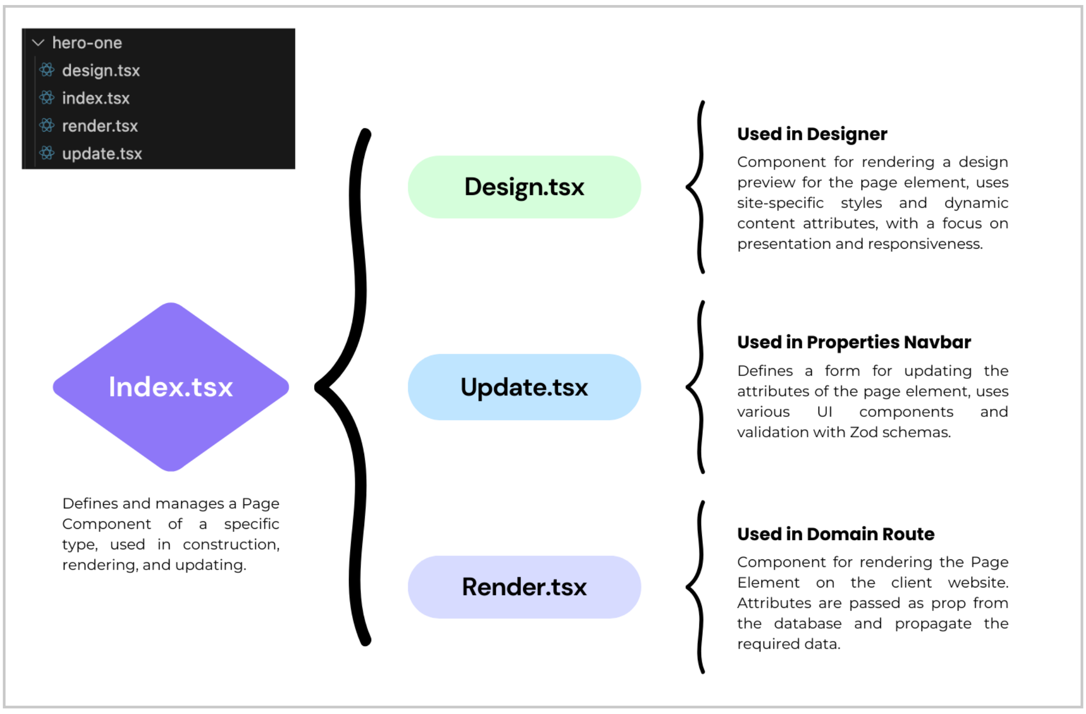

hero-one
design.tsx
index.tsx
render.tsx
update.tsx
Index.tsx
Defines and manages a Page Component of a specific type, used in construction, rendering, and updating.
Design.tsx
Update.tsx
Render.tsx
Used in Designer
Component for rendering a design preview for the page element, uses site-specific styles and dynamic content attributes, with a focus on presentation and responsiveness.
Used in Properties Navbar
Defines a form for updating the attributes of the page element, uses various UI components and validation with Zod schemas.
Used in Domain Route
Component for rendering the Page Element on the client website. Attributes are passed as prop from the database and propagate the required data.

Figure 24: PageComponents Structure

Designer Functionality

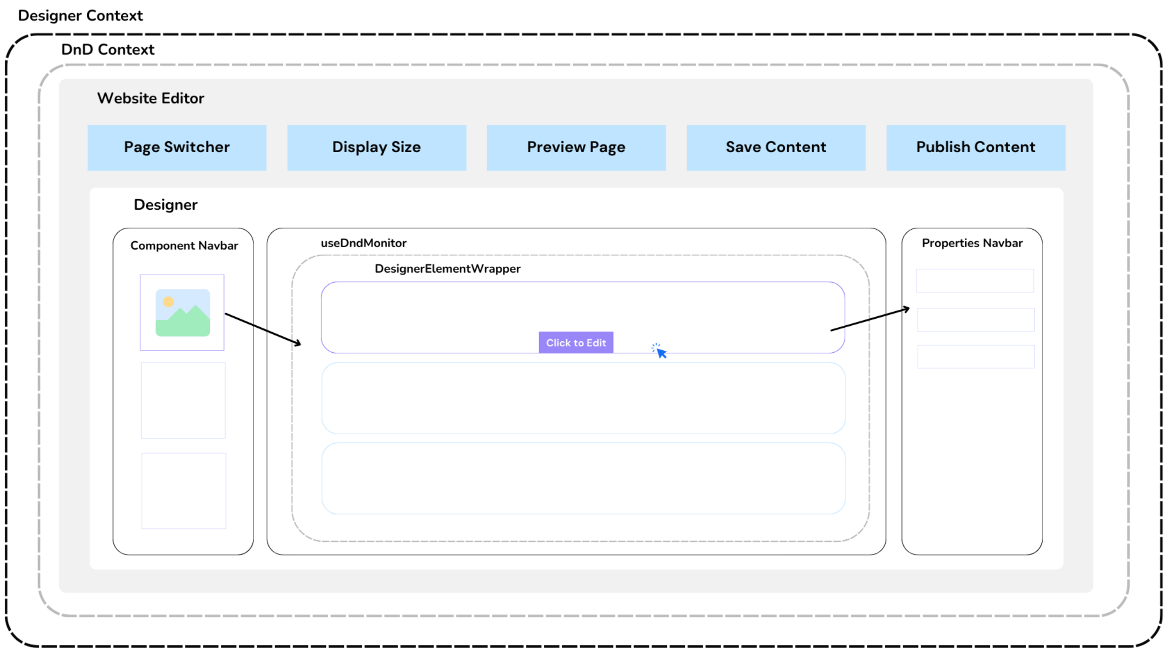


Figure 25: Designer Structure

Designer Context - The Designer Context Provider component is very important in managing the context of a given SitePage's components. It defines functions for the page structure; setElements takes in a list of page elements and stores them in a new array, enabling the loading of a page's structure. selectedElement is a state variable that stores the currently selected page element, facilitating operations like deletion or property editing on this specific element. addElement is a method that adds a new page element at a specified position within the elements array, allowing for precise placement in the page layout. removeElement is a function that finds and removes a page element from the array based on its unique ID, ensuring that specific elements can be precisely targeted and eliminated from the elements array. updateElement takes in an element ID and a new element. It searches the array for an element with a matching ID and sets the passed element to the element at that given index, thus facilitating element updates.

