Sagar Regmi

BUILDING A RESPONSIVE WEBSITE

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering, Information Technology

October 2022

ABSTRACT

Centria University of Applied Sciences	Date	Author
	October 2022	Sagar Regmi

Degree programme	
Bachelor of Engineering, Information Technology	
Name of thesis	
BUILDING A RESPONSIVE WEBSITE	
Centria supervisor	**Pages**
Jari Isohanni	39 + 3
Instructor representing commissioning institution or company	
Jari Isohanni	

The purpose of the thesis was to research and study the concept of creating a responsive web design and implement the study to build a website design that is optimised for adaptive web design.

The theoretical part of the thesis work explains the concepts of creating responsive web pages and elements to create an adaptive design, including flexible text, flexible grid, and adaptive media, as well as a theory on media queries. For practical purposes, a website as a prototype was built, tested on various screen resolution devices, and evaluated in terms of adaptive design.

The development of the project involved various steps. Initially, the requirements of the project were analysed. Based on the requirements, the project was designed and developed using Virtual Studio Code, a software development tool. As an outcome, a website that can adapt content according to the screen size was created. HTML5, CSS3, SASS, VS Code, NPM, CSS grid, and CSS flexbox were the technologies used in the project.

Key words

Responsive web design, CSS3, HTML5, SASS, Media Queries, Flexible layouts, Grid, Flexbox, Adaptive design

CONCEPT DEFINITIONS

HTML

(Hyper Text Markup Language) is the basic building block of any website which includes the structure of the web content.

CSS

(Cascading Style Sheets) is a style sheet language used to present and design the web content.

RWD

(Responsive Web Design) is a web design that will automatically adjust its display according to the screen size.

SASS

(Syntactically Awesome Stylesheet) is a CSS pre-processor.

SCSS

(Sassy Cascading Stylesheet) is a form SASS with different style of syntax.

NPM

(Node Package Manager) is the package manager for the Node JavaScript platform.

W3C

(World Wide Web Consortium) is an international community to develop web standards.

VS Code

(Virtual Studio Code) is a software development tool.

ABSTRACT

CONCEPT DEFINITIONS

CONTENTS

1 INTRODUCTION 1

2 THEORITICAL BACKGROUND 3

2.1 Web Design Languages 3

2.1.1 HTML 3

2.1.2 CSS 4

2.2 Responsive Web Design 4

2.3 Elements of Responsive Web Design 7

2.3.1 A flexible grid-based layout 7

2.3.2 Adoptable pictures and media 13

2.3.3 Media Queries 17

3 PROJECT ANALYSIS AND DEVELOPMENT 21

3.1 Requirement Analysis 21

3.2 Used Web Technologies 21

3.2.1 Virtual Studio Code 22

3.2.2 HTML5 24

3.2.3 CSS3 and SASS 25

3.2.4 Flexbox and CSS Gird 27

3.2.5 NPM 29

3.3 Development 30

3.3.1 Website without RWD Elements 30

3.3.2 Implementing RWD Elements 32

4 RESULTS 36

5 CONCLUSION 37

REFERENCES 38

APPENDICES

FIGURES

Figure 1. CSS syntax (W3Schools, CSS Syntax and Selectors 2019). 4
Figure 2. Canvas (Marcotte 2011). 5
Figure 3. Percentage of Global Mobile Traffic (BroadbandSearch 2019). 6
Figure 4. Variety of display screens (Hussain 2017). 7
Figure 5. Values set in pixel. 8
Figure 6. Formula to calculate relative type sizes (Marcotte 2011). 8
Figure 7. Implementing the relative type values. 9
Figure 8. Example website for understanding flexible grid-based layout. 9
Figure 9. Evaluation of the sizes in pixel of the example website. 10
Figure 10. Problem of a fixed width layout. 11
Figure 11. Changing pixels to relative type sizes. 12
Figure 12. Flexible grid-based layout with relative values. 13
Figure 13. Image and video overflow its container. 14
Figure 14. A constraint that prevents images from exceeding the width of their container (Marcotte 2011). 14
Figure 15. Media is nicely adjusted with the container. 15
Figure 16. Picture element with srcset and media. 16
Figure 17. Condition set on <picture> element in result. 17
Figure 18. Adding media types (Marcotte 2011). 18
Figure 19. Media type handheld (Marcotte 2011). 18
Figure 20. Media queries in practice. 19
Figure 21. Media Queries worked! 20
Figure 22. Survey by Stack Overflow in 2021 (StackOverflow 2021). 23
Figure 23. Viewport meta tag. 25
Figure 24. Starting HTML file. 25
Figure 25.CSS flexbox used in the project. 28
Figure 26. CSS grid used in the project. 29
Figure 27. Installation of a SASS compiler using NPM. 29
Figure 28. Running the SCSS compiler. 30
Figure 29. Website design without RWD elements. 31
Figure 30. Decreasing the viewport width of the website. 31
Figure 31. Relative value is set on font-size of the html element. 32

Figure 32. Everything is set on relative type value with the font-size of html. 33
Figure 33. “srcset” attribute for responsive images. 34
Figure 34. Implementing breakpoints with the help of @content, @if and @mixin SASS directives. .34
Figure 35. Adding the created @mixin for media queries with the help of @include for html font-size. 35

TABLES

Table 1. Requirements of the project website. 21
Table 2. Web technologies used for the project website. 22
Table 3. Popular features of virtual studio code. 24
Table 4. Popular features of CSS3. 26
Table 5. Features of SASS. 27
Table 6. The result of comparing and analyzing the current design with the requirement of the project website. 32
Table 7. The result of comparing and analyzing the updated design with the requirement of the project website. 36

1 INTRODUCTION

With the advancement of technology, everything is evolving. E-commerce websites can be used to buy and sell goods or services, websites can be used to gather and share information, and websites are considered one of the most impactful marketing tools. Research has shown a user-friendly website is very crucial for customers to get engaged with the company (Falvain , Gurrea and Orús 2009). So, today we can find many companies advancing their websites. In the past, websites were built with a fixed width with the expectation that all users would get a consistent experience, but today we use smartphones, big-screen digital TVs, tablets, netbooks, and many other electronic items with a variety of screen sizes. So, with the fixed width layout in different sizes, users find it difficult to operate the web content, and the user experience is poor. Therefore, responsive web design was built, which allows the website to work across multiple devices and screens without the need for backend solution (Frain 2012).

In the twenty-first century, the websites need to adapt web browser to keep up with the latest technological advancements. Therefore, they must be adaptable and accessible from any device. We must update our websites to be compatible with each new gadget that hits the market. The days of using just computers to access all the internet's content, pay their bills, and make presentations for meetings are long gone. Today, though, users are concentrating on using tablets and mobile devices. Users no longer rely on computers and laptops since smartphones and tablets are faster and more competent than they were in the past. For instance, users can read the news, pay bills on time with these gadgets. The audience's or users' expectations of how well websites should work are what designers need to understand to design responsive websites that meet their demands (Sekuloski 2021).
Building a website with a fixed width was reasonable less than a decade ago. All end users were supposed to have a consistent experience, according to expectations. Users with big resolution monitors just had plenty of room on each side of this set width, which was generally 960 pixels wide, or roughly (Frain 2012). However, as of January 2022, mobile has taken the lead, accounting for a little over 55 percent of the market, with desktop devices accounting for 42 percent. The remaining 3% may be attributed to tablets, which, while not commonly utilized, are nonetheless present in the online ecosystem (BroadbandSearch 2019).

The undeniable reality is that more and more people are utilizing smaller-screen devices to access the internet, yet 27- and 30-inch displays (along with different tablet and console devices) are also becoming

popular. The contrast between the smallest screens for online browsing and the largest screens is currently greater than ever. Fortunately, a solution exists for this constantly evolving browser and device environment. An HTML and CSS responsive web design enables a website to "just work" across various devices and displays. It enables a website's design and functionalities to adapt to its surroundings (screen size, input method, and device/browser capabilities). Before the advent of adaptable web design, it was usual for companies to maintain a distinct mobile website with a separate URL. Before sending the browser to the host server, it needs to identify the user agent on the server and the necessary URL on a desktop or mobile device. A responsive website also has the advantage of being developed without the use of server-based or backend technologies (Frain 2012).

The objective of the thesis was divided into two primary goals, first to understand the concept of building a webpage that can adjust its content according to the screen viewport and second to implement the concept by building a simple prototype. To achieve this two primary objectives of the thesis. The thesis was divided into 4 main chapters, starting with the introduction. The first part, "Introduction" section, shows the importance of responsive web design in modern times. Also, this chapter goes through the drawbacks of design in the past few decades. The second part, "Theoretical background" section, covers all the important definitions and concepts required for responsive web design, starting from basic web language to elements of responsive web design. The third part, "Project analysis and development" section, covers the process of building a website project starting from project requirements to setup to the final design. The final chapter, "Result," shows the outcome of the thesis and the project and compares the outcome with the requirements. Finally, the thesis ends with the conclusion.

2 THEORITICAL BACKGROUND

This section covers the first objective of the thesis to understand the concept of responsive web design. It is crucial to understand the concept and definitions of the technologies or the components that are important for web design. In addition, this section covers the important theoretical background of the elements of the responsive web design and other technologies.

2.1 Web Design Languages

Web designs are identified as a key factor for the success of the websites. Website influence users and online consumers perceptions and behaviours. A website design that is simple and easy to navigate provides clear, timely and accurate information (Falvain , Gurrea and Orús 2009). HTML, CSS, and JavaScript are the three main programming languages that are used widely to build a webpage. Each of these languages have their own uses and are equally important. HTML is essential to give content to the webpage whereas CSS features are used to present the content and JavaScript is used to give dynamic content to the webpage or to add features to the content. Although, JavaScript is very essential to build a website but only with the help of HTML and CSS features, a responsive web design can be built.

2.1.1 HTML

HTML stands for Hypertext Markup Language. It is a language that allows to markup content in a way that makes it easier to understand for technology and therefore for humans. HTML is essential, a website without CSS code or JavaScript functionality can be created but not without HTML (Sekuloski 2021). HTML is a standard markup language for creating web pages, it consists of a series of elements that describe the structure of a web page. The content of the webpage is written inside tags or elements. Most of the elements in HTML have an opening and closing tag, but there are a few instances where the closing tag is not needed. These elements are referred to as “self-closing” tags. Web browsers read HTML documents and display them accordingly. A browser does not display the HTML tags but uses them to determine how to display the document. Since the early days of the World Wide Web, there have been many versions of HTML (W3Schools, Introduction to HTML 2022).

2.1.2 CSS

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on the screen, on paper, or in other media. It defines styles for the web pages, including the design, layout, and variations in display for different devices and screen sizes. HTML was created to describe the content of a web page and was never intended to contain tags for formatting a web page. Developing large websites, where fonts and color information are added to each page, becomes a time-consuming and expensive process (W3Schools, CSS Introduction 2019). To solve this problem, the World Wide Web Consortium (W3C) created CSS.

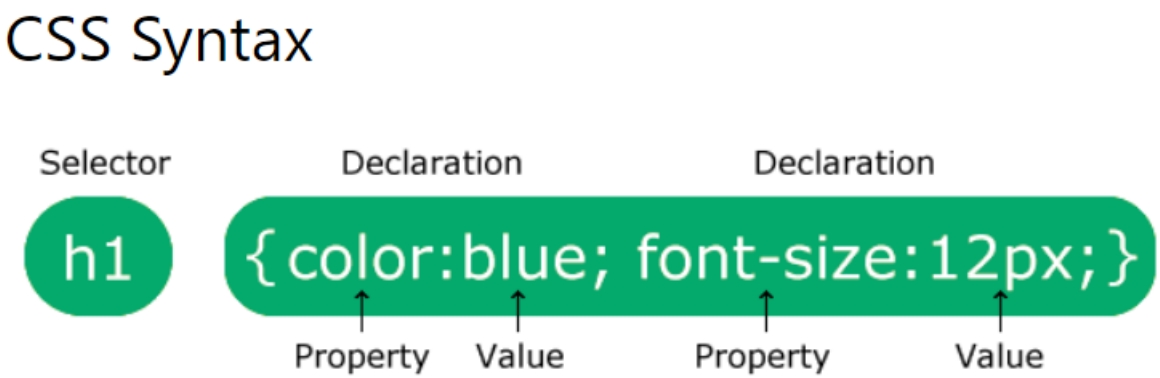


Figure 1. CSS syntax (W3Schools, CSS Syntax and Selectors 2019).

Figure 1 is a CSS syntax and in the selector, the HTML element which is intended to be styled is selected. The declaration block consists of one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon, and multiple CSS declarations are separated by semicolons, and declaration blocks are surrounded by curly braces (W3Schools, CSS Syntax and Selectors 2019).

2.2 Responsive Web Design

Responsive web design is about creating a web page that will automatically adjust its web content according to different screen sizes and viewports. Using responsive architecture, we can build a website that can adapt to the media that renders them. According to (Marcotte 2011), the web designing concept

was inherited from the canvas concept as shown in figure 2. Every artist starts with picking a canvas. A painter chooses a sheet of paper, a sculptor might select a block of stone. Whatever the medium, deciding on a canvas is a significant step in the creative process because, even before the first brush stroke, the canvas gives the piece a sense of scale, setting a boundary for the work yet to come (Marcotte 2011).

Figure 2. Canvas (Marcotte 2011).

Being influenced by this technique, Ethan Marcotte and other web designers tried to replicate the process by creating the canvas, a blank document with a certain width and height. Users' browser display and font settings influenced their designs as soon as they were uploaded online, so they had no control over how their designs would look on other people's screens. They begin by implementing constraints: they set type in pixels and develop fixed-width layouts that presume a minimum screen resolution. Implementing these constraints is a bit like selecting a canvas (Marcotte 2011). But the browser window was inconsistent, removing whatever constraints they place. This fixed width layout was not practical because even if a browser drops slightly below the minimum width, a site visitor might find it hard to read the content of the web page as it is alternated by horizontal scrollbars and clipped content (Marcotte 2011).

The landscape of browsers is moving more and further away from the traditional desktop computer, while at the same time, the sizes of the devices themselves are expanding and contracting. According to data published at (BroadbandSearch 2019), it can be predicted that using the Internet on devices with

small screens will be the most common practice. Tablet computers have been highly popular from 2010, after Apple launched iPad, allowing users a means to browse the internet that is neither totally "mobile" nor fully "desktop." While current gaming consoles and television have made widescreens, tablet computers have made widescreens more accessible (Marcotte 2011).

