

WEB TEMPLATE CUSTOMIZING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in
Electronics and Communication Engineering.

BY

RAMYALAKSHMI M (38130183)

ARTHI D (38130016)



DEPARTMENT OF ELECTRONICS& COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600119**

MARCH 2022



BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAMYALAKSHMI.M**(Reg. No. 38130183) and **ARTHI.D**(Reg. No. 38130016) who carried out the project entitled "**WEB TEMPLATE CUSTOMIZING**" under our supervision from December 2021 to April 2022.

Internal Guide

Mrs.L.Magthelin Therase.M.E.,

Head of the Department

Dr.T. Ravi, M.E., Ph.D.,

Submitted for Viva voice Examination held on _____

Name:

Signature:

Internal Examiner

External Examiner

DECLARATION

We **RAMYALAKSHMI.M**(38130183) and **ARTHI.D**(38130016) hereby declare that the Project Report entitled “**WEB TEMPLATE CUSTOMIZING**” done by us under the guidance of **MRS.L.MAGTHELIN THERASE.M.E.**, at Sathyabama Institute of Science and Technology, **Chennai** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

1.

2.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGMENT

We are pleased to acknowledge our sincere thanks to the Board of Management of SATHYABAMA for their kind encouragement in doing this project and completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. Nandhitha, Dean, School of Electrical and Electronics Engineering**, and **Dr. T. RAVI, Head of the Department, Dept. of Electrical and Electronics Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would also like to express our sincere and deep sense of gratitude to our internal guide , Mrs.L.Magthelin Therase.M.E., for her valuable guidance and suggestions during the progressive reviews for our project.

ABSTRACT

There is always a misconception and misunderstanding between the client and the company during the design process. Because, the client has to make their own designs which is not easy and also time consuming. Markup medium also using coding. It consumes more time. Negotiation is also an important problem. because, there is a huge variations in the price of designs. so, This ready made template sections will resolve this.

By choosing their desired part, with their desired budget. Both the company and the client will be happy. This is implemented by adding an extra feature in website which contains many sections with available designs.

Client can choose and summate then, email to the respective company by their registered mail id. Then the company construct a beautiful website of what client exactly wants.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION	
	1.1 WHAT IS WEB DESIGN	9
	1.2 DESIGN IS A KEY PART OF WEB DESIGN	9
	1.3 WEB DESIGN HAS MANY DIFFERENT ROLES	10
2.	LITERATURE SURVEY	12
3.	HTML-STRUCTURING THE WEB	
	3.1 HTML BASICS	14
	3.2 WHAT IS HTML	14
	3.3 ANATOMY OF AN HTML ELEMENT	15
	3.4 NESTING ELEMENTS	16
	3.5 EMPTY ELEMENTS	16
	3.6 ANATOMY OF AN HTML DOCUMENT	17
	3.7 IMAGES	18
	3.8 MARKING UP TEXT	19
	3.8.1 HEADINGS	19
	3.8.2 PARAGRAPH	19
	3.8.3 LISTS	19

4.	CSS	
	4.1 WHAT IS CSS	20
	4.2 CSS-SYNTAX	20
	4.3 SELECTOR	21
	4.3.1 TYPE SELECTOR	21
	4.3.2 UNIVERSAL SELECTOR	21
	4.3.3 DESCENDANT SELECTORS	21
	4.3.4 CLASS SELECTORS	22
	4.3.5 ID SELECTORS	23
	4.4 CSS-COLOUR	24
	4.4.1 CSS COLORS- HEX CODES	24
5.	BOOTSTRAP	
	5.1 ITS RESPONSIVE GRID	25
	5.2 ITS RESPONSIVE IMAGES	25
	5.3 ITS COMPONENT	25
	5.4 ITS JAVASCRIPT	26
	5.5 ITS DOCUMENTATION	26
	5.6 ITS CUSTOMIZABILITY	26
	5.7 ITS COMMUNITY	27
	5.8 ITS EXTERNAL TEMPLATES	27
6.	RESPONSIVE WEBSITE	
	6.1 SET APPROPRIATE RESPONSIVE BREAKPOINTS	28
	6.2 START WITH FLUID GRID	28
	6.3 TAKE TOUCHSCREENS INTO CONSIDERATION	29
	6.4 DEFINE TYPOGRAPHY	29
	6.5 USE PRE-DESIGNED THEME OR LAYOUT TO SAVE TIME	29
7.	6.6 TEST RESPONSIVENESS ON REAL DEVICES	
	ANGULAR-FRONT END APPLICATION	30
	7.1 SERVER(BACK-END) APPLICATION	31
8.	7.2 CREATE THE ANGULAR COMPONENT	33
9.	FIREBASE HOSTING	36
10.	RESULTS AND DISCUSSIONS	41
	SUMMARY AND CONCLUSION	42
	REFERENCES	

CHAPTER 1

INTRODUCTION

1.1 WHAT IS WEB DESIGN:

Web design is the planning and creation of websites. This includes a number of separate skills that all fall under the umbrella of web design. Some examples of these skills are information architecture, user interface, site structure, navigation, layout, colors, fonts, and overall imagery. All of these skills are combined with the principles of design to create a website that meets the goals of the company or individual from whom that site is being created. This article will take a look at the basics of website design and the various disciplines or skills that are a part of this industry.