DnD Context - The DndContext from @dnd-kit/core is used for drag-and-drop functionality. Wrapping the Website Editor with DndContext allows for defining draggable elements (using useDraggable) and droppable areas (using useDroppable), which are a core feature of this application. It is used to manage the state and events. Passing the DnDContext sensors allow it to work across different input modes i.e. mouse, touch, or keyboard sensors.

Website Editor - The WebsiteEditor component renders many important components; including Page Switcher, Display Size Switcher, Preview Button, Save Content, and Publish Content. This component manages various states related to the editor readiness and initialisation. It loads the Navbar, Footer and SitePage content into a single element's array variable which is passed to setElements in Designer Context, allowing for the SitePage elements to be loaded into the Designer.

Designer - The Designer component is the primary Drag and Drop environment. It uses a useDesigner hook for accessing the Designer Context and managing the state of the page elements, including their addition, selection, and removal. The design canvas changes based on the display sizes set, and elements can be dragged onto it, setSelected (for editing and deletion) or rearranged.

useDnDMonitor - useDndMonitor defines the complex interaction of elements during drag operations. It defines where a draggable element is dropped inside the page's structure i.e. whether it is over or under another PageElement. It also ensures only valid operations are performed, such as preventing the addition of multiple navigation bars or footers. A Designer Element Wrapper function shows individual PageElements. It provides draggable functionality and hover effects when rearranging the page structure.

Component Navbar - The Components Navbar on the left hand side of the website editor renders all of the different components available to the user. This is done through the component <SidebarButtonElement pageElement={PageElements.NavbarOne} />. The pageElement. designerButtonElement is extracted in this component. The title and associated image set in that PageElement's constructor are extracted and shown. They are applied using the useDraggable property to allow them to be dropped onto the Designer canvas.

Properties Navbar - When a PageElement is clicked in the Designer, it is setSelected in the Designer Context, which conditionally renders the Navbar. It extracts PropertiesForm = PageElements [selectedElement?.type].propertiesComponent; and then renders <PropertiesForm elementInstance = {selectedElement} sitePages = {sitePages} />. inside of the Navbar. This renders propertiesComponent: (props) => <Update {...props} />, the properties form of that component. When an update occurs, values are changed with updateElement() from Designer Context.

3.12 Feature Ten - Website Rendering

The Domain route is used for displaying the user's site on the internet. When a request comes in from a URL other than localhost or NEXT_PUBLIC_ROOT_DOMAIN, it is routed to the [domain] route. This is possible through the use of SaaSCustomDomains.

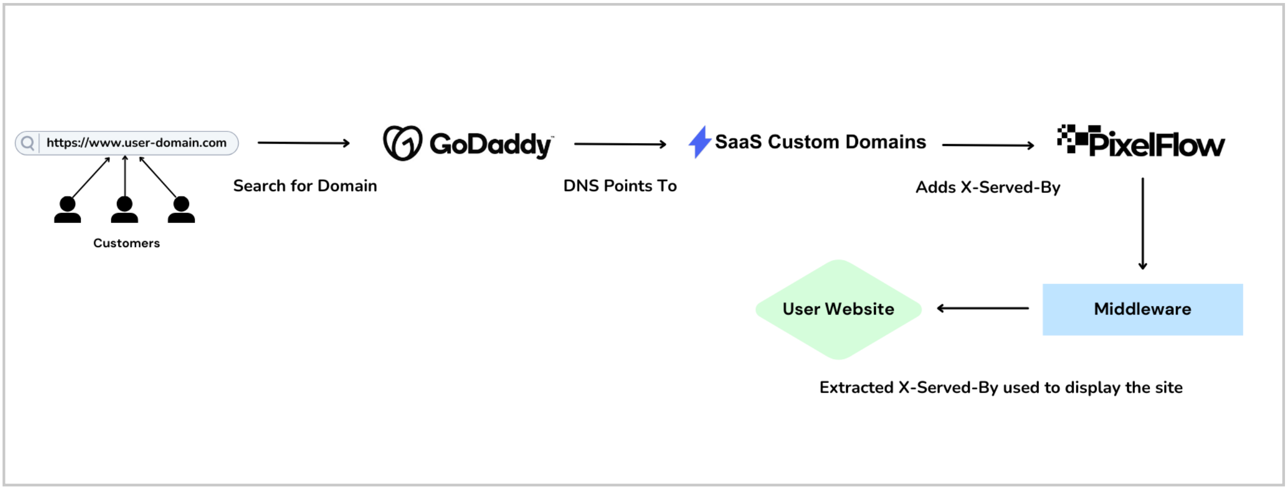
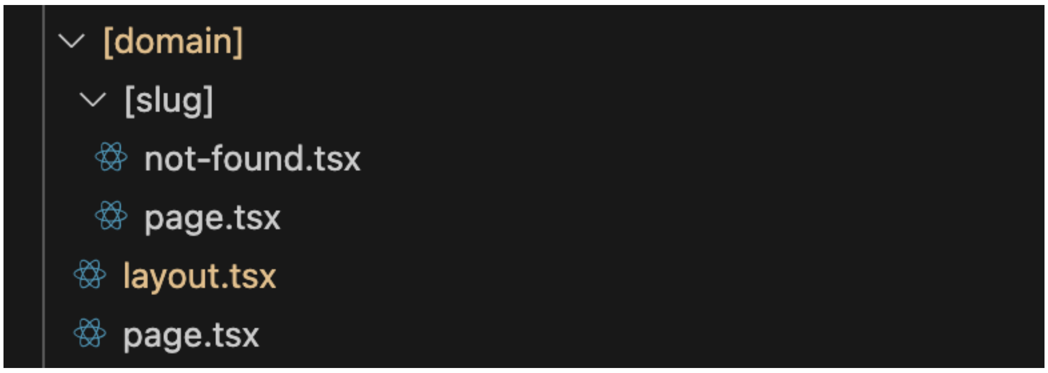