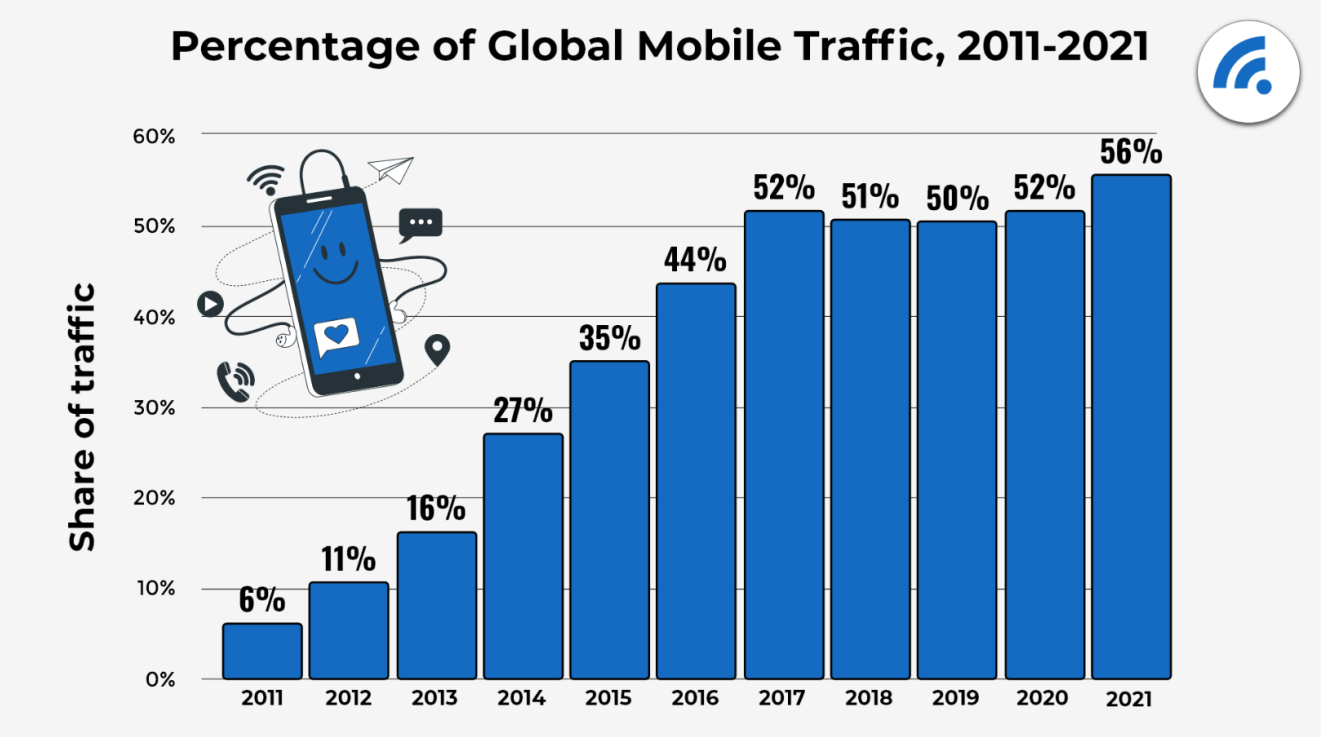


Figure 3. Percentage of Global Mobile Traffic (BroadbandSearch 2019).

So, web designers were designing for more devices, input types, and resolutions than ever before. The web is no longer limited to desktop computers, and it will not return. Unfortunately, early efforts by web designers to design outside the desktop seemed a lot like our old methods of doing things, such as implementing constraints on things that was not clear (Marcotte 2011).For instance, a site's owner assumed a 320-pixel page width when they quarantined the mobile experience to a different URL. When that link (http://www.bbc.co.uk/news/mobile/science-environment-13095307) is posted on social media, for example on Twitter or by email, the visitors will be confined to the small-screen-friendly view regardless of the device they use to read it. The reading experience on a desktop browser, on the other hand, was terrible (Marcotte 2011).

For good user experience, the web designer cannot keep on creating every new browser or device that appears. So, Ethan Marcotte coined the term Responsive Web Design. He consolidated three existing

techniques (flexible grid layout, flexible images, and media queries) into one unified approach and named it responsive web design (Frain 2012).

Figure 4. Variety of display screens (Hussain 2017).

The above figure 4 indicates how the layout of the same website can be modified to fit different devices. On the desktop, the viewport width is larger so can fit more content in a single row. However, when the screen size and orientation change, the content readjusts to adapt the shift. This gives a consistent and beautiful user experience across all form factors (Hussain 2017).

2.3 Elements of Responsive Web Design

Three essential components make up a responsive front-end layout: a flexible grid-based layout, adaptable pictures and media, and media queries. This method aids in developing a more adaptable, responsive approach to web design. By using this architecture, a design that can change to fit the limitations of the browser window or other rendering medium can be built (Marcotte 2011).

2.3.1 A flexible grid-based layout

In the early 90s, the layout structures were table-based, and all the sectioning was done with percentages. There were not any vast differences in browser viewports, so these layouts worked in a limited range of

viewports, so this layout was not future proof. A fluid layout is a type of webpage design in which the layout of the page resizes as the display size is changed. The web content component is designed in percentage widths, so it automatically adjusts to the user's screen (Frain 2012). Before understanding, a flexible grid layout it is important to understand the concept by creating a flexible font through an example in figure 5.

```
body{
    font-size:100%; /*16px Default  */
}
h1{
    font-size:30px;
}
h1 a {
    font-size:20px;
}
```

Figure 5. Values set in pixel.

The HTML body element has a font-size of 100% (In most browsers, it is 16px). If the font-size attribute is not declared in the content itself, then the material within the body inherits the font-size property. The element inside of the body are set on a fixed value. Whatever the display it is rendered, the element h1 and link would have a fixed value rather than a flexible value. A relative datatype "em" (em is a CSS unit relative to the font-size of the parent element) is used to solve this issue. To find the exact value in a relative datatype, Ethan Marcotte has devised a formula shown in figure 6.

```
target ÷ context = result
```

Figure 6. Formula to calculate relative type sizes (Marcotte 2011).

With the help of the formula in figure 6 all the font-sizes of the element inside of the body element are set on a relative datatype so the text on the webpage changes based on the font-size of the body element, allowing to make all font-sizes flexible. In figure 7, the formula to calculate relative type size is implemented.

```
/* Normal 16 px = 1 em;
 Ethan Marcotte formula;
 target/context=result
*/
body{
    font-size:100%; /*16px Default*/
}
h1{
    font-size:1.875em; /*30px;
                target/context=result
                30/16=1.875em*/
}
h1 a {
    font-size:0.67em; /*20px;
                target/context=result
                20/30=0.67em*/
}
```

Figure 7. Implementing the relative type values.

As with flexible font size, grid layout can be set proportionally. Every aspect of the grid can be set as proportions of their containing element, rather than unchanging, inflexible pixels (Marcotte 2011). For instance, in figure 8.

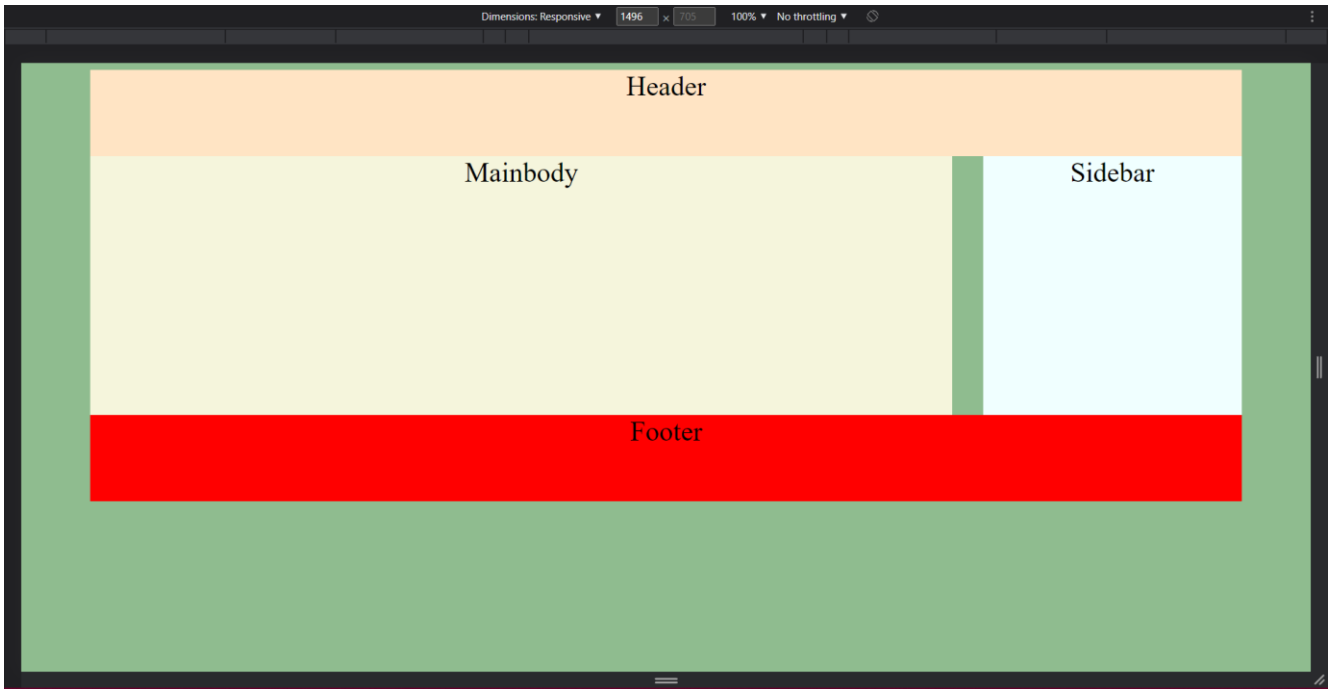


Figure 8. Example website for understanding flexible grid-based layout.

Evaluating the figure 8, the container, or the canvas, has a fixed width of 1335.6px. The main and sidebar are aligned sidewise using float left and right. The main body and sidebar have a fixed width of 1000px and 300px, respectively, with a margin between the content of 35. 6px. Altogether, the main body, margin, and sidebar give us a total width of 1335.6px.

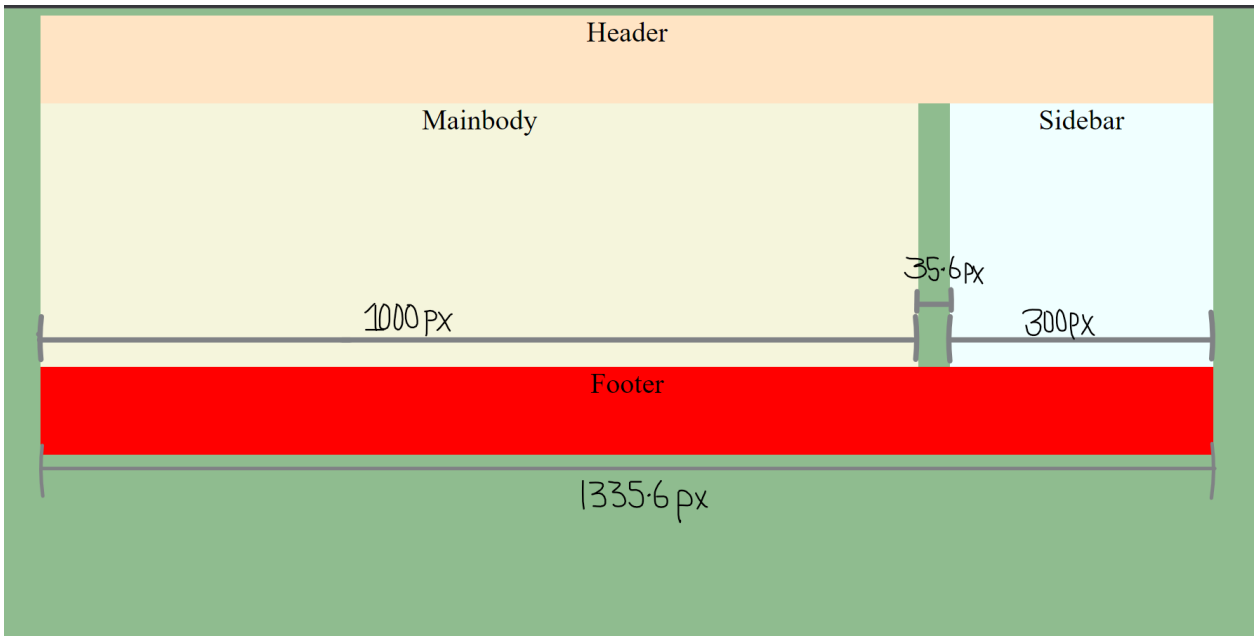


Figure 9. Evaluation of the sizes in pixel of the example website.

The Layout in figure 9 matched perfectly in terms of numbers, but the layout is inflexible. Our grid cannot change its content with the screen size, forcing a horizontal scrollbar upon the reader if the window drops even slightly below as shown in figure 10.

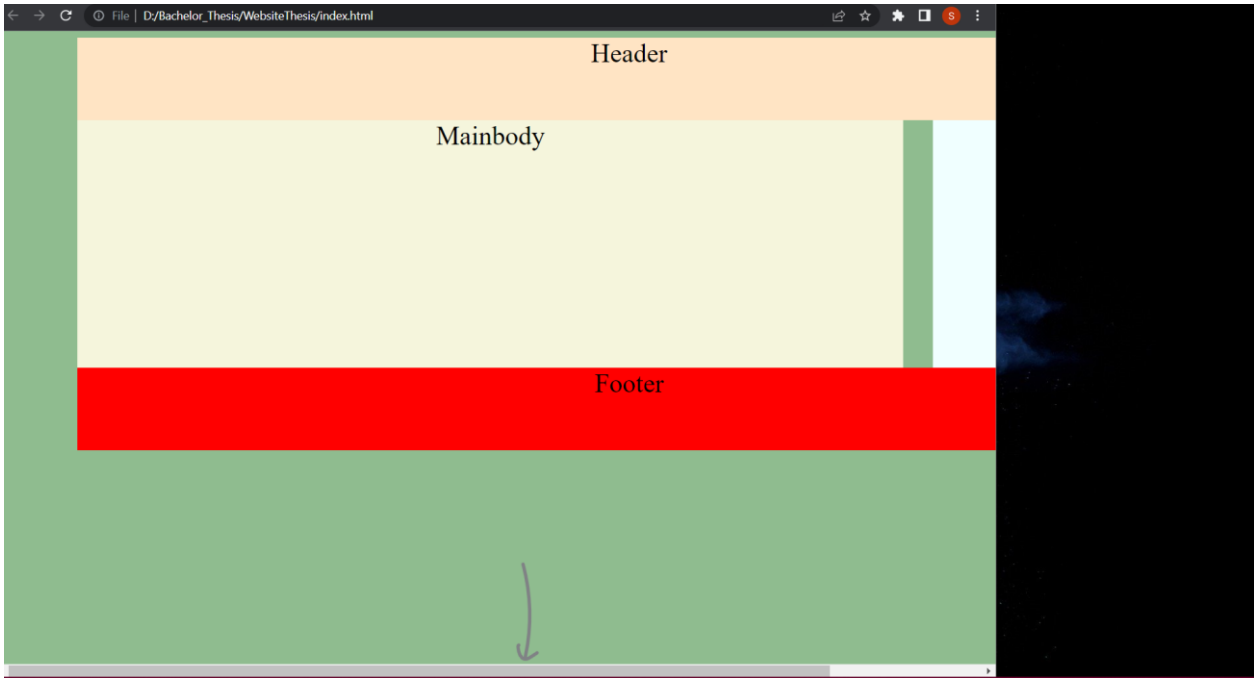


Figure 10. Problem of a fixed width layout.