1.2 DESIGN IS KEY PART OF WEB DESIGN:

Design, obviously, is a key part of "web design." What does this mean exactly? Design includes both the principles of design — balance, contrast, emphasis, rhythm, and unity — and the design elements — lines, shapes, texture, color, and direction. By putting these things together, a web designer creates websites, but a good web designer understands not only the principles of design but also the constraints of the Web. For example, a successful web designer will be skilled in typographic design principles, while also understanding the challenges of web type design and specifically, how it differs from other kinds of type design. In addition to understanding the limitations of the Web, a successful web professional also has a firm grasp of the strengths of digital communication.

1.3 WEB DESIGN HAS MANY DIFFERENT ROLES:

When you work as a web designer, you may be tasked with creating (or working on) entire sites or just individual pages and there is a lot to learn to be a well-rounded designer, including the following:

- **HTML** — This is the structure of web pages, creating the foundation of all websites
- **CSS** — This is how web pages are visually styled. CSS (Cascading Style Sheets) handles the entire look of sites, including layout, typography, colors, and more
- **JavaScript** — This governs certain behaviors on websites and can be used for a variety of interactions and features
- **CGI programming** — CGI, and the next few entries (PHP, ASP, etc.) are all different flavors of programming languages. Many sites do not require any of these languages, but sites that are more feature-rich will certainly need to be coded using some of these languages
- **PHP, ASP, ColdFusion scripting**
- **XML**
- **Information architecture** — The way a site's content and navigation is structured and presented helps make for a successful site that is easy and intuitive to use
- **SEO** — Search engine optimization ensures that websites are attractive to Google and other search engines and that people looking for the products, services, or information features on that site can find it once they look for it online

- **Server management** — All websites need to be hosted. The management of the servers that host those sites is an important web design skill
- **Web strategy and marketing** — Having a website is not enough. Those sites will also need to be marketed with an ongoing digital strategy
- **E-commerce and conversions**
- **Design** — Creating the visual look and feel of websites has always been an important aspect of the industry
- **Speed** — A successful site is one that loads quickly on a wide variety of devices, regardless of a visitor's connection speed. Being able to tune the performance of sites is a very valuable skill
- **Content** — People come to websites for the content that those sites contain. Being able to create that content is a critically important component in the world of website design

There's also so much more areas and skills that cross into the field of web design, but most designers don't try to cover all of them. Instead, a web designer will generally focus on one or two areas where they can excel. The other items in web design that are needed are ones that they can partner with others on as part of a larger web design team.

CHAPTER 2

LITERATURE SURVEY

2.1 Garrett,Renee,Chiu,Jason,Zhang,Ly,Young,Sean D:

Inferences

In this paper, we have seen web design and user engagement where Proper design has become a critical element needed to engage website and mobile application users. Little research has been conducted to define the specific elements used in effective design and to define a short list of elements frequently used in research. Proper design has become a critical element needed to engage website and mobile application users. However, little research has been conducted to define the specific elements used in effective website and mobile application design. Review and consolidate research on effective design and to define a short list of elements frequently used in research. The design elements mentioned most frequently in the reviewed literature were navigation, graphical representation, organization, content utility, purpose, simplicity, and readability.

2.2 Md.Shamsul Araffin,Yi Jiang

Inferences

In this paper, we have seen that Developing a dynamic website using the online website builder weebly for viking fortune where Website usability,the website as a marketing tool and an understanding of the requirements of a good website. This bachelor's thesis consists of two sections, a theoretical section and implementation. The theoretical section provides a literature review of website usability, the website as a

marketing tool and an understanding of the requirements of a good website. The implementation section focuses on creating a functional website for the case company by using an online website building platform. Qualitative research methods have been used in this thesis project.

2.3 Tan shin yen, Siti Hajar Halili

Inferences

In this paper, we have seen that Exploring the use of word press in a literature lesson based on Assure model where wordpress in a literature lesson based on assure model. This study explored the effectiveness on using of blog specifically WordPress in a literature lesson based on ASSURE model. A qualitative study was employed by the researcher and interviews were conducted with five secondary students in a Malaysian secondary school. The literature module used in this study was developed using the ASSURE model which acts as a guide for lesson planning and instruction which involves instructional technology. In terms of findings, three key themes identified: (1) “I may not favor blogging, but it is beneficial for my learning”, (2) “I tend to self-reflect more than I constructively contribute”, and (3) “Blogging opens my eyes to learning literature”. With the rapid expansion of technology, the study provided wider implications and help institutions to understand which methods work best to develop course materials and techniques appropriately for students. Literature has always been regarded as an area too abstract and complex, resulting in its reputation of being a 'dry' subject for students. Nevertheless, given the proper guidance and conducive platform, students' literature response can be highly enhanced; the act of blogging can cater to such needs.