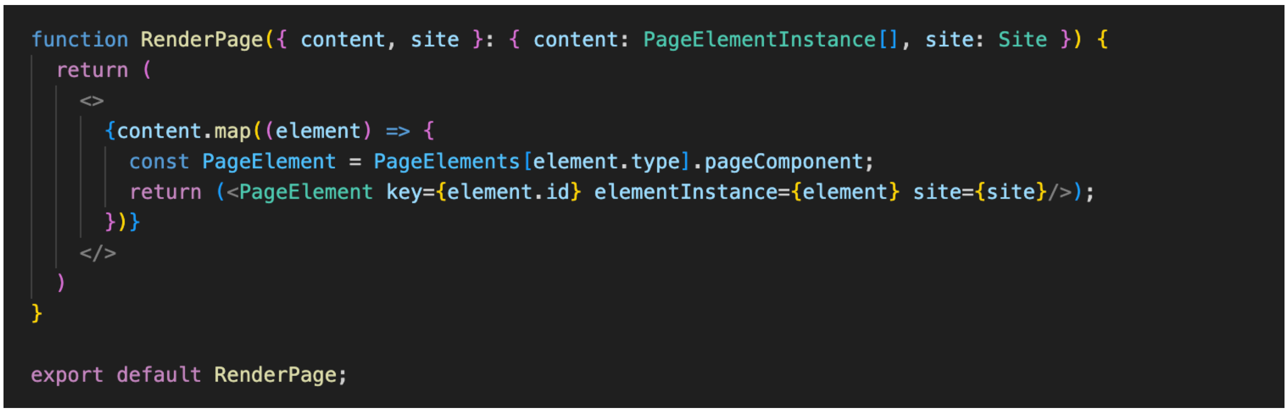
Figure 26: Domain Routing Workflow

When the custom domain is requested by a client, standard internet procedure occurs to resolve the domain to a DNS provider. Inside the domain registrar, GoDaddy in this case, the user would have been instructed to add custom A records which point the domain to SaaSCustomDomains. When this request is then sent to SaaSCustomDomains, it checks for the domain, verifies its connection and is able to route the request to PixelFlow. The Middleware extracts customDomain = requestHeaders.get('X-Served-For') and returns NextResponse.rewrite(new URL(`/${customDomain}${path}`, req.url));



```
[domain]
  [slug]
    not-found.tsx
    page.tsx
  layout.tsx
  page.tsx
```

Figure 27: Domain Route

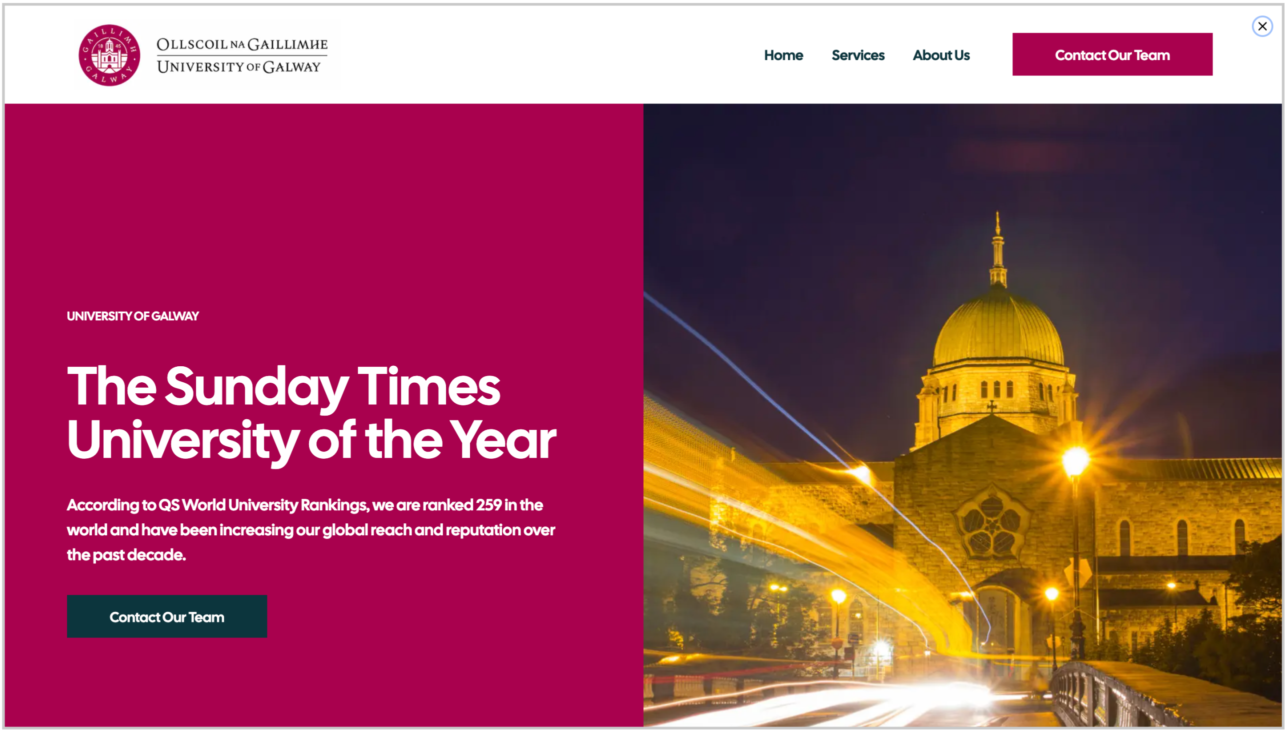


```
function RenderPage({ content, site }: { content: PageElementInstance[], site: Site }) {
  return (
    <>
      {content.map((element) => {
        const PageElement = PageElements[element.type].pageComponent;
        return (<PageElement key={element.id} elementInstance={element} site={site}/>);
      })}
    </>
  )
}

export default RenderPage;
```