Instead of a value set in pixels, those widths should be expressed in relative proportional terms. If the widths of different containers are linked in relative proportion, the grid can resize itself with the viewport and the relative type sizes can be determined with the formula shown in figure 6. In case of margin and padding, the context is different for each. When setting flexible margins on an element, your context is the width of the element's container, whereas when setting flexible padding on an element, your context is the width of the element itself, which makes sense because in the box model, padding is described in relation to the width of the box itself (Marcotte 2011). In the example website, padding is not used but margin should be in relative proportional terms and the relative type sizes can be determined with the formula shown in figure 6.

```css
/*
 Ethan Marcotte formula;
 target/context=result
*/
html{
    height:100vh; /* 705.6px is the defalut width value of the browser in which the project is tested */
    width:100%; /* 1496px is the defalut width value of the browser in which the project is tested */
}
body{
    margin-left:6.69%;/*100.2px; 100.2/1496=0.0669 = 6.69*/
    margin-right:6.69%;/*100.2px; 100.2/1496=0.0669 = 6.69*/
    width:89.27%;  /*1335.6px; 1335.6/1496= 0.8927 = 89.27% */
    background-color: darkseagreen;
}

.container {
    width:100%;/*1335.6px; */
    font-size:xx-large;
    text-align:center;
}
header {
    width:100%; /*1335.6px; 1335.6/1335/6 = 1 = 100*/
    height:14.17vh; /*100px; 100/705.6=0.1417 = 14.17 */
    background-color: bisque;
}
.main {
    width:74.87%; /*1000px; 1000/1335.6 = 0.7487 = 74.87*/
    float:left;
    height:42.51vh;/*300px; 300/705.6=0.4251 = 42.51*/
    background-color:beige
}
.aside{
    width:22.46%; /*300px; 300/1335.6 = 0.2246 = 22.46*/
    margin-left:2.66%; /*35.6px;*/
    float:right;
    height:42.51vh;/*300px; 300/705.6=0.4251 = 42.51*/
    background-color:azure;
}
footer{
    width:100%; /*1335.6px; 1335.6/1335/6 = 1 = 100*/
    margin-top:42.51vh;/*300px; 300/705.6=0.4251 = 42.51*/
    height:14.17vh; /*100px; 100/705.6=0.1417 = 14.17 */
    background-color: red;
}
```

Figure 11. Changing pixels to relative type sizes.

In figure 11, all the fixed values were changed to relative values with the formula derived by (Marcotte 2011) and shown in figure 6. Now, our webpage is flexible as shown in figure 12. It can adjust its content according to the screen size, the horizontal scrollbar is removed, and all the content, including the margin, is adjusted according to the available width as all values are in relational terms. Building a responsive webpage is not entirely about math. The formula shown in figure 6 which was derived by (Marcotte 2011) makes it easy to calculate those proportions but a fluid grid is just one foundation (Marcotte 2011).

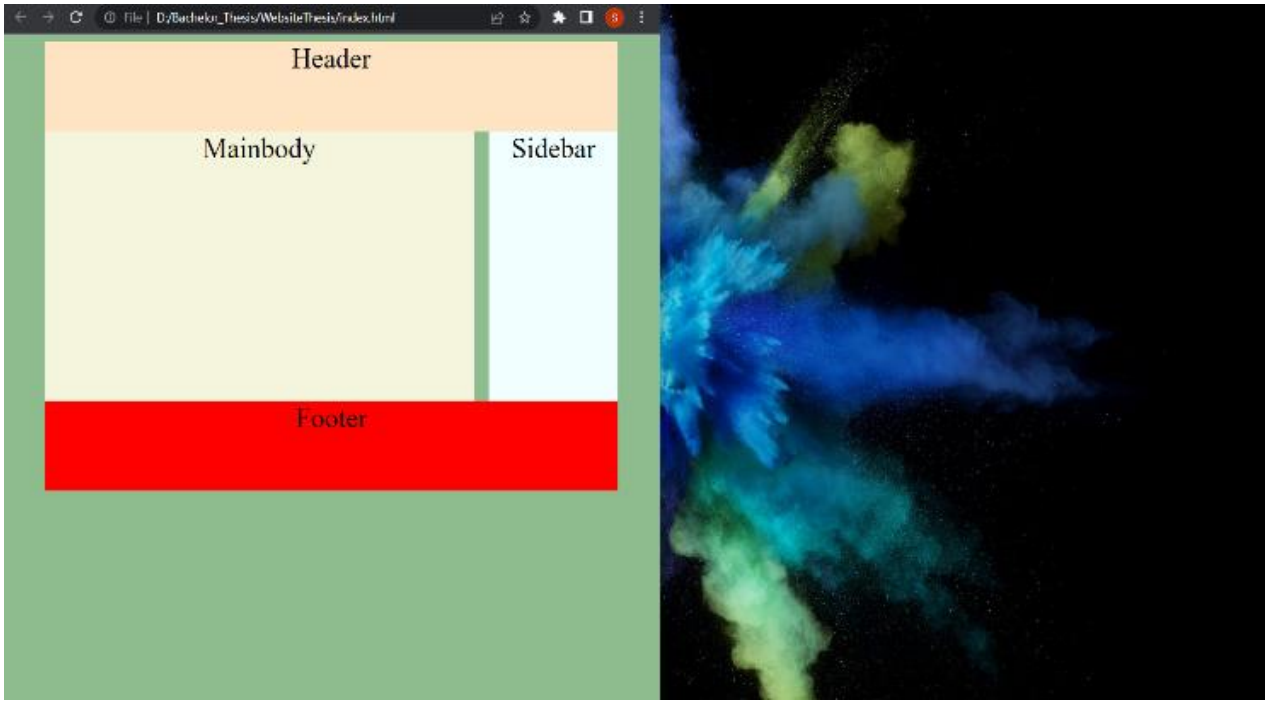


Figure 12. Flexible grid-based layout with relative values.

2.3.2 Adoptable pictures and media

A flexible container with 70vw must render an image and a video. The image and the video are a fixed-width elements. The width of the element with the image and video is set to 100% of its container but because our medias are much wider than its container, the excess content overflows its container as shown in figure 13.

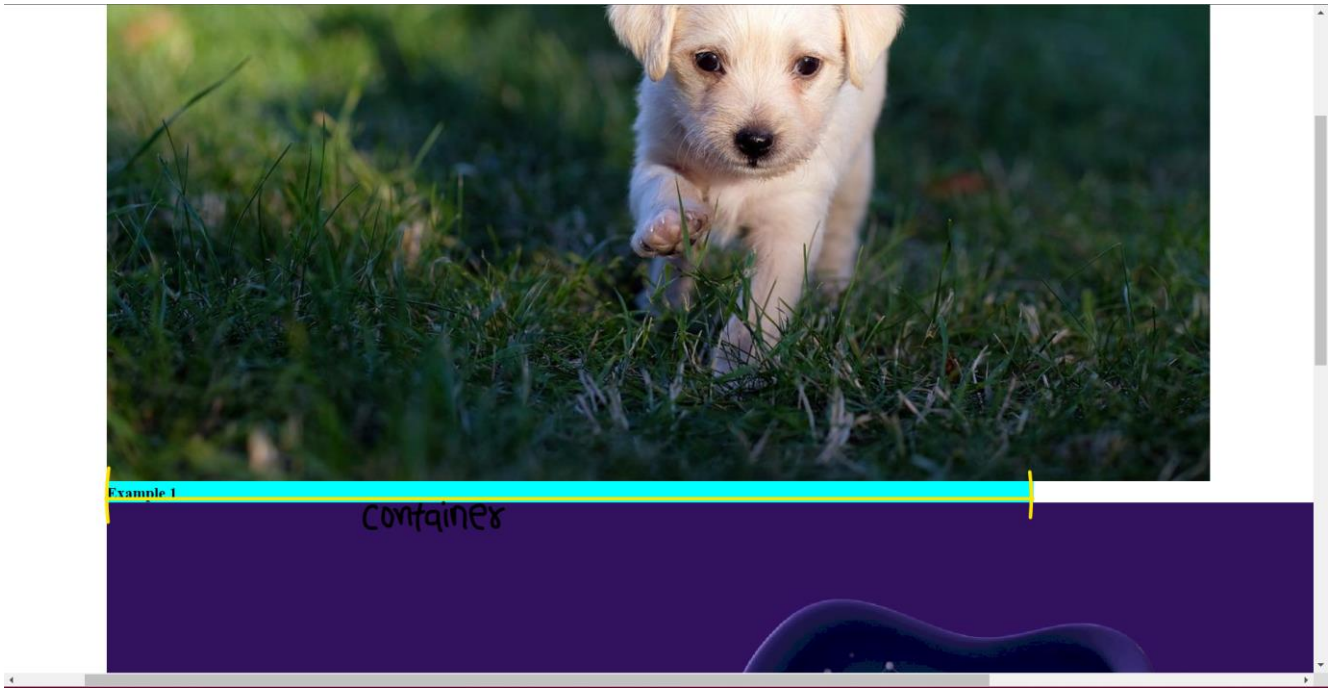


Figure 13. Image and video overflow its container.

This is because there are not any constraints used to our media element that could make it flexible. To do so, a constraint that prevents media from exceeding the width of their container must be implemented and the constraint is shown in figure 14.

```
img {
  max-width: 100%;
}
```

Figure 14. A constraint that prevents images from exceeding the width of their container (Marcotte 2011).

This constraint was discovered by Richard Rutter. This rule helps “img” element to render whatever size it wants, if it is narrower than its container and instructs images to never exceed the width of their containers, and whereas width:100% forces images to always match the width of their containing element. Rendering images and media was a problem for developers in the past because browsers did not support it, but modern browsers have evolved to the point where they resize the images proportionally as our

flexible container resizes itself (Marcotte 2011). This rule can also to most fixed-width elements, like video and other rich media. In figure 15, the max-width rule in img and video element were implemented.

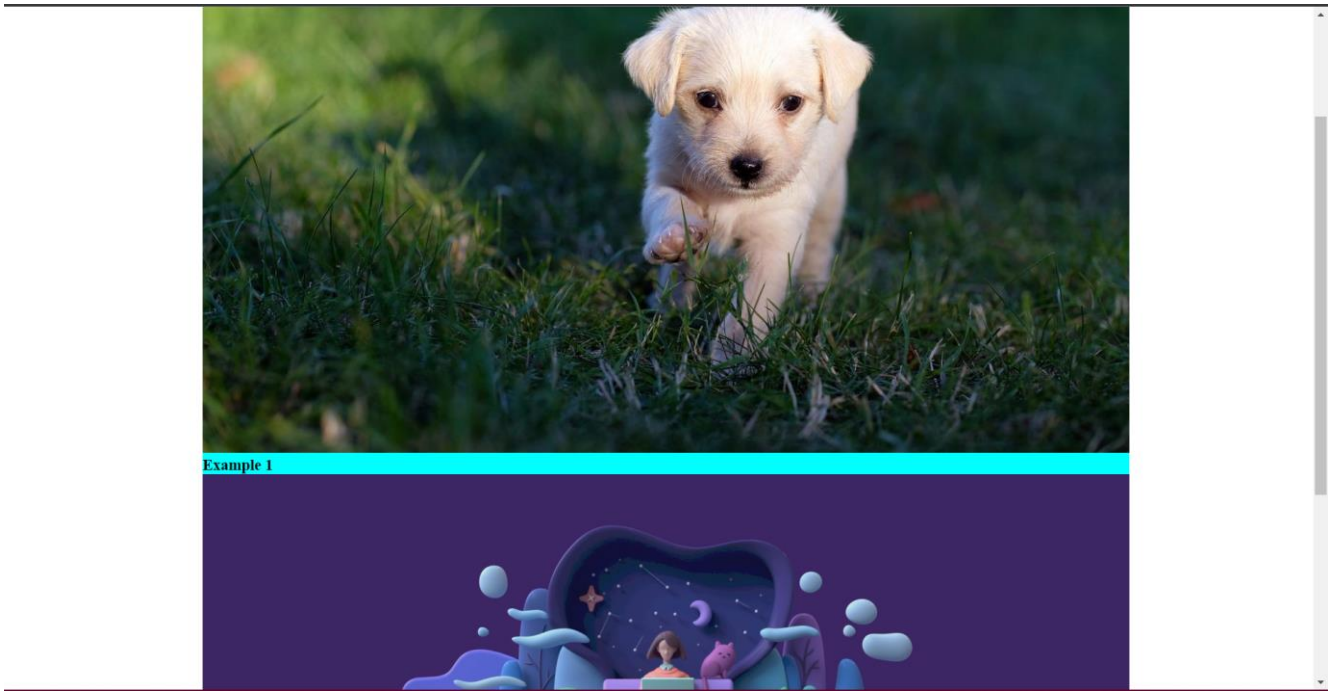


Figure 15. Media is nicely adjusted with the container.

The issue with this method is that it just adjusts the picture appearance on the display, leaving the actual image in the HTML text unaltered. This implies that users will continue to download the same picture resolution with a greater size regardless of their device or viewport size, resulting in unnecessary band-width use and possibly affecting site performance, particularly for mobile users. Also, scaling down the picture to very small might distort the image, especially if there is text inside of it, thus it is better to display an entirely separate image if the viewport width is too narrow, such as for phone view (Firdaus 2013).

```
<picture>
    <!-- When the vieport width is grater than 900px -->
    <source
    srcset="dog_landscape_large.jpg"
    media="(min-width:900px)"
    class=" img img_1" alt="image_1">

    <!-- When the vieport width is grater than 600px and less than 900px -->
    <source
    srcset="dog_landscape_small.jpg"
    media="(min-width:600px and max-width:900px)"
    class=" img img_1" alt="image_1">
    <!-- When the vieport width is smaller than 600px -->
    <source
    srcset="dog_portrait_small.jpg"
    media="(max-width:600px)"
    class=" img img_1" alt="image_1">

    <!--Default - If the browser do not support picture element -->
    <img src="dog_landscape_large.jpg" alt="image_1" class=" img img_1">

</picture>
```

Figure 16. Picture element with srcset and media.