CHAPTER 3

HTML- STRUCTURING THE WEB

3.1 HTML basics :

HTML (**H**yper**T**ext **M**arkup **L**anguage) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and its functions.

3.2 WHAT IS HTML :

HTML is a *markup language* that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on. For example, take the following line of content:

My cat is very grumpy

If we wanted the line to stand by itself, we could specify that it is a paragraph by enclosing it in paragraph tags

```
<p>My cat is very grumpy</p>
```

3.3 ANATOMY OF AN HTML ELEMENT:

The main parts of our element are as follows:

1. **The opening tag:** This consists of the name of the element (in this case, `p`), wrapped in opening and closing **angle brackets**. This states where the element begins or starts to take effect — in this case where the paragraph begins.
2. **The closing tag:** This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
3. **The content:** This is the content of the element, which in this case, is just text.
4. **The element:** The opening tag, the closing tag, and the content together comprise the element.

Elements can also have attributes that look like the following:

Attributes contain extra information about the element that you don't want to appear in the actual content. Here, `class` is the attribute *name* and `editor-note` is the attribute *value*. The class attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same class value) with style information and other things.

An attribute should always have the following:

1. A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
2. The attribute name followed by an equal sign.
3. The attribute value wrapped by opening and closing quotation marks.

3.4 NESTING ELEMENTS :

You can put elements inside other elements too — this is called **nesting**. If we wanted to state that our cat is **very** grumpy, we could wrap the word "very" in a element, which means that the word is to be strongly emphasized:

```
<p>My          cat          is          <strong>very</strong>          grumpy.</p>
```

Copy to Clipboard

You do however need to make sure that your elements are properly nested. In the example above, we opened the <p> element first, then the element; therefore, we have to close the element first, then the <p> element. The following is incorrect:

```
<p>My cat is <strong>very grumpy.</p></strong>
```

The elements have to open and close correctly so that they are clearly inside or outside one another. If they overlap as shown above, then your web browser will try to make the best guess at what you were trying to say, which can lead to unexpected results. So don't do it!

3.5 EMPTY ELEMENTS :

Some elements have no content and are called **empty elements**. Take the `<img>` element that we already have in our HTML page:

```

```

This contains two attributes, but there is no closing `</img>` tag and no inner content. This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.

3.6 ANATOMY OF AN HTML DOCUMENT:

That wraps up the basics of individual HTML elements, but they aren't handy on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Let's revisit the code we put into our index.html.

Here, we have the following:

- `<!DOCTYPE html>` — doctype. It is a required preamble. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However these days, they don't do much and are basically just needed to make sure your document behaves correctly. That's all you need to know for now.
- `<html></html>` — the `<html>` element. This element wraps all the content on the entire page and is sometimes known as the root element.
- `<head></head>` — the `<head>` element. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- `<meta charset="utf-8">` — This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this and it can help avoid some problems later on.

- `<title></title>` — the `<title>` element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.
- `<body></body>` — the `<body>` element. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

3.7 IMAGES:

As we said before, it embeds an image into our page in the position it appears. It does this via the `src` (source) attribute, which contains the path to our image file.

We have also included an `alt` (alternative) attribute. In this attribute, you specify descriptive text for users who cannot see the image, possibly because of the following reasons:

1. They are visually impaired. Users with significant visual impairments often use tools called screen readers to read out the `alt` text to them.
2. Something has gone wrong causing the image not to display. For example, try deliberately changing the path inside your `src` attribute to make it incorrect. If you save and reload the page, you should see something like this in place of the image:

The keywords for `alt` text are "descriptive text". The `alt` text you write should provide the reader with enough information to have a good idea of what the image conveys. In this example, our current text of "My test image" is no good at all. A much better alternative for our Firefox logo would be "The Firefox logo: a flaming fox surrounding the Earth."

3.8 MARKING UP TEXT

This section will cover some of the essential HTML elements you'll use for marking up the text.

3.8.1 HEADINGS

Heading elements allow you to specify that certain parts of your content are headings — or subheadings. In the same way that a book has the main title, chapter titles, and subtitles, an HTML document can too. HTML contains 6 heading levels, `<h1>` - `<h6>`, although you'll commonly only use 3 to 4 at most.