Figure 28: Render Page Component

The RenderPage component is responsible for displaying the correct content associated with the website to the screen. It works by taking in the content, which is typed as a PageElementInstance[]. Each individual PageElementInstance = { id: string; type: ElementsType; extraAttributes?: Record<string, any>; } - which we can loop through for each component that has been added to the site. We are getting the "pageComponent" property, which maps to the specific Render.tsx file associated with that element. This is defined pageComponent: (props) => <Render {...props} />.



UNIVERSITY OF GALWAY

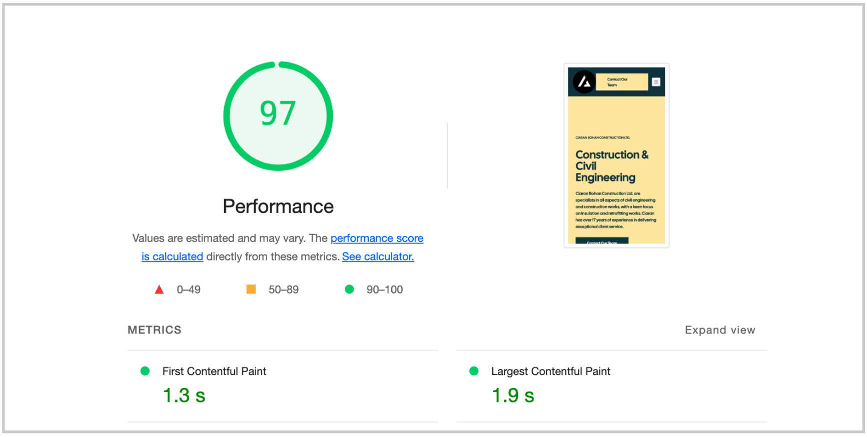
The Sunday Times University of the Year

According to QS World University Rankings, we are ranked 259 in the world and have been increasing our global reach and reputation over the past decade.

Figure 29: Hero Section Rendering

Section Four - Code Quality

4.1 Sample Website



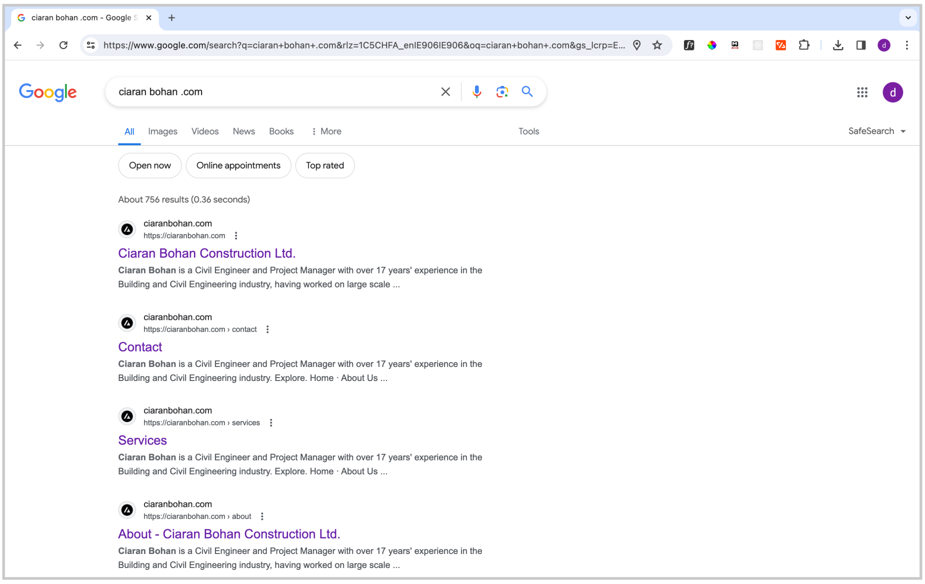
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

First Contentful Paint
1.3 s

Largest Contentful Paint
1.9 s

Figure 30: Website Lighthouse Score - SEO, Best Practices, Accessibility and Load Speed

Lighthouse is an open-source tool provided by Google for measuring the performance of webpages. It is primarily used to assist developers in enhancing performance, accessibility, SEO, and adherence to best practices. It performs an audit on each of these areas of the site and provides feedback on how to improve its performance. Lighthouse audits can to run through Chrome DevTools. A score of 97 represents a highly optimised webpage. Google [21] suggests "To provide a good user experience, sites should strive to have a good score (90-100). A "perfect" score of 100 is extremely challenging to achieve and not expected."



https://www.google.com/search?q=ciaran+bohan+.com&rlz=1C5CHFA_enIE906IE906&oq=ciaran+bohan+.com&gs_lcrp=E...

ciaran bohan .com

About 756 results (0.36 seconds)