To solve this problem World Wide Web Consortium (W3C), a group called Responsive Image Community Group (Groups) proposed a new element called <picture>. This picture element enables to set the source of the image based on a condition (Firdaus 2013). In figure 16, the picture element is implemented and three different source of images were included based on the condition. With the help of the picture element and a max-width contraint shown in figure 14, a image is now reponsive. Similarly, all other media can be responsive. Figure 17 shows the outcome after implementation of these contraints. Now, the image never overflow the containing element and the image changes according to the display in which it is rendered on.

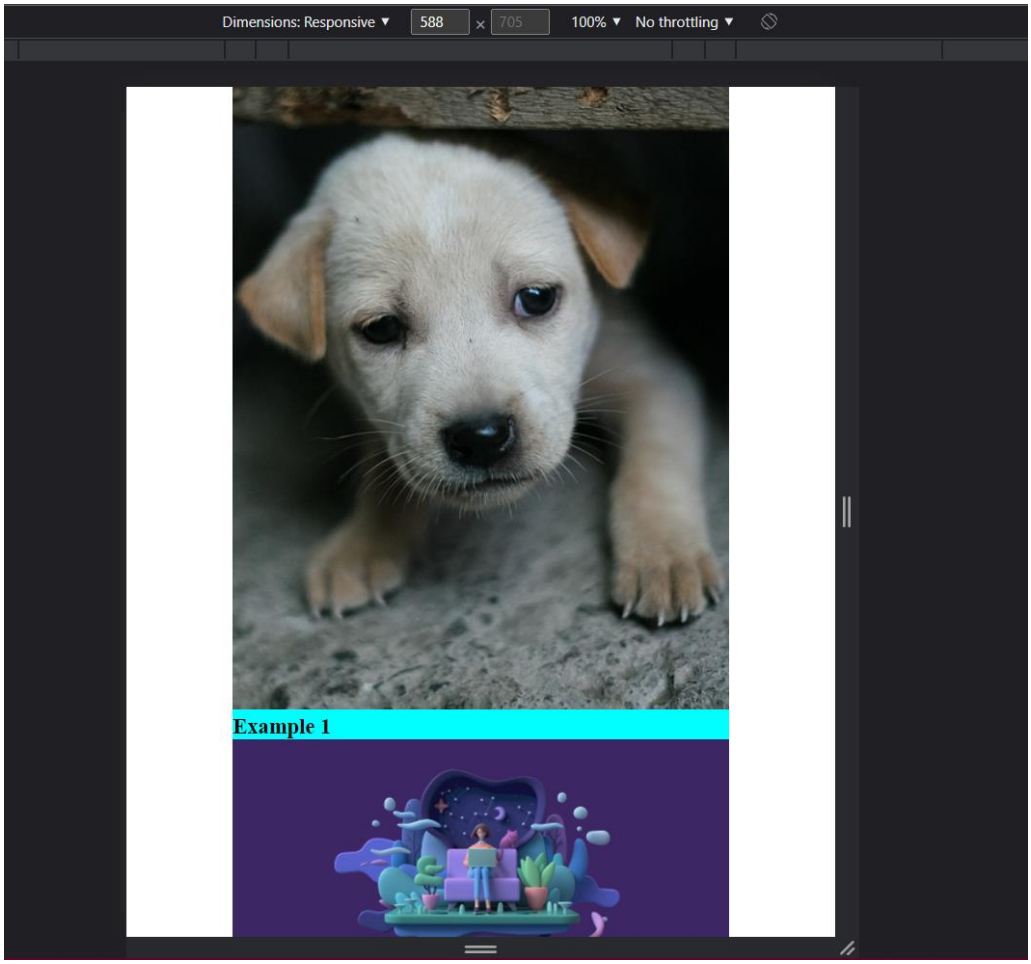


Figure 17. Condition set on <picture> element in result.

2.3.3 Media Queries

Media Queries is one of the CSS3 modules. It allows us to target specific CSS styles depending on the display capabilities of a device. In simple term, Media queries allow to customize the presentation of web pages without any modification of the markup by applying a condition on especially the viewport width. Media queries are already widely used and have a broad level of browser support (Firefox 3.6 +, Safari 4+, Chrome 4+, Opera 9.5+, iOS Safari 3.2+, Opera mobile 10+, Android 2.1+, and Internet Explorer 9+). Without the CSS3 media query module, it is not possible to target CSS styles at device capabilities, such as the viewport width (Frain 2012). Before Media queries, the W3C introduced media types, part of the CSS2 specification (Recommendation, Media types 2011). Media types were created so that web designers could load CSS for each type of browser or device. To address this, the W3C created a list of media types, attempting to classify each browser or device under a broad, media-specific

category. In practice, that meant customizing the media attribute of a link as shown in figure 18 (Marcotte 2011).

```
<link rel="stylesheet" href="global.css" media="all" />
<link rel="stylesheet" href="main.css" media="screen" />
<link rel="stylesheet" href="paper.css" media="print" />
```

Figure 18. Adding media types (Marcotte 2011).

By selecting @media now the developers could design the page for each type of browser or devices. This still did not fix our problem in the small screen browser as media type helps to figure out the device type like printer and browser. When small screen browsers, like phones and tablets, were popular, the designer could have simply added the handheld media type in the stylesheet as shown in figure 19 (Marcotte 2011).

```
<link rel="stylesheet" href="tiny.css" media="handheld"/>
```

Figure 19. Media type handheld (Marcotte 2011).

The problem with this approach was that the early mobile phones did not have sufficient capable browsers, so the designers largely ignored this approach. And when capable small-screen browsers finally did appear, there were not many handheld CSS files scattered in the web, which resulted in designers reading default screen-based stylesheets. Also, this approach was not future proof. Realizing the problems of media types, the W3C introduced media queries in CSS3 (W3CRecommendation 2022). Media queries are a mechanism for identifying not only the types of media but also inspecting the physical characteristics of the devices and browsers that render our content (Marcotte 2011). Since the media query is a logical expression, it can resolve to true or false. The query results will be true if the media type specified in the media query matches the device type on which the document is displayed, and if all expressions in the media query are satisfied. When a media query is true, the associated style sheet or style rule is applied to the target device. Like how a "if" statement would only execute a piece of code if the condition was true, media queries function as conditional statements (Hussain 2017).

```
@media screen and (min-width: 320px) and (max-width: 480px){
    .main{
        width:100%;
        float:none;
        min-height:300px;
       }
       .aside{
        width:100%;
        float:none;
        min-height:200px;
        margin-left:0px;
       }
       footer{
           margin-top:0px;
       }
}
```

Figure 20. Media queries in practice.

For Example, when the flexible layout is compressed to a very narrow viewport width. The content inside can also get narrow but with the help of media queries we can target a narrower viewport width and organize the layout. Although, the layout is flexible, but the element does not have enough space to render the minimum size of the content. It tries to compress the content which is not practical to user experience. Media queries can target the viewport width and customize the layout so that every element have enough space. In figure 12, the layout is responsive but if the viewport width is more narrowed then the content also gets compressed, and user might not get proper view of the content. In this situation media queries can be implemented and based on the condition if the viewport width is narrowed beyond a point than the media queries is used as shown in figure 20. Figure 21 shows the result after the implementation of media queries.

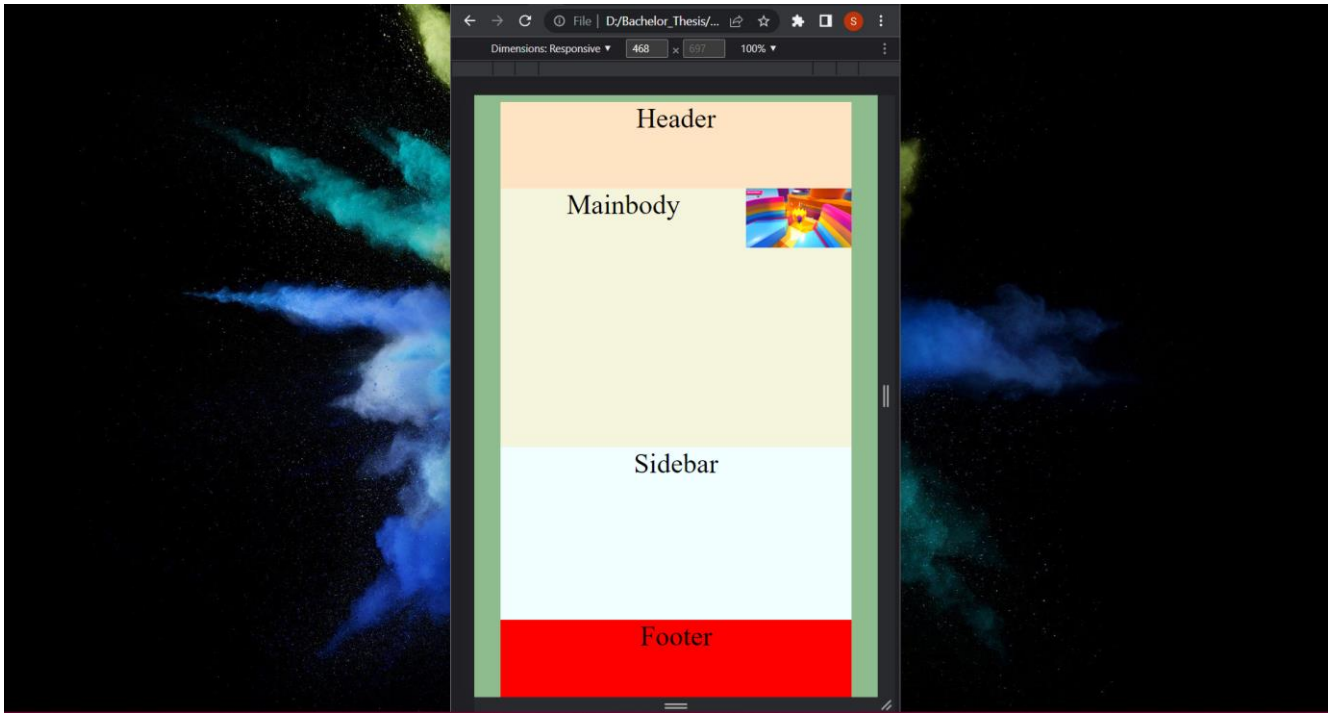


Figure 21. Media Queries worked!

3 PROJECT ANALYSIS AND DEVELOPMENT

As mentioned in the introduction, the primary goal of the thesis was divided. First, to understand the concept of responsive web design and its elements and second, to design a modern looking website fully responsive. To build a responsive web page, it is important to analyze the requirements of a responsive website.

3.1 Requirement Analysis

Analysing the requirements of the responsive website. Table 1 shows the element of responsive web design that must be implemented in the project.

Table 1. Requirements of the project website.

No	Requirement
1.	It should have Flexible text.
2.	It should have Flexible grid.
3.	It should have Flexible media and photos.
4.	It should have a proper use of Media queries for every screen size.

3.2 Used Web Technologies

Before working on any project, the first thing is to set up and analyze the tools that are required. Similarly, to build a website certain web technologies can be used. Table 2 shows the web technologies used for the project while building of the responsive website.

Table 2. Web technologies used for the project website.

Tools	**Used**
Virtual Studio Code	As a Software development tool.
HTML5	For the Content of the webpage.
SASS/CSS3	Overall Designing of the webpage.
CSS Grid and Flexbox	For responsive layout.
NPM (node-sass)	Compiler Package for SASS

3.2.1 Virtual Studio Code

Virtual Studio Code, also commonly referred to as VS Code, is a source code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. As shown in figure 22, VS Code was ranked the most popular developer environment tool (StackOverflow 2021). The reason of being the most popular code editor is its features and having these features in hand, VS Code is perfect for building web projects. As VS Code had a build-in web support, while working on web project no need to download any other extension. Table 3 shows some of the main features of virtual studio code.

Figure 22. Survey by Stack Overflow in 2021 (StackOverflow 2021).

Table 3. Popular features of virtual studio code.

Feature	**Detail**
Extension and Support	VS Code usually support all the programming language but, if not then programmer can down-land the required extension. Also, there are many extensions like Prettier which helps code formatting and clean code.
Intelli-Sense	It can detect if any snippet of code is left incomplete. Also, autocompletion of various syntaxes and declarations.
Terminal Support	In-build terminal or command prompt is very handy features for programmers.
Multi-Projects	Multiple projects containing multiple files/folders can be opened simultaneously.
Web Support	Build-in support for Web applications.
Git Support	Resource can be pulled from Git Hub and vice-versa.

3.2.2 HTML5

HTML 5 is a newly updated version of HTML. HTML5 removes superfluous markup elements, which both slow the data down and make it faster. HTML5 provides a new semantic element which provides more meaningful code to search engines and it also enables feedback to the user on basic site interactivity such as form submissions. This clear and faster loading code is good for responsive design. The <!DOC-TYPE html> declaration defines that this document is an HTML5 document. Viewport meta tag is one of the main components of responsive web design is added inside the <head> tag as shown in figure 23 and is used to control the scale of the web page. As adding the following viewport meta tag with initial scale set to 1 will allow the web page to be scaled by 100 percent of the viewport size (Firdaus 2013).

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Figure 23. Viewport meta tag.

Starting an HTML page is an essential part. The exact syntax of each tag inside the head section can be generated automatically in the code editor as shown in figure 24. In VS Code, this can be done by typing the "!" exclamation mark without quotes, and an emmet abbreviation will pop up, and pressing the tab key can generate the starting HTML5 code (Sekuloski 2021). The HTML page for the project is given in appendix 1.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Figure 24. Starting HTML file.

3.2.3 CSS3 and SASS

CSS3 is the updated version of CSS with different advance modules and features. CSS3 helps to solve a lot of problems while designing responsive websites. For example, the new viewport width and viewport height values in CSS3 makes it very handy to set the value of the elements in relation with the display viewport width and height it is rendered on. Table 4 shows the main features of CSS3.

Table 4. Popular features of CSS3.

Features	**Detail**
CSS Grids	CSS3 grids are used to build two-dimensional layout.
Flexbox	CSS3 flexbox is used to build one-dimensional layout.
Media Queries	With the help of @media, condition-based CSS can be created.
Gradients	Linear, Radial and Repeating gradients for backgrounds.
Transition, Transformations and Animations	With the help of transition, hover, and focus element effect in timing function. Also Rotate, Scale and Skew effects can be added with transformation and animate any object without flash.
Viewport height and width	New units for width(vw), height(vh) and font-size(vmin) which are in relative to device viewport.
Other	Box-shadow, Opacity, Border Radius and many more.