3.8.2 PARAGRAPH

As explained above, `<p>` elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content:

```
<p>This is a single paragraph</p>
```

3.8.3 LISTS

A lot of the web's content is lists and HTML has special elements for these. Marking up lists always consists of at least 2 elements. The most common list types are ordered and unordered lists:

1. **Unordered lists** are for lists where the order of the items doesn't matter, such as a shopping list. These are wrapped in a `<ul>` element.
2. **Ordered lists** are for lists where the order of the items does matter, such as a recipe. These are wrapped in an `<ol>` element.

Each item inside the lists is put inside an `<li>` (list item) element.

CHAPTER 4

CSS

4.1 What is CSS:

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

4.2 CSS- Syntax:

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –

- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.
- **Property** – A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** – Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

4.3 Selector

4.3.1 Type Selectors:

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings –

```
h1 {
    color: #36CFFF;
}
```

4.3.2 Universal Selectors:

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type –

```
* {
    color: #000000;
}
```

This rule renders the content of every element in our document in black.

4.3.3 Descendant Selectors:

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to element only when it lies inside tag.

```
ul          em {
    color: #000000;
}
```

4.3.4 Class Selectors:

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {  
  color: #000000;  
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example –

```
h1.black {  
  color: #000000;  
}
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

You can apply more than one class selectors to given element. Consider the following example –

```
<p class = "center bold">
```

This para will be styled by the classes *center* and *bold*.

4.3.5 ID Selectors

You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

```
#black {  
  color: #000000;  
}
```

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example –

```
h1#black {  
  color: #000000;  
}
```

This rule renders the content in black for only `<h1>` elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example –

```
#black h2 {  
  color: #000000;  
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

4.4 CSS- COLOUR

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

You can specify your color values in various formats. Following table lists all the possible formats –

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%) ;}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

These formats are explained in more detail in the following sections –

4.4.1 CSS Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

CHAPTER 5

BOOTSTRAP

The web app development tool kit was created by former Twitter employees Mark Otto and Jacob Thornton.

5.1 ITS RESPONSIVE GRID

No more spending hours coding your own grid—Bootstrap comes with its own grid system predefined. Now, get straight to filling your containers with content. Defining custom breakpoints for each column is a snap using their extra small, small, medium, large, and extra-large breaks. Also simply stick to the default as it might already meet the needs of your site.

5.2 ITS RESPONSIVE IMAGES

Bootstrap comes with its own code for automatically resizing images based on the current screen size. Just add the `.img-responsive` class to your images, and the predefined CSS rules take care of the rest. Let Bootstrap resize your images for you. It can even change the shape of your images with the addition of classes like `img-circle` and `img-rounded`, and that's without going back and forth between the code and your design software.

5.3 ITS COMPONENT

Bootstrap comes with a whole barrellful of components you can easily tack onto your web page, including: Navigation bars, Dropdowns, Progress bars, Thumbnails. Not only is it a breeze to add eye-catching design elements to your webpage, you'll also be able to rest assured knowing that every one of them will look great no matter the screen

size or device used to view them. That's a lot of ready-made functionality right at your fingertips. For a more complete list of addable features, check out the component documentation.

5.4 ITS JAVASCRIPT

Bootstrap also allows developers to take advantage of over a dozen custom jQuery plugins. This library gives you even more room to play with interactivity, offering up easy solutions for modal popups, transitions, image carousels, and—one of my personal favorites—a plugin called scrollpsy, which automatically updates your navigation bar as you scroll through a page.

5.5 ITS DOCUMENTATION

Simply put, Bootstrap's documentation is some of the best we've ever seen. Every piece of code is described and explained in explicit detail on their website. Explanations also include code samples for basic implementation, simplifying the process for even the most beginner of beginners. All you need to do is choose a component, copy and paste the code into your page, and tweak from there.

5.6 ITS CUSTOMIZABILITY

One of the main critiques when it comes to frameworks such as Bootstrap is their size—the weight they throw around can really slow down your application upon first load. The current version of Bootstrap's CSS file, for example, is a whopping 119 KB. While this may not seem especially large compared to image and video files, for a CSS file, that's enormous! What it allows you to do to combat this, however, is customize which functionality you want to include in your download. By simply going to

their Customize and Download page, you can check off the features you won't need for your application, trimming the weight off your file and saving your users the additional load time.

5.7 ITS COMMUNITY

As with so many open-source projects, Bootstrap has a large community of designers and developers behind it. Being hosted on GitHub makes it easy for developers to modify and contribute to Bootstrap's codebase. It also makes it easy for people to collaborate, lend their advice, and interact with peers and fellow users.

5.8 ITS EXTERNAL TEMPLATES

As its popularity grew, people started creating templates based on Bootstrap in order to accelerate the web development process even further. There are many websites out there dedicated to sharing and buying custom templates based on Bootstrap.