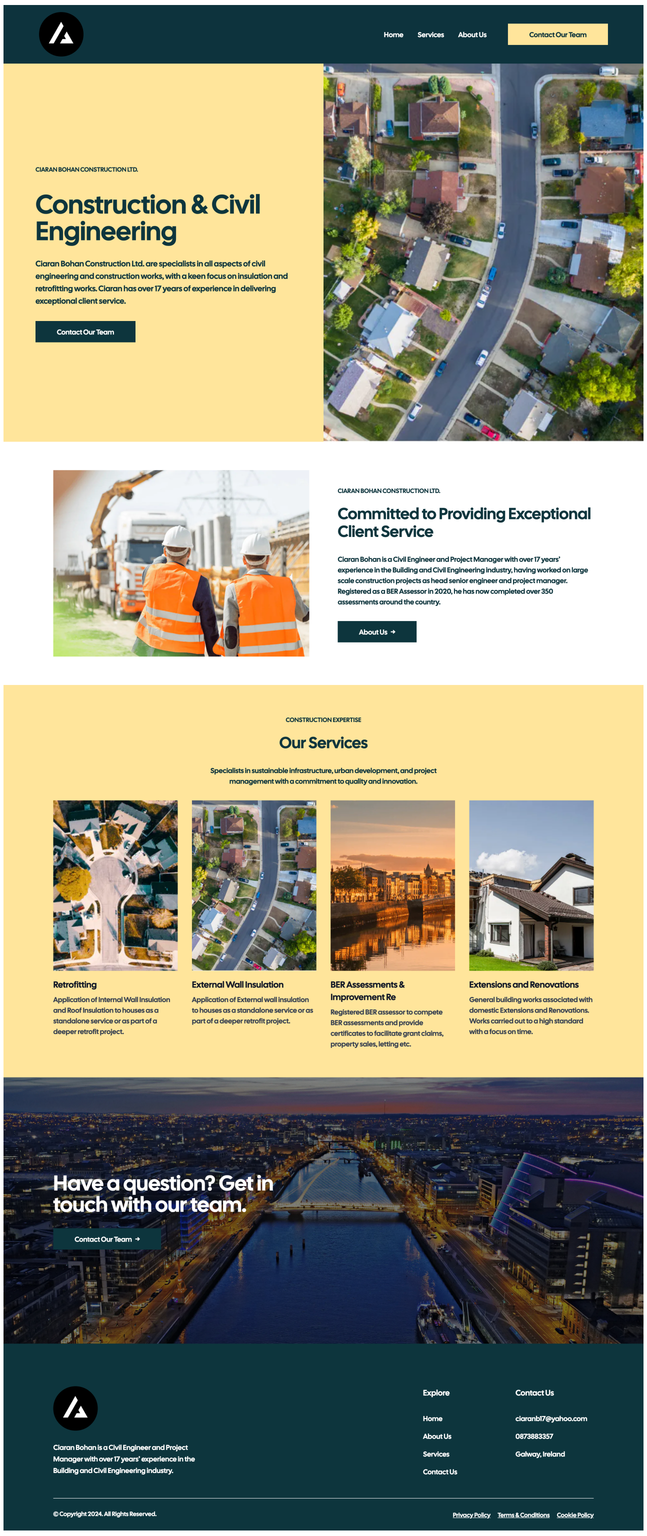
ciaranbohan.com
https://ciaranbohan.com
Ciaran Bohan Construction Ltd.
Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry, having worked on large scale ...

ciaranbohan.com
https://ciaranbohan.com › contact
Contact
Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry. Explore. Home · About Us ...

ciaranbohan.com
https://ciaranbohan.com › services
Services
Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry. Explore. Home · About Us ...

ciaranbohan.com
https://ciaranbohan.com › about
About - Ciaran Bohan Construction Ltd.
Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry, having worked on large scale ...

Figure 31: Sample Website Indexing on Google



Home Services About Us Contact Our Team

CIARAN BOHAN CONSTRUCTION LTD.

Construction & Civil Engineering

Ciaran Bohan Construction Ltd. are specialists in all aspects of civil engineering and construction works, with a keen focus on insulation and retrofitting works. Ciaran has over 17 years of experience in delivering exceptional client service.

Contact Our Team

CIARAN BOHAN CONSTRUCTION LTD.

Committed to Providing Exceptional Client Service

Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry, having worked on large scale construction projects as head senior engineer and project manager. Registered as a BER Assessor in 2020, he has now completed over 350 assessments around the country.

About Us →

CONSTRUCTION EXPERTISE

Our Services

Specialists in sustainable infrastructure, urban development, and project management with a commitment to quality and innovation.

Retrofitting

Application of Internal Wall Insulation and Roof Insulation to houses as a standalone service or as part of a deeper retrofit project.

External Wall Insulation

Application of External wall insulation to houses as a standalone service or as part of a deeper retrofit project.

BER Assessments & Improvement Re

Registered BER assessor to compete BER assessments and provide certificates to facilitate grant claims, property sales, letting etc.

Extensions and Renovations

General building works associated with domestic Extensions and Renovations. Works carried out to a high standard with a focus on time.

Have a question? Get in touch with our team.

Contact Our Team →

Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry.