SASS is a CSS preprocessor, an extension of CSS that adds power and elegance to the basic language. There are two different ways to write SASS syntax. First is the normal SASS syntax and second is SCSS syntax. Both syntaxes are similar but based on the preference any one can be used. SCSS has a similar syntax like CSS, so it is widely popular. For this project, SCSS syntax was used and using a compiler later was compiled to CSS file as shown in appendix 2 . This pre-processor scripting language is compiled into cascading style sheets with the help of NPM package. Table 5 shows all the features of SASS.

Table 5. Features of SASS.

Feature	**Detail**
Variables	They are reusable values such as colors, font-sizes, etc.
Nesting	To nest selectors inside of one another, allowing us to write less code.
Operators	For mathematical operations inside of CSS.
Partials and imports	To write CSS in different files and importing them all into one single file.
Mixins	To write reusable pieces of CSS code.
Functions	Like @mixins, but they produce a value that can be used.
Extends	To make different selectors inherit declarations that are common to all of them.
Control directives	For writing complex code using conditionals and loops.

3.2.4 Flexbox and CSS Gird

CSS Flexile Box Layout, known as Flexbox, solves the shortfalls of existing layout techniques, such as inline-block, floats, and tables. It is one of the features of CSS3 and helps to build one-dimensional layouts. Using flexbox, the content can be vertically arranged, it makes it incredibly simple to reorder, align and organize item. With only fewer lines of code, Flexbox properties make it easier for items to align, direct and order in required place (Sekuloski 2021). Figure 25 shows flexbox that was used in the project website.

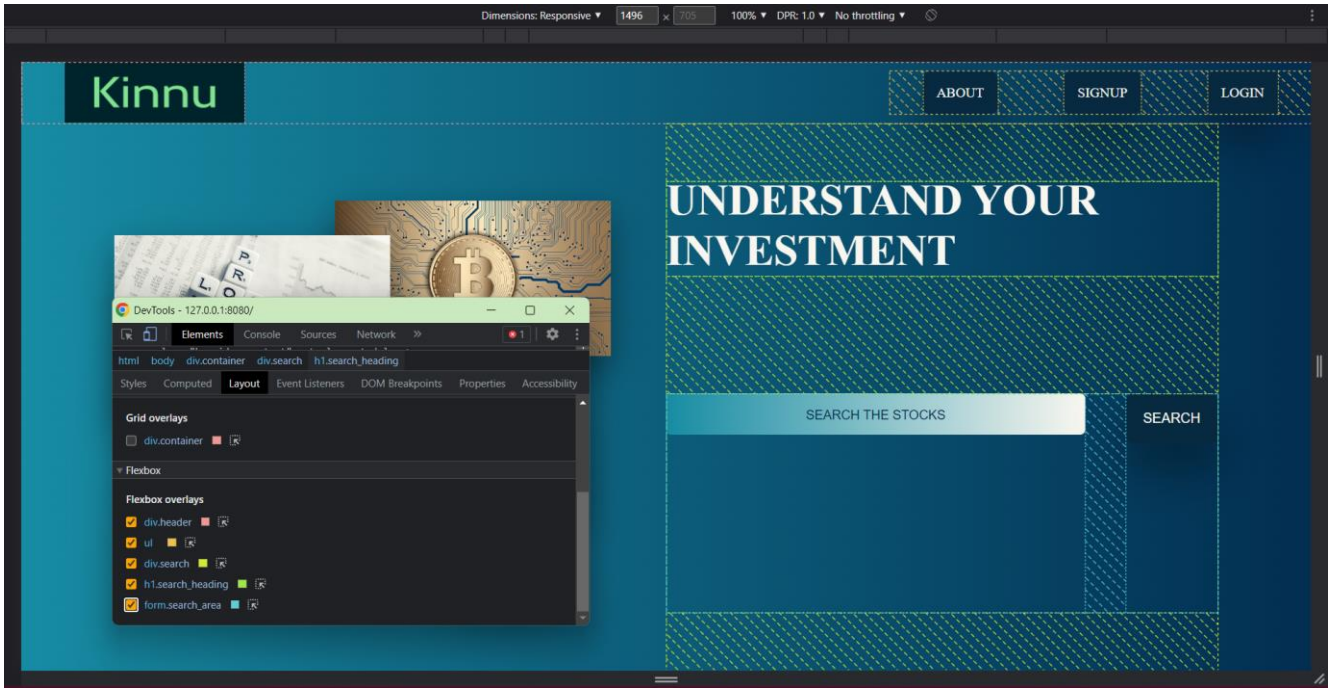


Figure 25.CSS flexbox used in the project.

The CSS Grid is without a doubt a revolutionary layout system. The primary distinction between the CSS Grid and Flexbox layout systems is that CSS grid is a two-dimensional system. Therefore, while working with a Flexbox container, we have two options for setting the layout: either we may arrange the objects in a row or a column. Therefore, we must utilize a two-dimensional system, such as the CSS Grid system, if we wish to arrange objects along two axes (Sekuloski 2021). Flexbox is substantially less powerful than CSS Grid. This does not imply that Flexbox should be exclusively replaced with CSS Grid. But both systems can be utilized and mix their greatest features. A stunning responsive layout can be created by fusing this layout features. There are no set criteria for what to employ; depending on the circumstances. (Sekuloski 2021). In this project, we have used both CSS gird and Flexbox depending on the requirements. Figure 26 shows the CSS grid used for the project website.

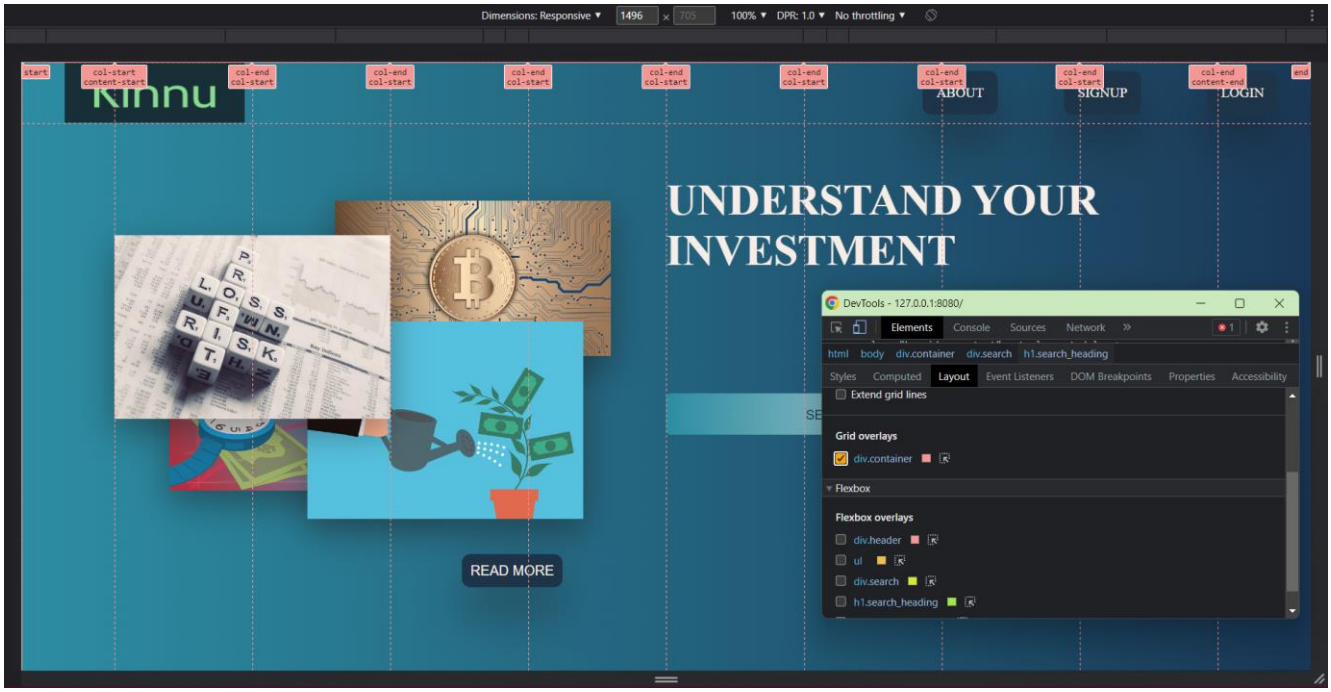


Figure 26. CSS grid used in the project.

3.2.5 NPM

NPM is a simple command line interface that allows developers to install and manage packages on their local computers. There are all kinds of open-source tools, libraries and frameworks needed for modern development. NodeJS and NPM package ecosystem is very often used by developers. In this project we will need only one package that is a compiler of SASS. To do so we first download NodeJS and then before installing our package, the first thing we always do is to create a package. json file which will contain the definitions of our project and where NPM will write the packages that is used. To do that, we use the NPM command. This will create a package.json file in the folder of the project. Using the NPM command line, we can now install the SASS compiler as shown in figure 27.

```
PS D:\Programming\JavaScript\basics-02-added-scripts-imports> npm install sass-node --save-dev
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
[..................] - fetchMetadata: sill resolveWithNewModule is-fullwidth-code-point@1.0.0 checking installable status
```

Figure 27. Installation of a SASS compiler using NPM.

The package “sass-node” which compile SASS to CSS will be saved on as a dev dependency in package.json file. In the scripts of the package.json file, we then specify the name of the package and location

where it gets compiled. Now, with the help of the script name which is “compile:sass” in this project, we can run a NPM command which will compile the SASS to CSS as shown in figure 28.

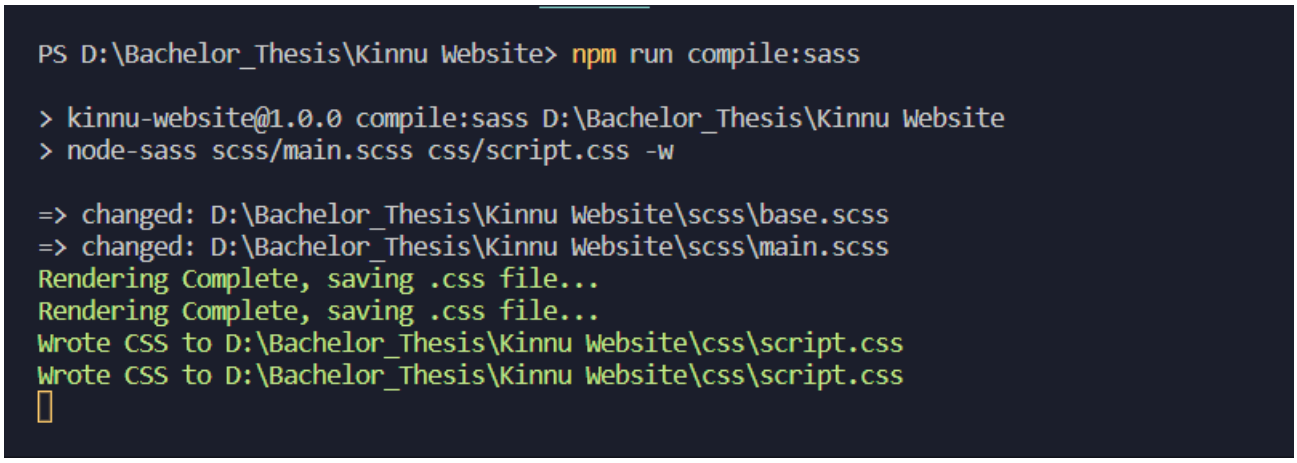


Figure 28. Running the SCSS compiler.

3.3 Development

The design of the webpage was influenced by Shopify(https://www.shopify.com/). Using virtual studio code, project was set up. The main plan of the project was to implement all the concept of creating a responsive web design. It is crucial for website to have maintainable code so with the influence of the book (Martin 2008), the project is setup with clean code.

3.3.1 Website without RWD Elements

To understand the concept of the responsive design, the website was first thoroughly designed without the help of elements of responsive web design that was coined by Ethan Marcotte in his book (Marcotte 2011).

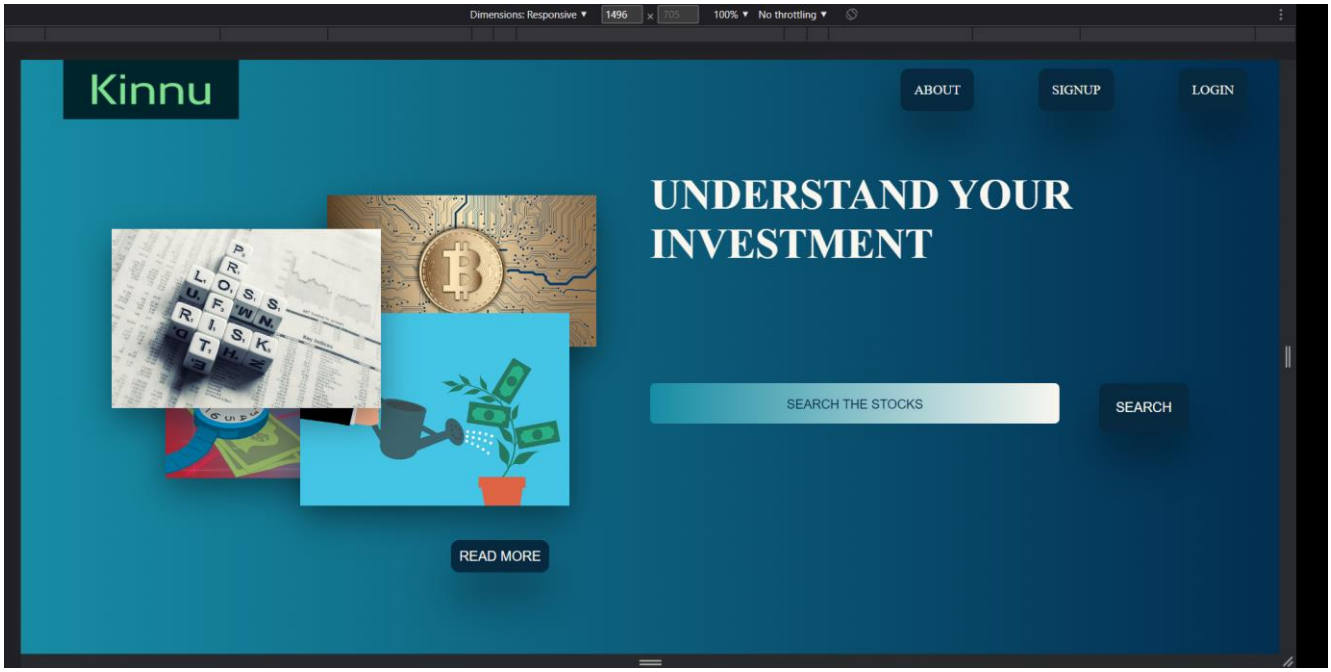


Figure 29. Website design without RWD elements.