CHAPTER 6

RESPONSIVE WEBSITE

6.1. SET APPROPRIATE RESPONSIVE BREAKPOINTS

In responsive design, a breakpoint is the “point” at which a website’s content and design will adapt in a certain way in order to provide the best possible user experience. Every website is accessed via devices with different screen sizes and resolutions. The software has to render perfectly across each screen size. Content or images cannot be distorted, cut out, or obscured. To allow this, developers have to use responsive breakpoints, sometimes called CSS breakpoints or media query breakpoints. These are points defined in the code. Website content responds to these points and adjusts itself to the screen size to display the accurate layout.

6.2. START WITH A FLUID GRID

Websites were based on pixel measurements. Now, however, they are built on what is called a fluid grid. Basically, a fluid grid positions and sets web elements on a site in proportion to the screen size it is displayed on. Instead of making things in a single, specific size set in pixels, elements on a fluid grid will respond and resize to fit the size of the screen. A fluid grid is divided into columns; heights and widths are scaled, not set to fixed dimensions. The proportions of text and elements depend on the screen size. Formulating the rules of a fluid grid is done by working on the website’s source code. A fluid grid also helps with keeping a site visually consistent across multiple devices. It also offers closer control over alignments and enables faster design-related decision-making.

6.3 TAKE TOUCHSCREENS INTO CONSIDERATION

When wondering how to make a website responsive, think of touchscreens. Most mobile devices (phones and tablets) are now equipped with touchscreens. Some laptops are also catching up, offering touchscreen along with the keyboard functions. Naturally, a responsive website will have to calibrate itself for being accessed via touchscreens. For example, let's say there is a drop-down menu on the homepage. On desktop view, each menu item must be large enough so that it can be pressed with a fingertip of a touchscreen. On mobile screens, smaller elements like buttons should also be easier to detect and select. To do so, use images, CTAs or optimize these elements to render properly on multiple screens.

6.4. DEFINE TYPOGRAPHY

Generally, web developers define font sizes with pixels. These work on static websites, but responsive websites need a responsive font. Font size must change with respect to parent container width. This is necessary to make typography adjust to screen size and be easily readable on multiple devices.

6.5. USE A PRE-DESIGNED THEME OR LAYOUT TO SAVE TIME

If developers and designers are wondering how to build a responsive website on an exceptionally tight deadline, they can opt for using a theme or pre-designed layout with built-in responsive properties. wordpress provides multiple options in this regard (both free and paid). All designers have to do, after picking a theme, is to decide on color, branding, and text.

6.6. TEST RESPONSIVENESS ON REAL DEVICES

When researching how to make a website mobile responsive, something that often gets overlooked is the necessity of testing on real devices. Developers can tweak the code all they want, but its functionality has to be verified in real user conditions

CHAPTER 7

ANGULAR-FRONT END APPLICATION

7.1 SERVER (BACK-END) APPLICATION

Perform the steps described in one of the following topics to create a backend application:

Client (Front-End) Application

Perform the following steps to create and configure an Angular client-side application

Create New Application

1. Node.js and npm if they do not exist on your machine. Check whether the latest version of **npm** is installed.
2. Open the command prompt and install the Angular CLI (Command line interface tool) globally.

console

```
npm install -g @angular/cli
```

3. Select a folder to store all the files related to the client-side functionality. Navigate to that folder and run the following command:

console

```
ng new angular-report-designer --strict=false
```

This command creates the angular-report-designer folder that contains a new Angular application. Refer to the Angular documentation for information on the application structure.

Install Libraries

1. Open the **package.json** configuration file in the **Angular application** and add the following packages:
2. ...
3. "dependencies": {
4. ...
5. "devextreme": "21.2.7",
6. "devexpress-richedit": "21.2.7",
7. "@devexpress/analytics-core": "21.2.7",
8. "devexpress-reporting-angular": "21.2.7"
9. },

7.2. Create the Angular Component

Replace the **\src\app\app.module.ts** file content with the following code:

```
import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';
import { DxReportDesignerModule } from 'devexpress-reporting-angular';
import { AppComponent } from './app.component';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    DxReportDesignerModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Replace the `\src\app\app.component.html` file content with the following code that uses the dx-report-designer Angular component

```
<div>
<dx-report-designer [reportUrl]="reportName" height="700px" >
  <dxrd-request-options
[getDesignerModelAction]="getDesignerModelAction"    [host]="host"></dxrd-
request-options>
  </dx-report-designer>
</div>
```

Replace the `\src\app\app.component.ts` file content with the following code:

```
import { Component, ViewEncapsulation } from '@angular/core';
import 'devexpress-reporting/dx-richedit';