Explore

Home

About Us

Services

Contact Us

Contact Us

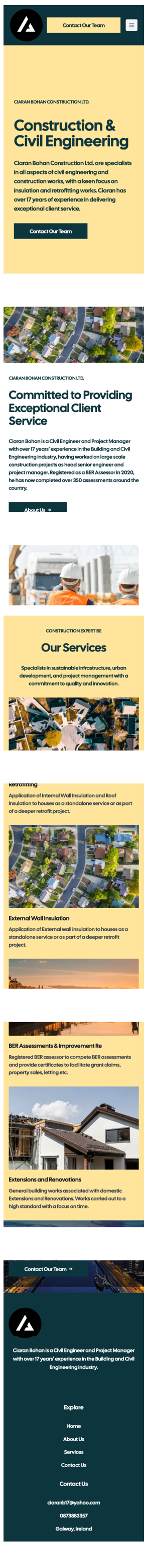
ciaranb17@yahoo.com

0873883357

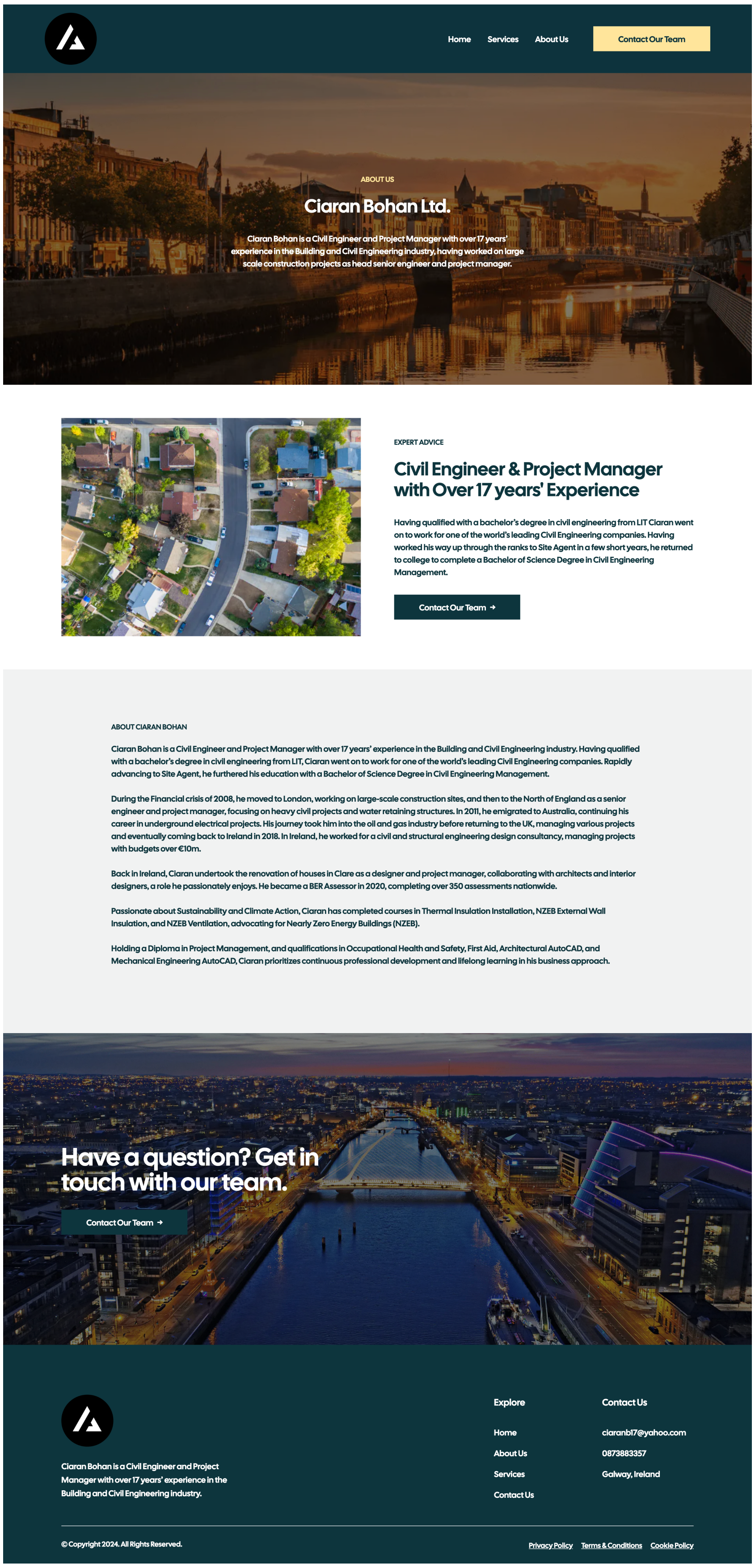
Galway, Ireland

© Copyright 2024. All Rights Reserved.

Privacy Policy Terms & Conditions Cookie Policy



This sample landing page demonstrates the quality of the webpage that can achieved using the PixelFlow platform. It is fully responsive across all screen sizes and would serve as an excellent website for a local business, such as a Construction Contractor.



ABOUT US

Ciaran Bohan Ltd.

Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry, having worked on large scale construction projects as head senior engineer and project manager.

EXPERT ADVICE

Civil Engineer & Project Manager with Over 17 years' Experience

Having qualified with a bachelor's degree in civil engineering from LIT Ciaran went on to work for one of the world's leading Civil Engineering companies. Having worked his way up through the ranks to Site Agent in a few short years, he returned to college to complete a Bachelor of Science Degree in Civil Engineering Management.

ABOUT CIARAN BOHAN

Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry. Having qualified with a bachelor's degree in civil engineering from LIT, Ciaran went on to work for one of the world's leading Civil Engineering companies. Rapidly advancing to Site Agent, he furthered his education with a Bachelor of Science Degree in Civil Engineering Management.

During the Financial crisis of 2008, he moved to London, working on large-scale construction sites, and then to the North of England as a senior engineer and project manager, focusing on heavy civil projects and water retaining structures. In 2011, he emigrated to Australia, continuing his career in underground electrical projects. His journey took him into the oil and gas industry before returning to the UK, managing various projects and eventually coming back to Ireland in 2018. In Ireland, he worked for a civil and structural engineering design consultancy, managing projects with budgets over €10m.

Back in Ireland, Ciaran undertook the renovation of houses in Clare as a designer and project manager, collaborating with architects and interior designers, a role he passionately enjoys. He became a BER Assessor in 2020, completing over 350 assessments nationwide.

Passionate about Sustainability and Climate Action, Ciaran has completed courses in Thermal Insulation Installation, NZEB External Wall Insulation, and NZEB Ventilation, advocating for Nearly Zero Energy Buildings (NZEB).