This website looks good in figure 29, but the website is not responsive. The webpage was created on desktop-first view. It is only applicable for desktop view which has min-width:1200px to max-width 1800px. If the viewport width is decrease or increase, it will exactly show, the important website to be responsive in figure 30.

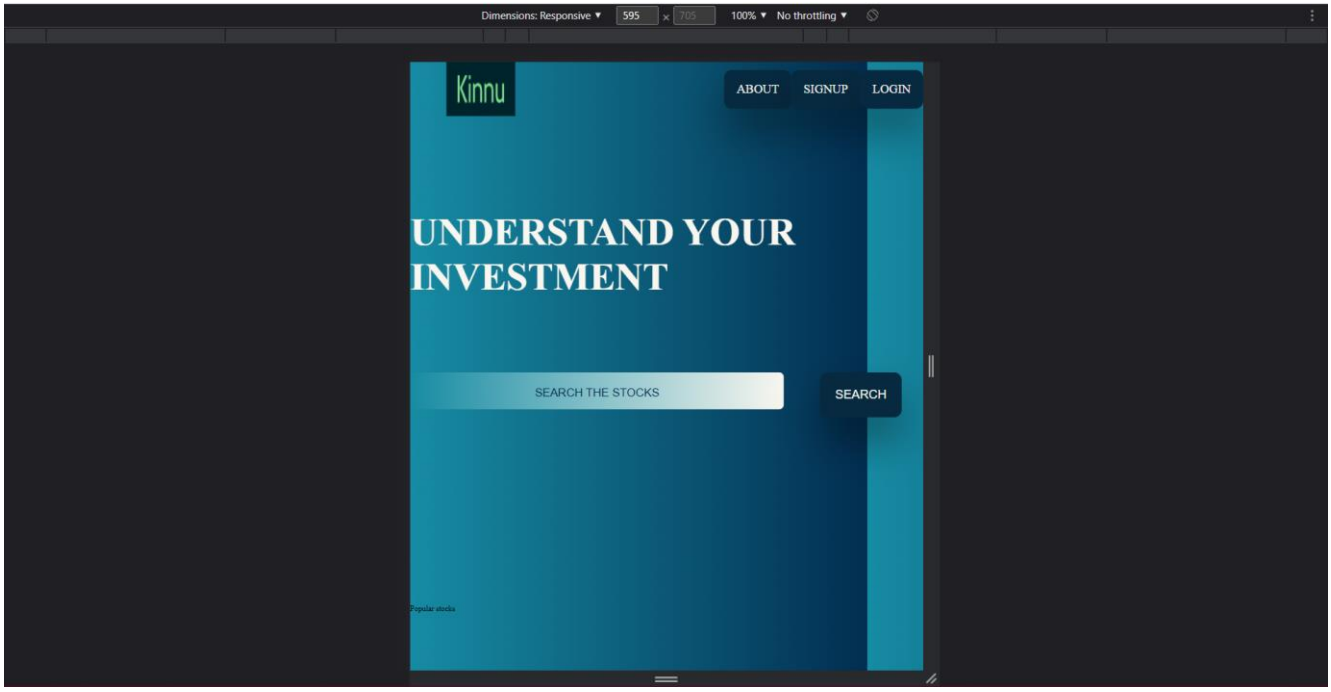


Figure 30. Decreasing the viewport width of the website.

Figure 30 is the mobile view of the website. If a user wanted to open the website in a phone, this is what the user experience. As the website was created on a fixed values so the element will not be adjusting its content according to the width of the display. The font-size of the main element is set in fixed pixel value, but it should be in relative value. Similarly, all the values of every element are in pixels. If the current design is compared and analysed with the requirement of the project website from table 1, the result is shown in table 6.

Table 6. The result of comparing and analyzing the current design with the requirement of the project website.

Requirement	**Yes/No**	**Reason**
Flexible text	No	Fixed values in text
Flexible grid	No	Fixed values in height and width.
Flexible images	No	Not used
Media Queries	No	Not used

3.3.2 Implementing RWD Elements

Analysing the requirements of the responsive website, the following should be taken into consideration. First, all the fixed values set on the CSS properties should be change to proportional values. In the book, Responsive Web design, Ethan Marcotte invented a formula to determine the exact values in relative term (Marcotte 2011). Using that we change the fixed values pixel to relative values as shown in figure 31.

```
// Normal 16 px = 1 rem;
// Ethan Marcotte formula;
// target/context=result
html{
    font-size:62.5%; // 10px; 10/16=0.625 =62.5%(1 rem= 10px;)
}
```

Figure 31. Relative value is set on font-size of the html element.

Similarly, all other property including width and height values were set on rem (a CSS unit relative to the font-size of an html element) as shown in figure 32. This result to make all the elements of html in proportion with the font-size of the html element. If the font-size of the html element is changed, all other elements will adapt with the change. This is a one of the important steps forward for building a responsive web design.

```
// Ethan Marcotte formula:
// target/context=result
&_heading{
    // context is the height of the parent element = 635.04px
    // target is the required px = 190.5px
    height:17.24%; //190.5px; // 190.5/635.04 = 0.1724 = 17.24%

    display:flex;
    align-items:end;

    // context is based on the main font-size that we set = 1 rem = 10 px;
    // target is the required px= 50px;
    font-size: 5rem;// 50px; // 50/10 = 5

    text-transform: uppercase;
    font-weight: 900;
    color:$white;
}
&_area{
    // context is the height of the parent element = 640px
    // target is the required px = 640px
    width:100%;// 640px; // 640/640=1 =100%;

    // context is the height of the parent element = 635.04px
    // target is the required px = 254.01px
    height:39.99%; // 254.01/635.04= 0.3999 = 39.99%

    display:flex;
    flex-direction: row;
    align-items:start;
    justify-content: space-between;
    &_box{
        // context is the height of the parent element = 640px
        // target is the required px = 486.4px
        width:76%;// 486.4px; // 486.4/640 = 0.76 =76%

        // context is the height of the parent element = 254.01px
        // target is the required px = 48.25px
        height:18.99%;// 48.25px; // 48.25/254.01 =0.1899 = 18.99%
```

Figure 32. Everything is set on relative type value with the font-size of html.

Similarly, the property of images is set in percentage in relation with the parent element. Also, the constraint max-wifth:100% was implemented to make sure the media never overflows the containing element. If a user rendered the website in small devices like phone than the web browser does not want to download a big image rather small size image will save the data and helps in good performance of the website. With the help of “srcset” attribute in HTML we can specify multiple images of different size

or quality. Together with the size attribute they create responsive images that adjust according to media connection or browser size as shown in figure 33.

```
<div class="stock_gallary">
            <div class="gallary">
                <img srcset="./Extra/image_1_small.jpg 640w,
                             ./Extra/image_1_large.jpg 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_1" class="gallary_photo gallary_photo_1"
                    src="./Extra/image_1_large.jpg"
                >

                <img srcset="./Extra/image_2_small.jpg 640w, ./Extra/image_2_large.jpg 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_2" class="gallary_photo gallary_photo_2"
                    src="./Extra/image_2_large.jpg"
                >
                <img srcset="./Extra/image_3_small.png 640w, ./Extra/image_3_large.png 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_3" class="gallary_photo gallary_photo_3"
                    src="./Extra/image_3_large.png"
                >
                <img srcset="./Extra/image_4_small.png 640w, ./Extra/image_4_large.png 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_4" class="gallary_photo gallary_photo_4"
                    src="./Extra/image_4_large.png"
                >
            </div>
```

Figure 33. “srcset” attribute for responsive images.

There are many numbers of devices with different display sizes. It is impossible to design for each. So, a common breakpoint is determined for few devices. For this project, 4 breakpoints were chosen including phone, tablet-portrait, tablet-landscape, and big-desktop. The website was designed on desktop-first view, so the default design is for desktop size. Using a @mixin Sass directive, which helps to create CSS code that can be reused throughout the website. For example, creating a @mixin for button, which later can be used to design all other button throughout the website. Similarly, with the help of @mixin, @content and @if SASS directives, I have implemented media queries as shown in figure 34.

```
@mixin respond($breakpoint){
    @if $breakpoint == big-desktop{
        @media only screen and (min-width: 112.5em) {@content};  //1800px//Ethen Marcotte Formula; target/context=result 1800/16=112.5em
    }
    @if $breakpoint == tab-land{
        @media only screen and (max-width: 75em) {@content}; //1200px //1200/16= 75 em
    }
    @if $breakpoint == tab-port{
        @media only screen and (max-width: 56.25em) {@content}; //900px //900/16= 56.25em
    }
    @if $breakpoint == phone{
        @media only screen and (max-width: 37.5em) {@content}; //600px //600/16=37.5em
    }
}
```

Figure 34. Implementing breakpoints with the help of @content, @if and @mixin SASS directives.

We can now use @include <name> to include @mixin to the required elements of html. So, wherever the change is needed, we can include the media queries with the help of @mixin as shown in figure 35.

```
// Normal 16 px = 1 rem;
// Ethan Marcotte formula;
// target/context=result
html{
    font-size:62.5%; // 10px; 10/16=0.625 =62.5%(1 rem= 10px;)
    @include respond(big-desktop){
            font-size: 87.5%;  // 14px; 14/16=0.875 =87.5%
        }

    @include respond(tab-land){
            font-size: 56.25%; // 9px; 9/16=0.5625 =56.25%
        }
    @include respond(tab-port){
            font-size: 43.75%; // 7px; 7/16=0.4375 = 43.75%
        }
    @include respond(phone){
            font-size: 31.25%; // 5px; 5/16=0.3125 =31.25%
        }
}
```

Figure 35. Adding the created @mixin for media queries with the help of @include for html font-size.

4 RESULTS

A responsive website which can adapt its content according to the screen sizes is created with the help of the different element and components of responsive web design including flexible layout and text, adoptable pictures, media queries, CSS grid and flexbox and different relative term values like rem, viewport width, and viewport height. The adaptability of the webpage was tested on various breakpoints shown in appendix 3, and the result was successful. Table 7 compares the final design with the requirement design for the project website and every section was marked.

Table 7. The result of comparing and analyzing the updated design with the requirement of the project website.

Requirement	**Yes/No**	**Reason**
Flexible text	Yes	"rem" value is used throughout for responsive text as the value is relative to the font-size of an html element.
Flexible grid	Yes	Flexible CSS grid and Flexbox were used, and relative values were set on width and height.
Flexible images	Yes	Images and its size changes according to the viewport width.
Media Queries	Yes	Media Queries are implemented for four different breakpoints.

5 CONCLUSION

The theory was primarily driven by two goals. Understanding the theory of a responsive website was the initial objective. Another objective was to put into practice the knowledge and construct a basic prototype responsive website that can adjusts it content. With these objectives in mind, the thesis began with theoretical subjects about responsive web design and its components, with the support of in-depth research from several books. To implement the project, a basic understanding on learn how to create a website from scratch was needed, so with the help of a comprehensive course from Udemy a lot of problem were solved (https://www.udemy.com/course/advanced-css-and-sass/). A website was created with the help of the course and research from several front-end development books. The source code of the website is uploaded to GitHub (https://github.com/RegmiSagar-2001/Responsive-Web-Design-Project).

In addition to the development and design of the website, a thorough analysis of several web service technologies and development frameworks was carried out. Numerous software development technologies were understood practically. In summary, the goal was accomplished when comparing the initial objective of the thesis with the results. Additionally, numerous important lessons regarding a variety of web development topics, including the foundation of NodeJS and the entire NPM package ecosystem, various Sass directives, CSS grid, Flexbox, project planning, and implementation, and clean code, were learned.

REFERENCES

BroadbandSearch. 2019. *Mobile Vs. Desktop Usage (Latest 2019 Data).* https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics.

Falvain , Carlos, Raquel Gurrea, and Carlos Orús. 2009. "Web Design: a key factor for the website success." *Journal of Systems and Information Technology* (Emerald Group Publishing Limited) 11: 168-184. https://doi.org/10.1108/13287260910955129.

Firdaus, Thoriq. 2013. *Responsive Web Design by Example Beginner's Guide.* Birmingham: Packt Publishing Ltd.

Frain, Ben. 2012. *Responsive Web design with HTML5 and CSS3.* Packt.

Groups, W3C Community and Business. n.d. *Responsive Issues Community Group.* https://www.w3.org/community/respimg/.

Hussain, Frahaan. 2017. *Responsive Web Design by Example.* Birmingham: Packt Publishing Ltd.

Marcotte, Ethan. 2011. *Responsive Web Design.* New York: Jeffrey Zeldman.

Martin, Robert Cecil. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall, 2009.

Recommendation, W3C. 2011. "Media types." *www.w3.org.* 07 06. Accessed 12 07, 2021. https://www.w3.org/TR/CSS21/media.html#media-types.

Sekuloski, Rick. 2021. *Web Design with HTML5 and CSS: Learn how to design, create and built responsive websites using the best HTML5 and CSS practices.* ISBN.

StackOverflow. 2021. "Stack Overflow Developer Survey 2021." *stackoverflow.com.* https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-integrated-development-environment.

W3CRecommendation. 2022. “Media Queries Level 3.” *www.w3.org*. 05 04. https://www.w3.org/TR/mediaqueries-3/.

W3Schools. 2019. *CSS Introduction.* https://www.w3schools.com/Css/css_intro.asp.

W3Schools. 2019. *CSS Syntax and Selectors.* https://www.w3schools.com/css/css_syntax.asp.

W3Schools. 2022. *Introduction to HTML*. https://www.w3schools.com/html/html_intro.asp.