@Component({
  selector: 'app-root',
  encapsulation: ViewEncapsulation.None,
  templateUrl: './app.component.html',
  styleUrls: [
    "../../node_modules/jquery-ui/themes/base/all.css",
    "../../node_modules/devextreme/dist/css/dx.light.css",
    "../../node_modules/devexpress-richedit/dist/dx.richedit.css",
    "../../node_modules/@devexpress/analytics-core/dist/css/dx-
analytics.common.css",
    "../../node_modules/@devexpress/analytics-core/dist/css/dx-
analytics.light.css",
    "../../node_modules/@devexpress/analytics-core/dist/css/dx-
querybuilder.css",
    "../../node_modules/devexpress-reporting/dist/css/dx-
webdocumentviewer.css",
    "../../node_modules/devexpress-reporting/dist/css/dx-
reportdesigner.css" ] })
export class AppComponent {
  title = 'DXReportDesignerSample';
  getDesignerModelAction = "/ReportDesigner/GetReportDesignerModel";
  // The report name.
  reportName = "TestReport";
  // The backend application URL.
  host = 'https://localhost:44389/' ;}
```

CHAPTER 8

FIREBASE HOSTING

Firebase Hosting gives you a fast, secure, and reliable way to host your app's static assets (HTML, CSS, JavaScript, media files, etc.) as well as to serve dynamic content and host microservices.

Our production-grade hosting is backed by a global content delivery network (CDN). Hosting serves your content over SSL, by default, and can be used with your own custom domain or on your project's subdomains at no cost on web.app and firebaseapp.com.

Step 1: Install the Firebase CLI

Visit the [Firebase CLI documentation](#) to learn how to install the CLI or update to its latest version.

Step 2: Initialize your project

To connect your local project files to your Firebase project, run the following command from the root of your local project directory:

```
firebase init hosting
```

During project initialization, from the Firebase CLI prompts

1. Select a Firebase project to connect to your local project directory.

The selected Firebase project is your "default" Firebase project for your local project directory. To connect additional Firebase projects to your local project directory, set up project aliases.

2. Specify a directory to use as your public root directory.

This directory contains all your publicly served static files, including your index.html file and any other assets that you want to deploy to Firebase Hosting. The default for the public root directory is called public.

- ☐ You can specify your public root directory now or you can specify it later in your firebase.json configuration file.
- ☐ If you select the default and don't already have a directory called public, Firebase creates it for you.
- ☒ If you don't already have a valid index.html file or 404.html file in your public root directory, Firebase creates them for you.

2. Choose a configuration for your site.

- ☐ If you select to make a one-page app, then Firebase automatically adds rewrite configurations for you.
 - ☐ At the end of initialization, Firebase automatically creates and adds two files to the root of your local app directory:
 3. A firebase.json configuration file that lists your project configuration. Learn more about this file on the [configure hosting behavior page](#).
 4. A .firebaserc file that stores your project aliases
- ☐ **Step 3: Deploy to your site**
 - ☐ To deploy to your site, run the following command from the root of your local project directory:
 - ☐ `firebase deploy --only hosting`

- ☐ **Note:** By running this command with the **--only hosting** flag, you're only deploying your Hosting content and config. If you *also* want to deploy other project resources or configurations (like functions or database rules), run this command with a comma-separated list in the flag (for example, **--only hosting, functions**).
- ☐ This command deploys your Hosting content and config to the following Firebase-provisioned subdomains:

5. *PROJECT_ID*.web.app

6. *PROJECT_ID*.firebaseapp.com

- ☐ Learn more about deploys and even locally testing your site
- ☐ **Next steps**
- ☐ Now your site is ready to share with the world!

7. Continue to improve your site. Test locally, share changes at a temporary preview URL, then deploy to your live site. Follow this step-by-step guide

8. Learn about further hosting capabilities:

- ☐ Configure hosting behavior
- ☐ Connect a custom domain
- ☐ Serve dynamic content and host microservices

9. Take a look at the full documentation for the Firebase CLI

10. Prepare to launch your app:

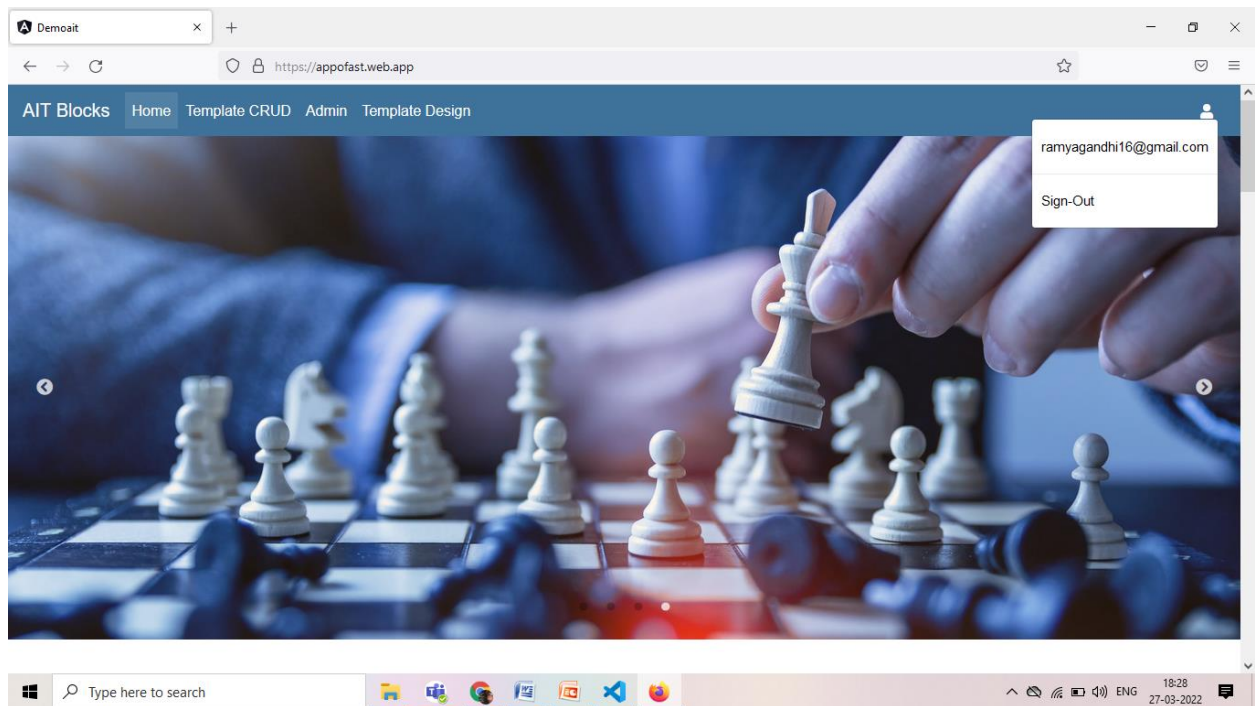
- ☐ Set up budget alerts for your project in the Google Cloud Console.
- ☐ Monitor the usage and billing dashboard in the Firebase console to get an overall picture of your project's usage across multiple Firebase services. You can also visit the Hosting dashboard for more detailed usage information.

Review the Firebase launch checklist

CHAPTER 9

RESULTS

Front page-UI



Add user in Admin module

The screenshot shows a web application interface for managing users. The browser address bar indicates the URL is `https://appofast.web.app/admin`. The navigation bar includes links for 'AIT Blocks', 'Home', 'Template CRUD', 'Admin', and 'Template Design'. The main heading is 'Employee CRUD'. Below this, there are two buttons: '+ADD USER' and 'DELETE'. The form for adding a user consists of two input fields: 'Enter name' and 'Enter role', followed by a red 'SAVE' button. Below the form is a 'Filter' input field. A table displays the current list of users with columns 'Name' and 'Role'. The table contains three entries: 'devi' (employee), 'dev' (sadmin), and 'ramyalakshmi446' (sadmin). At the bottom of the table, there is a pagination control showing 'Items per page: 3' and '1 - 3 of 5', along with navigation arrows. The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray information including the time '18:28' and date '27-03-2022'.