Holding a Diploma in Project Management, and qualifications in Occupational Health and Safety, First Aid, Architectural AutoCAD, and Mechanical Engineering AutoCAD, Ciaran prioritizes continuous professional development and lifelong learning in his business approach.

Have a question? Get in touch with our team.

Ciaran Bohan is a Civil Engineer and Project Manager with over 17 years' experience in the Building and Civil Engineering industry.

Explore	Contact Us
Home	ciaranb17@yahoo.com
About Us	0873883357
Services	Galway, Ireland
Contact Us	

© Copyright 2024. All Rights Reserved.

Figure 32: Sample About Page

4.2 Application Feedback

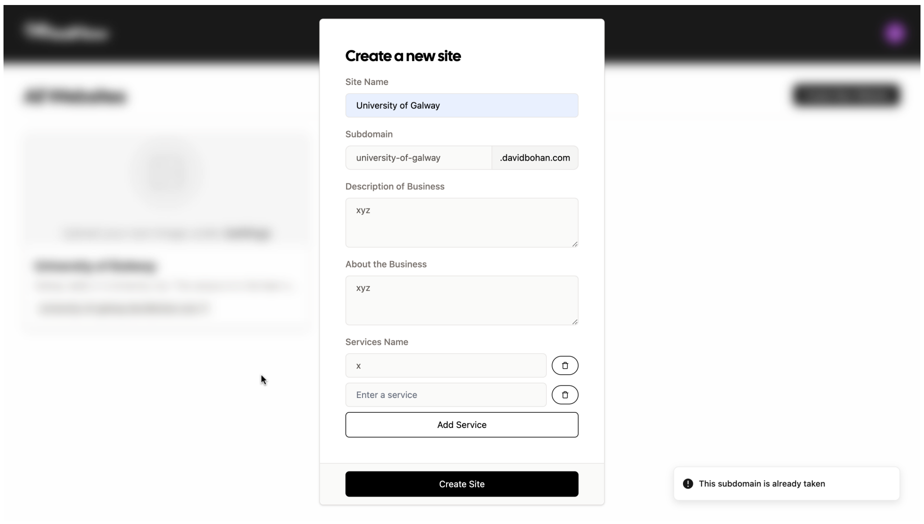


Figure 33: Toast Used to Show Errors to the User

Toast notifications are UI states which briefly appear on a website or app to inform the user about an event. This could be a success message or an error alert. Error handling was implemented across the application to provide feedback to the user. In the above example, a toast error is thrown when a subdomain is already taken by a different user.

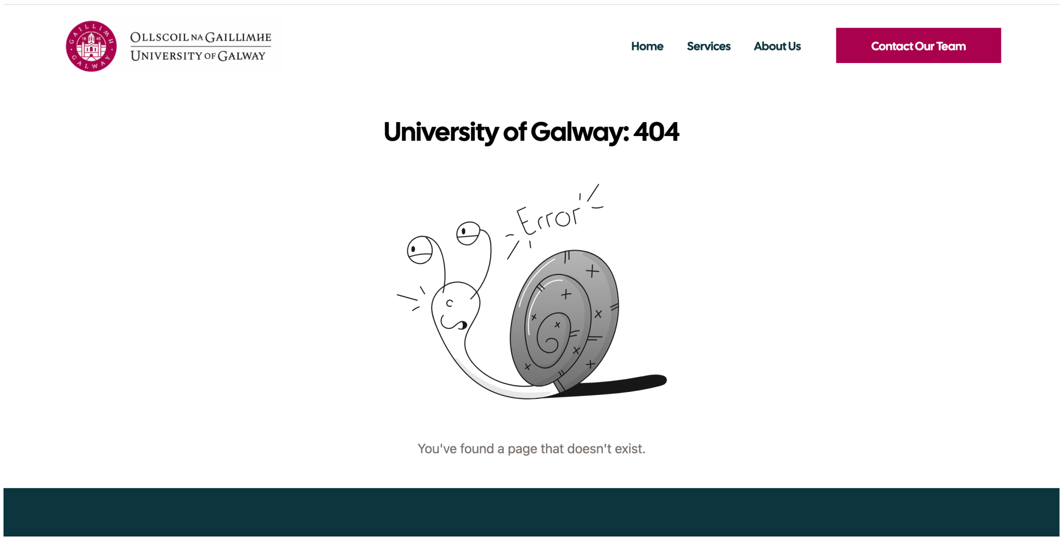


Figure 34: 404 Pages Shown Where Page Does not Exist

404 Pages and NotFound pages are displayed when a user navigates to a URL which they either don't have permission to access, or the page does not exist. This avoids a generic Vercel error page from showing and causing confusion to the user.

4.3 TypeScript and ESLint

JavaScript is an interpreted language, which means that it needs to have an interpreter in the environment to read the actual source code and execute it. This differs from a compiled language like Java or C, which statically analyse all of your code in advance and then compiles it down to a binary which can be run. Compiled languages require type definitions. JavaScript is a dynamically typed language, meaning we don't need to use any explicit type definitions to use vanilla JavaScript code. The type is associated with a runtime value and not the actual variables or functions in the code.

TypeScript is a strongly typed superset of JavaScript, which enhances code quality primarily through static type checking. This allows us to catch errors in our code at compile time rather than at runtime, which significantly improves our chance of finding bugs related to type mismatches. It also improves code readability, and maintains compatibility across different environments.

ESLint is a static code analysis tool for identifying problematic patterns found in JavaScript code. It enforces coding standards and styles, leading to more consistent and error-free code. It integrates seamlessly into the development workflow, offering immediate feedback to developers and ensuring that code conforms to the agreed-upon standards. Linting is run on the application before it is committed to the codebase.