APPENDIX 1

HTML file of the project

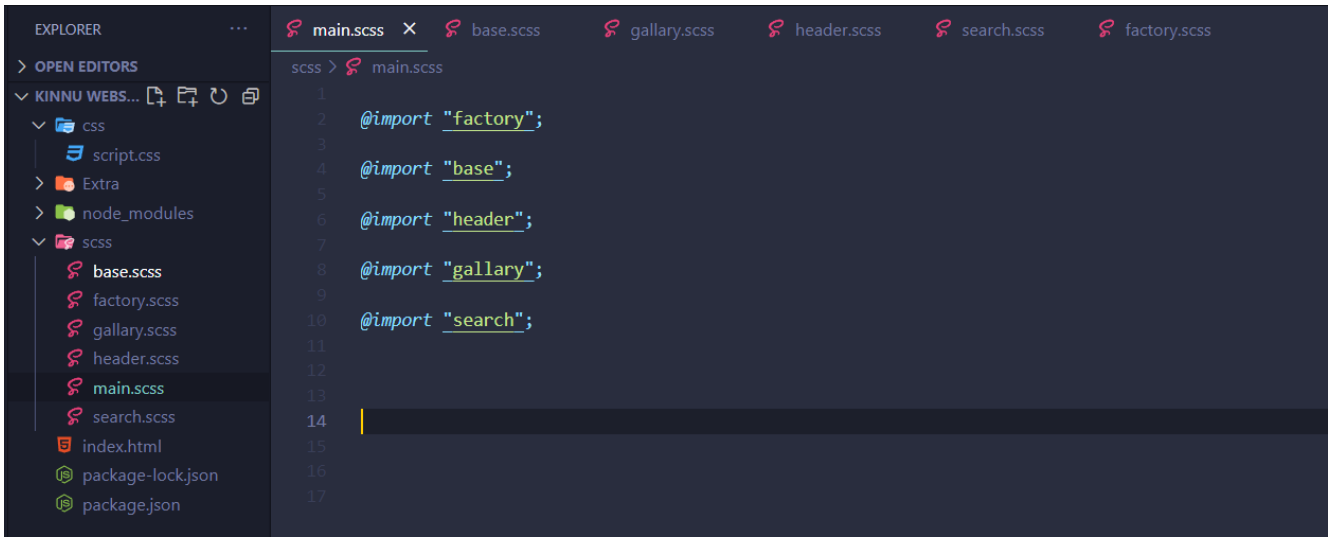
index.html > html > body > div.container > div.stock_gallary > div.gallary

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>
        Kinnu
    </title>
    <link rel="stylesheet" href="./css/script.css">
</head>
<body>
    <div class="container">
        <!-- <div class="bg-video">
            <video src="./Extra/intro_video.mp4" type="intro_video.mp4" class="bg-video-content" autoplay muted loop>
                Your browser is not supported!
        </video> -->
        <div class="header">
            <img src="Extra/cover.png" alt="" class="logo">
            <ul>
                <li><a href="https://www.google.com/" class="about-btn">About</a></li>
                <li><a href="https://www.google.com/" class="signup-btn">SignUp</a></li>
                <li><a href="https://www.google.com/" class="login-btn">Login</a></li>
            </ul>

        </div>
<div class="stock_gallary">
            <div class="gallary">
                <img srcset="./Extra/image_1_small.jpg 640w,
                             ./Extra/image_1_large.jpg 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_1" class="gallary_photo gallary_photo_1"
                    src="./Extra/image_1_large.jpg"
                >

                <img srcset="./Extra/image_2_small.jpg 640w, ./Extra/image_2_large.jpg 1280w"
```

```
                <img srcset="./Extra/image_2_small.jpg 640w, ./Extra/image_2_large.jpg 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_2" class="gallary_photo gallary_photo_2"
                    src="./Extra/image_2_large.jpg"
                >
                <img srcset="./Extra/image_3_small.png 640w, ./Extra/image_3_large.png 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_3" class="gallary_photo gallary_photo_3"
                    src="./Extra/image_3_large.png"
                >
                <img srcset="./Extra/image_4_small.png 640w, ./Extra/image_4_large.png 1280w"
                    sizes="(max-width: 900px) 27vw, (max-width: 600px) 50vw, 320px"
                    alt="gallary_4" class="gallary_photo gallary_photo_4"
                    src="./Extra/image_4_large.png"
                >
            </div>
            <button class="btn-readmore">Read more</button>
        </div>
        <div class="search">
            <h1 class="search_heading">Understand your investment</h1>
            <form class="search_area">
                <input type="text" placeholder="Search the stocks" class="search_area_box">
                <button class="search_area_btn">Search</button>
            </form>
        </div>
    </div>
</body>
</html>
```

APPENDIX 2

Creating SCSS file for every component and importing all file to main.scss file.



Compiled CSS

css > script.css > ...

```css
* {
  box-sizing: border-box;
  margin: 0;
  Padding: 0; }

html {
  font-size: 62.5%; }
  @media only screen and (min-width: 112.5em) {
    html {
      font-size: 87.5%; } }
  @media only screen and (max-width: 75em) {
    html {
      font-size: 56.25%; } }
  @media only screen and (max-width: 56.25em) {
    html {
      font-size: 43.75%; } }
  @media only screen and (max-width: 37.5em) {
    html {
      font-size: 31.25%; } }

body {
  background-image: linear-gradient(to right, #178CA4, #022E51); }

.container {
  height: 100vh;
  width: 100%;
  display: grid;
  grid-template-rows: 10vh 90vh min-content;
  grid-template-columns: [start] 1fr [content-start] repeat(8, [col-start] minmax(min-content, 16rem) [col-end]) [content-end] 1fr [end]; }
  @media only screen and (min-width: 112.5em) {
    .container {
      grid-template-columns: [start] 1fr [content-start] repeat(8, [col-start] minmax(min-content, 20rem) [col-end]) [content-end] 1fr [end]; } }
  @media only screen and (max-width: 75em) {
    .container {
      grid-template-columns: [start content-start] repeat(8, [col-start] minmax(min-content, 16rem) [col-end]) [content-end end]; } }
  @media only screen and (max-width: 37.5em) {
    .container {
      width: 100%;
      grid-template-rows: [heading-start start] 15vh [heading-end news-start] 50vh [news-end search-start] 50vh [search-end end];
      grid-template-columns: [start content-start] repeat(4, 25vw) [content-end end]; } }

.header {
  display: flex;
  height: 10vh;
  justify-content: space-around;
  align-items: center;
  grid-column: start/end;
  align-content: center; }
  @media only screen and (max-width: 37.5em) {
    .header {
      grid-row: heading-start/heading-end;
      grid-column: start/end; } }

.logo {
  width: 15vw;
  height: 100%;
  margin-left: 5rem; }
  @media only screen and (max-width: 37.5em) {
```

```
  @media only screen and (max-width: 37.5em) {
    .logo {
      width: 35vw; } }
  @media only screen and (max-width: 37.5em) {
    .logo {
      height: 90%; } }

ul {
  margin-left: 50vw;
  width: 35vw;
  top: 0;
  right: 0;
  text-decoration: none;
  list-style: none;
  display: flex;
  flex-direction: row;
  justify-content: space-around; }
  @media only screen and (min-width: 112.5em) {
    ul {
      margin-left: 25vw; } }
  @media only screen and (max-width: 75em) {
    ul {
      margin-left: 35vw; } }
  @media only screen and (max-width: 37.5em) {
    ul {
      margin-left: 10vw; } }
  @media only screen and (max-width: 37.5em) {
    ul {
      width: 45vw; } }
  @media only screen and (min-width: 112.5em) {
    ul {
      width: 30vw; } }
  @media only screen and (max-width: 37.5em) {
    ul {
      justify-content: space-between; } }

a {
  text-decoration: none;
  color: #F9F7F0; }

li {
  display: inline-block;
  border: none;
  border-radius: 1rem;
  text-align: center;
  cursor: pointer;
  text-transform: uppercase;
  color: #F9F7F0;
  background-color: #072A40;
  padding: 1.6rem;
  box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
  transition: all .3s;
  font-size: 1.6rem; }
  li:hover {
    background-color: black;
    transform: scale(1.1);
    box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); }
  @media only screen and (max-width: 56.25em) {
```

```
  @media only screen and (max-width: 56.25em) {
    li {
      display: inline-block;
      border: none;
      border-radius: 1rem;
      text-align: center;
      cursor: pointer;
      text-transform: uppercase;
      color: #F9F7F0;
      background-color: #072A40;
      padding: 1.8rem;
      box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
      transition: all .3s;
      font-size: 1.7rem; }
      li:hover {
        background-color: black;
        transform: scale(1.1);
        box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }
  @media only screen and (max-width: 75em) {
    li {
      display: inline-block;
      border: none;
      border-radius: 1rem;
      text-align: center;
      cursor: pointer;
      text-transform: uppercase;
      color: #F9F7F0;
      background-color: #072A40;
      padding: 2rem;
      box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
      transition: all .3s;
      font-size: 1.9rem; }
      li:hover {
        background-color: black;
        transform: scale(1.1);
        box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }

.stock_gallary {
  grid-column: content-start/6;
  position: relative;
  transition: all .2s; }
  @media only screen and (max-width: 75em) {
    .stock_gallary {
      grid-column: content-start/5; } }
  @media only screen and (max-width: 37.5em) {
    .stock_gallary {
      grid-row: news-start/news-end;
      grid-column: start/end; } }

.btn-readmore {
  position: absolute;
  top: 50rem;
  right: 12rem;
  font-size: 1.4rem;
  display: inline-block;
  border: none;
  border-radius: 1rem;
  text-align: center;
```

```
    border-radius: 1rem;
    text-align: center;
    cursor: pointer;
    text-transform: uppercase;
    color: #F9F7F0;
    background-color: #072A40;
    padding: 1rem;
    box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
    transition: all .3s;
    font-size: 1.6rem; }
    .btn-readmore:hover {
      background-color: black;
      transform: scale(1.1);
      box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); }
    @media only screen and (max-width: 75em) {
      .btn-readmore {
        top: 50rem;
        display: inline-block;
        border: none;
        border-radius: 1rem;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        color: #F9F7F0;
        background-color: #072A40;
        padding: 2rem;
        box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
        transition: all .3s;
        font-size: 1.6rem; }
        .btn-readmore:hover {
          background-color: black;
          transform: scale(1.1);
          box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }
    @media only screen and (max-width: 56.25em) {
      .btn-readmore {
        top: 60rem;
        display: inline-block;
        border: none;
        border-radius: 1rem;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        color: #F9F7F0;
        background-color: #072A40;
        padding: 2rem;
        box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
        transition: all .3s;
        font-size: 1.6rem; }
        .btn-readmore:hover {
          background-color: black;
          transform: scale(1.1);
          box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }
    @media only screen and (max-width: 37.5em) {
      .btn-readmore {
        top: 62rem;
        display: inline-block;
        border: none;
        border-radius: 1rem;
```

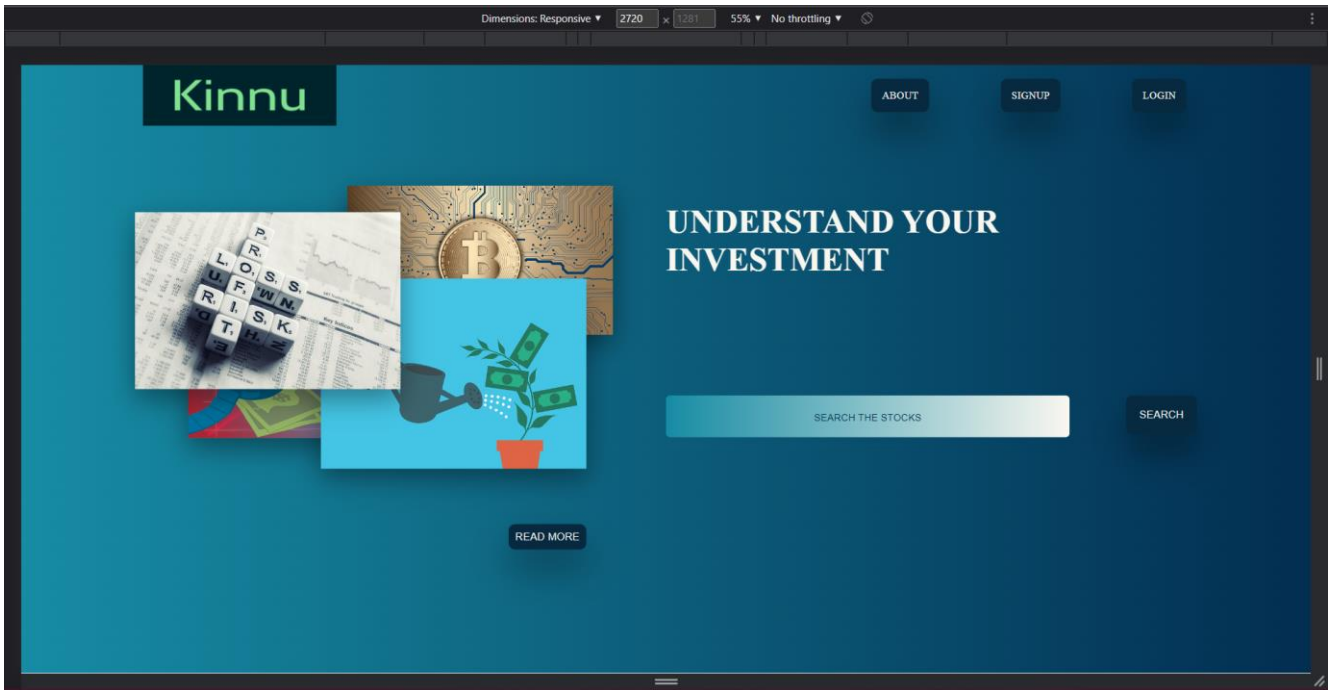
```
        border-radius: 1rem;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        color: #F9F7F0;
        background-color: #072A40;
        padding: 1.8rem;
        box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
        transition: all .3s;
        font-size: 1.6rem; }
        .btn-readmore:hover {
          background-color: black;
          transform: scale(1.1);
          box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }
    @media only screen and (min-width: 112.5em) {
      .btn-readmore {
        top: 60rem;
        display: inline-block;
        border: none;
        border-radius: 1rem;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        color: #F9F7F0;
        background-color: #072A40;
        padding: 1rem;
        box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
        transition: all .3s;
        font-size: 1.6rem; }
        .btn-readmore:hover {
          background-color: black;
          transform: scale(1.1);
          box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); } }

  .gallary {
    width: 100%;
    position: relative; }
    @media only screen and (max-width: 37.5em) {
      .gallary {
        margin-left: 5rem; } }
    .gallary:hover .gallary_photo:not(:hover) {
      transform: scale(0.9); }
    .gallary_photo {
      width: 50%;
      position: absolute;
      box-shadow: 0 1.5rem 4rem rgba(0, 0, 0, 0.4);
      z-index: 1;
      transition: all .2s;
      outline-offset: .6rem; }
      .gallary_photo:hover {
        outline: 0.8rem solid #022E51;
        box-shadow: 0 3rem 5rem rgba(0, 0, 0, 0.8);
        transform: scale(1.05);
        z-index: 100; }
      .gallary_photo_1 {
        top: 13rem;
        left: 0;
```

```
        left: 0;
        z-index: 10; }
        @media only screen and (max-width: 56.25em) {
          .gallary_photo_1 {
            top: 20rem; } }
      .gallary_photo_2 {
        top: 9rem;
        left: 40%; }
        @media only screen and (max-width: 56.25em) {
          .gallary_photo_2 {
            top: 14rem; } }
      .gallary_photo_3 {
        top: 25rem;
        left: 10%; }
        @media only screen and (max-width: 56.25em) {
          .gallary_photo_3 {
            top: 30rem; } }
      .gallary_photo_4 {
        top: 23rem;
        left: 35%; }
        @media only screen and (max-width: 56.25em) {
          .gallary_photo_4 {
            top: 28rem; } }

  img {
    max-width: 100%; }

  .search {
    grid-column: 6/content-end;
    display: flex;
    align-items: center;
    justify-content: space-around;
    flex-direction: column;
    align-content: center; }
    @media only screen and (max-width: 75em) {
      .search {
        grid-column: 5/content-end; } }
    @media only screen and (max-width: 37.5em) {
      .search {
        grid-row: search-start/search-end;
        grid-column: start/end;
        margin-top: 20rem; } }
    @media only screen and (max-width: 56.25em) {
      .search {
        justify-content: center; } }
    .search_heading {
      height: 17.24%;
      display: flex;
      align-items: end;
      font-size: 5rem;
      text-transform: uppercase;
      font-weight: 900;
      color: #F9F7F0; }
      @media only screen and (max-width: 37.5em) {
        .search_heading {
          align-items: center;
          text-align: center; } }
```

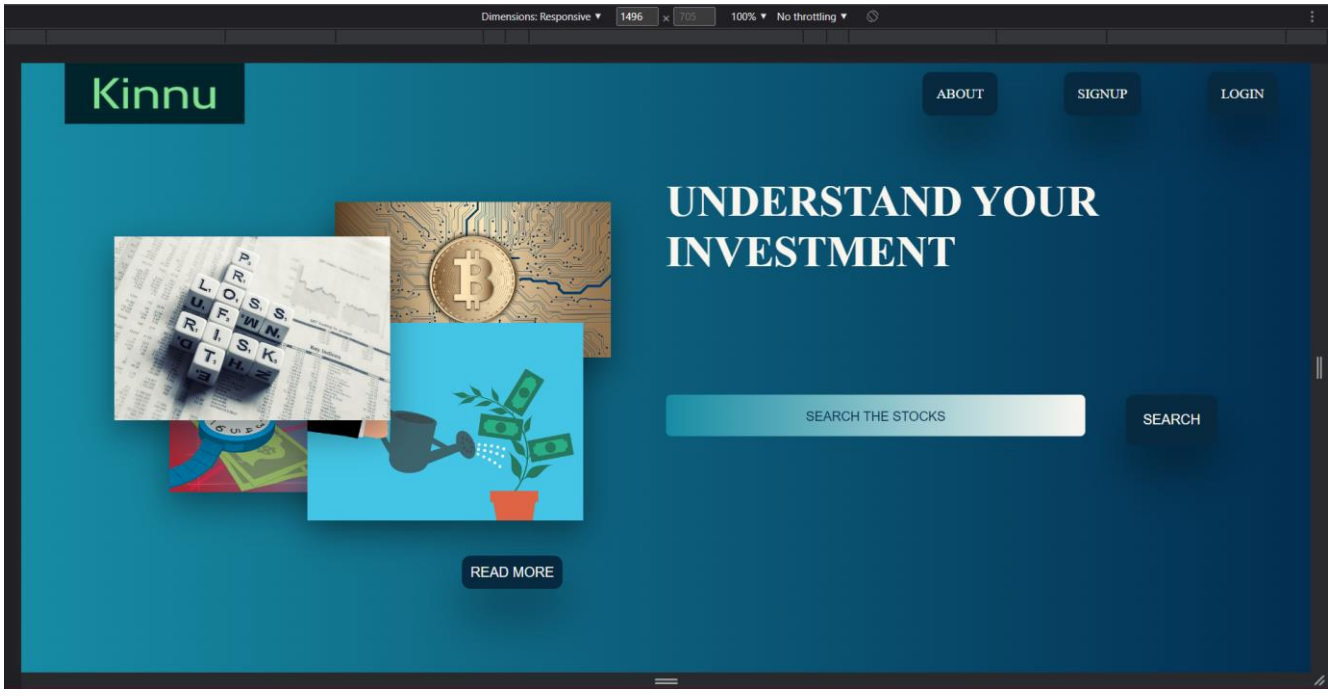
```
        text-align: center; } }
    .search_area {
      width: 100%;
      height: 39.99%;
      display: flex;
      flex-direction: row;
      align-items: start;
      justify-content: space-between; }
      @media only screen and (max-width: 56.25em) {
        .search_area {
          margin-top: 8rem; } }
      @media only screen and (max-width: 37.5em) {
        .search_area {
          width: 85%; } }
      .search_area_box {
        width: 76%;
        height: 18.99%;
        border-radius: 0.6rem;
        background: linear-gradient(to right, #178CA4, #F9F7F0);
        color: black;
        border: none;
        font-size: 2rem;
        font-weight: 100;
        transition: all .3s; }
        .search_area_box:hover {
          background-color: #e3dab9;
          transform: scale(1.1);
          box-shadow: 0 2rem 4.5rem rgba(0, 0, 0, 0.4); }
        .search_area_box::placeholder {
          color: #022E51;
          text-transform: uppercase;
          text-align: center;
          font-size: 1.5rem; }
        .search_area_box:focus {
          border: none;
          outline: none;
          transform: scale(0.95); }
      .search_area_btn {
        display: inline-block;
        border: none;
        border-radius: 1rem;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        color: #F9F7F0;
        background-color: #072A40;
        padding: 2rem;
        box-shadow: 0 3rem 5.5rem rgba(0, 0, 0, 0.4);
        transition: all .3s;
        font-size: 1.6rem; }
        .search_area_btn:hover {
          background-color: black;
          transform: scale(1.1);
          box-shadow: 0 4rem 6rem rgba(0, 0, 0, 0.4); }
```

APPENDIX 3

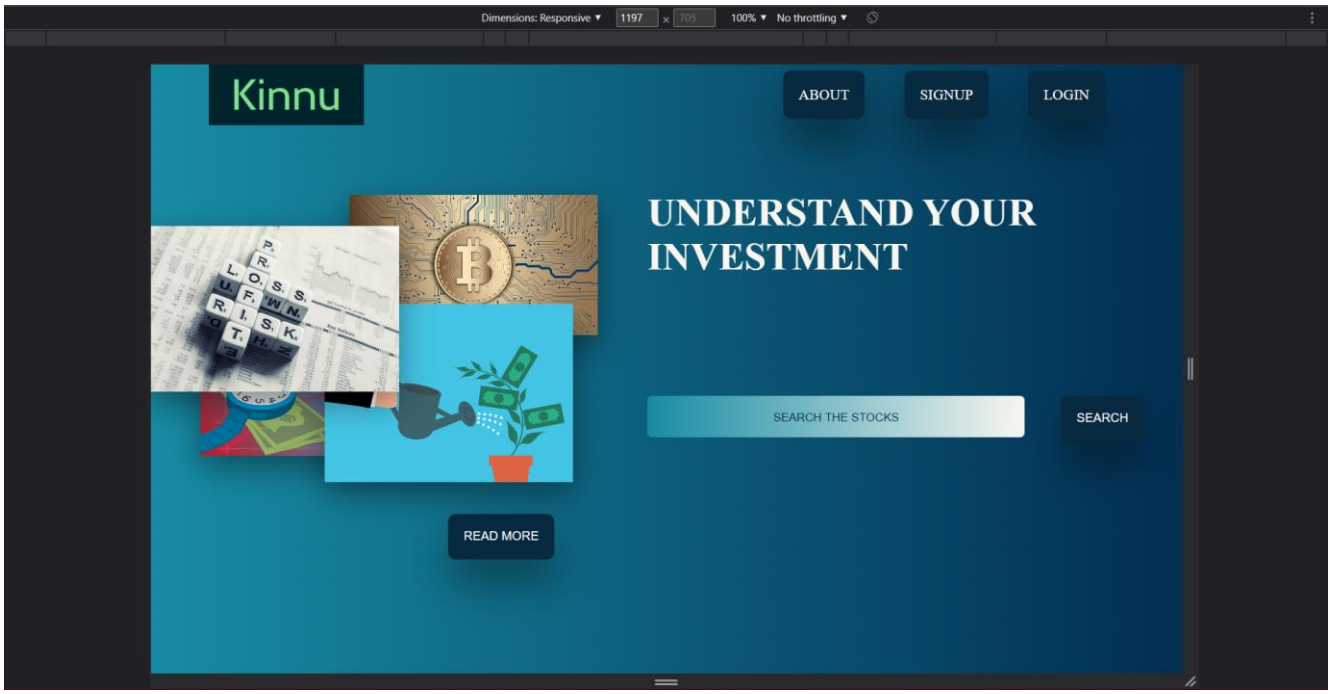
The website on various breakpoints.



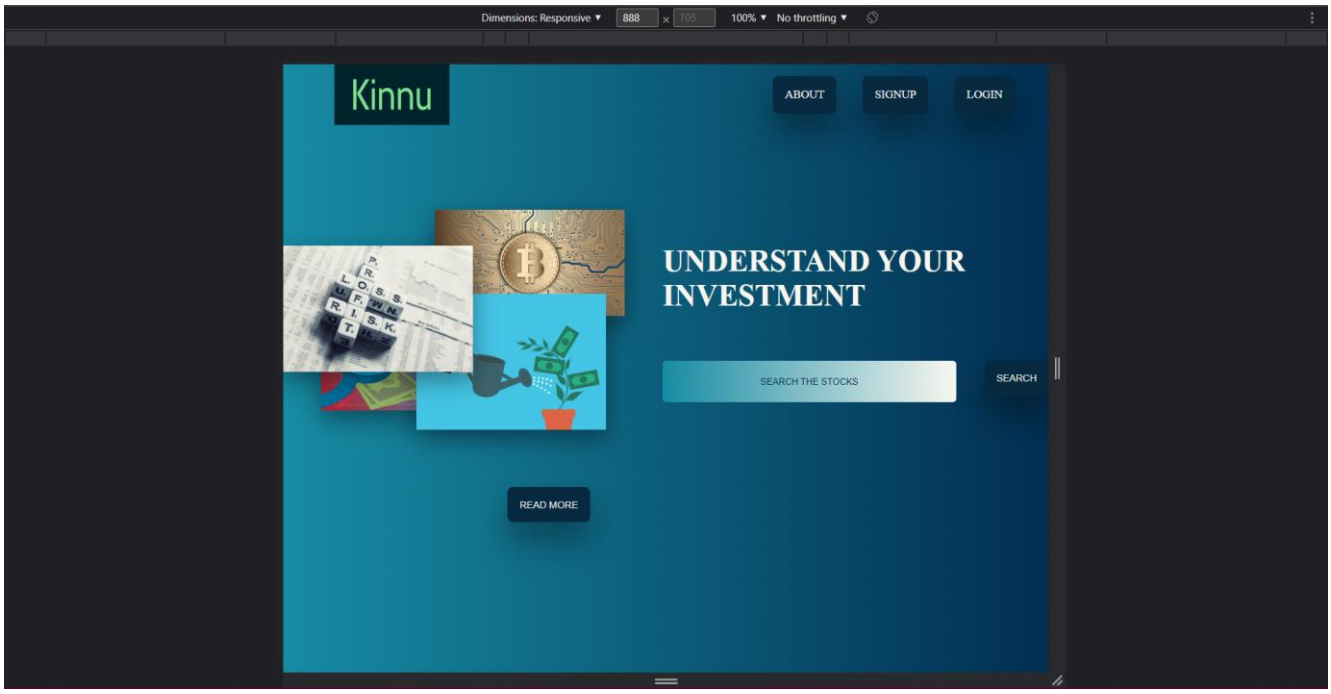
Website in big screen-view (1800px and above).



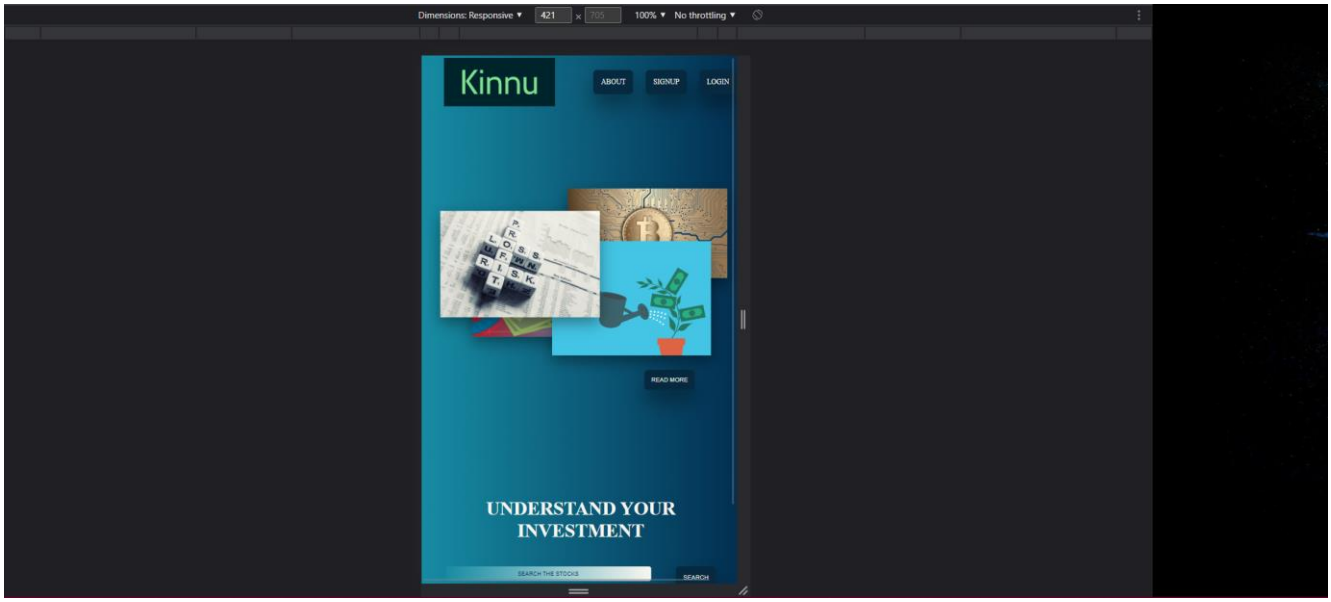
Website in desktop-view (1200px-1800px).



Website in table landscape-view (900px-1200px).



Website in tablet portrait-view (600px-900px).



Website in phone-view (600px and below).