Employee CRUD

+ADD USER DELETE

Enter name

Enter role

SAVE

Filter

Name	Role
devi	employee
dev	sadmin
ramyalakshmi446	sadmin

Items per page: 3 1 - 3 of 5

Add template in Admin module

The screenshot shows a web application interface for managing templates. At the top, there is a navigation bar with the following links: AIT Blocks, Home, Template CRUD, Admin, and Template Design. The 'Admin' link is currently selected. Below the navigation bar, there are two buttons: '+ADD NEW TEMPLATE' and 'DELETE'. A pink camera icon is positioned below the '+ADD NEW TEMPLATE' button. The form for adding a new template consists of three input fields: 'Type eg. contact, team..', 'Name eg. c1', and 'Price'. Below these fields is a red 'SAVE' button. A 'Filter' input field is located below the 'SAVE' button. At the bottom of the form, there is a table with the following columns: Category, Name, and Price. The table contains one row with the following data: ecom, ec2, and 33.

AIT Blocks Home Template CRUD Admin Template Design

+ADD NEW TEMPLATE DELETE

Type eg. contact, team..

Name eg. c1

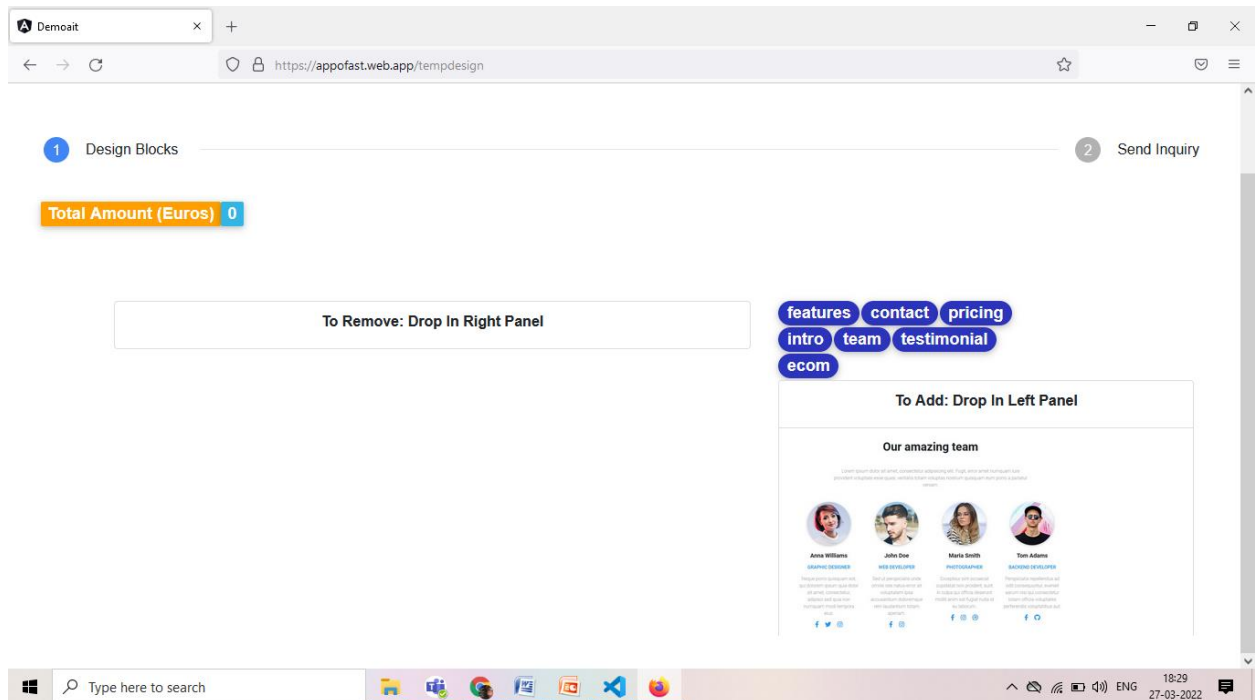
Price

SAVE

Filter

Category	Name	Price
ecom	ec2	33

Select template in user module



Contact Us in User Module

The screenshot displays a web browser window with a single tab titled 'Demoait'. The address bar shows the URL 'https://appofast.web.app/tempdesign'. The page features a progress indicator at the top with two steps: '1 Design Blocks' and '2 Send Inquiry', with the second step being the active one. The main content area is titled 'Contact us' and contains a form with three input fields: 'Name', 'E-mail', and 'Phone'. Below these fields is a 'SEND' button. A 'BACK' button is located at the bottom left of the form area. The Windows taskbar at the bottom shows the search bar with the text 'Type here to search', several application icons, and system status icons on the right, including the date '27-03-2022' and time '18:30'.

1 Design Blocks 2 Send Inquiry

Contact us

Name

E-mail

Phone

SEND

BACK

Type here to search

18:30
27-03-2022

CHAPTER 10

SUMMARY AND CONCLUSION

Website templates are main attributes of a website. It is like a form of letter that is designed to justify separation of presentation and substance of the website. The main function of template is to generate and produce web pages at a fast speed. The algorithm and the programming language of a website template remain pre-defined and work schematically once given instructions.

It is a wise to choose good and effective template of a websites. A good-looking website template is eye catching and it shows the creativity of the web developer, designer or web master. Starting up a website requires both time and money. Creative web developers always try to create and deploy applications in templates which is flexible and easy to maintain so that users of the websites can work freely.

The basic knowledge of internet and web developing is enough. It is rather easy to create a website these days because there are thousands of web templates available. One needs to select the best one from the list and use it.

REFERENCES

- Arslan,R.S.,&Sachin-Kizil,A(2010). How can the use of blog software facilitate the writing process of English language learners. Computers Assisted Learning
- Blackstone,B.Spiri,J.,Naganuma (2007), Blogs in English Language teaching and learning, Pedagogical uses and student responses. Reflections on English Language Teaching.
- Blood,R(2004), How blogging software reshapes the online community. Communication of the ACM,47(12)53-55.
- Churchill,D(2011), Web 2.0, in education; A study of the explorative use of blogs with a post graduate class. Innovation in education and teaching international.
- Cohen,K.R(2006). A welcome for blogs,Continuum: Journal of media & cultural studies.
- Ducate, L.C,&Lomica L., (2008) Adventures in the blogosphere:From blog readers to blog writers Computer Assisted Language Learning.
- Kaye,B.K (2004). Web site story: An explanatory study of why weblog users say they use weblogs.