TypeScript can be used together with ESLint to offer a comprehensive approach to maintaining high code quality. This combination of static type checking, adherence to coding standards, and the enforcement of stylistic guidelines improves code quality.

Section Five - Conclusion

5.1 Challenges

During the course of completing this project, I faced many challenges which tested not only my programming ability, but also my perseverance. The technical complexity of building such a large application with multiple functionalities was a huge challenge. By dedicating a significant amount of time in the planning and research phase, and by carefully outlining the functionalities and their interactions, I approached the project with a solid plan. This proved to be hugely beneficial as I moved into the implementation phase.

This is by far the largest project I have ever worked on, and the most difficult. Prior to starting this project, I had not built any sizeable applications using Next Js. I had previous experience with React but adapting to Next Js was difficult to start. Each component seemed to require learning a new skill, DnD kit, domain integrations, learning technical SEO and metadata etc. A considerable portion of time had to be dedicated to learning and experimentation with the various technologies. The technologies used had excellent documentation, which was incredibly helpful in learning how to apply them effectively.

Time management also was a big challenge. Working to break the project down into sprints and using an agile approach helped me to stay consistent and ensured the project was delivered to a high standard. Prioritising the development of core functionalities first helped to create a strong foundation to which I could add smaller features. The use of third-party integrations also allowed me to focus on the core features, for example I originally implemented authentication using NextAuth, and switching to Clerk saved a huge amount of time.

I completed the majority of this project during the Christmas break from college. This allowed me to balance the demands of this project with other academic responsibilities and personal commitments. Having not started so early, I would not have been able to complete this entire application.

5.2 Lessons Learned

Building this project has been really rewarding and a great learning experience. One of the clear takeaways was the importance of clear planning, and scope definition at the beginning of projects. Setting goals, prioritising tasks and setting timeframes for complete was critical for managing a project of this size.

Furthermore, I gained an appreciation for the need for user feedback when developing software. Things that may be obvious for the person developing the project with perfect knowledge of how the entire back-end works, may not be obvious to someone knew to the software.

Documentation was important for learning and understanding complex code. I spent a considerable amount of time struggling to implement features and fixing bugs. This taught me resilience and to keep trying when things aren't working. This project was much larger and much more complex than I thought I was capable of achieving.

5.3 Future Work

I plan to further develop this application and bring it to market. Having already spoke to some local businesses, the response so far seems positive and I hope to turn this into a real-world application. From preliminary testing, the SEO capabilities and performance of the websites are already near that of other modern website builders. Over the coming months, I want to improve the user experience, particularly in the automatic generation of the site. I want to add a feature for truly personalised websites to be generated, and regenerated if the user decides. This will include adding more page element options and expanding the design possibilities.

Incorporating analytics features would also provide users with insights into their website's performance. This would also come with the need to add a cookie banner to the websites. Adding e-commerce functionalities could open up new opportunities for businesses looking to establish or expand their online sales channels. Finally, the ability for a web design agency to sign up and manage their client websites from within the application. The goal for future work is to enhance the platform's scalability, functionality, and user-friendliness, making it a powerful tool.

5.4 Final Conclusion

The completion of PixelFlow marks a significant milestone in my academic and professional journey. It has been challenging at times but overall a rewarding experience. I am glad that so much effort went into this project and I am proud of what was achieved. This would not have been possible without the help of my supervisor and feedback from friends/family.

Section Six - References

[1] Zippa. (n.d.). Front-end developer jobs salary. Available at: https://www.zippia.com/developer-jobs/salary/

[2] Forbes. (n.d.). How much does a website cost? Available at: https://www.forbes.com/advisor/business/seo-cost/

[3] Wix. (n.d.). Free Website Builder | Create a Free Website. Available at: https://www.wix.com/

[4] Boston Consulting Group. (April 2020). BCG: The real cost of poor website quality. Available at: https://www.g2.com/products/g2/reviews

[5] Salesforce. (2017). Retail campaign 2017 eBook. Available at: https://www.salesforce.com/services/professional-services/resources/

[6] Webflow. (n.d.). Create a custom website | Visual website builder. Available at: https://www.webflow.com/

[7] Squarespace. (n.d.). Website Builder — Create a Website in Minutes. Available at: https://www.squarespace.com/

[8] Wix. (n.d.). How long does it take to build a website? Available at: https://www.wix.com/blog/how-long-does-it-take-to-build-a-website

[9] WordPress. (n.d.). Create a Free Website or Blog. Available at: https://wordpress.com/

[10] Weebly. (n.d.). Free Website Builder: Build a Free Website or Online Store. Available at: https://www.weebly.com/

[11] GitHub. (n.d.). Where the world builds software. Available at: https://github.com/

[12] Notion. (n.d.). Write, plan, share. With AI at your side. Available at: https://www.notion.so/

[13] Vercel. (n.d.). Next.js by Vercel - The React Framework. Available at: https://nextjs.org/

[14] Vercel. (n.d.). Develop. Preview. Ship. Available at: https://vercel.com/

[15] Prisma. (n.d.). Prisma - Next-generation Node.js and TypeScript ORM for Databases. Available at: https://www.prisma.io/

[16] Tailwind Labs. (n.d.). Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. Available at: https://tailwindcss.com/

[17] Clerk. (n.d.). Clerk - User Management & Authentication for Modern Applications. Available at: https://clerk.dev/

[18] SaaSCustomDomains. (n.d.). Custom domains and white-labelling for SaaS. Available at: https://saascustomdomains.com/

[19] Resend. (n.d.). Email for developers. Available at: https://resend.com/

[20] OpenAI. (n.d.). OpenAI. Available at: https://openai.com/

[21] Unsplash. (n.d.). Beautiful Free Images & Pictures. Available at: https://unsplash.com/

[22] Google Developers. (n.d.). Lighthouse performance scoring. Available at: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring