Dynamic Color Theming and Relative Color Predicting in HTML Documents

H.M.W.S.K.N. Herath

2018

Dynamic Color Theming and Relative Color Predicting in HTML Documents

A dissertation submitted for the Degree of Master of Computer Science

H.M.W.S.K.N. Herath

University of Colombo School of Computing

2018

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: H.M.W.S.K.N. HERATH

Registration Number: 2015/MCS/034

Index Number: 15440349

_____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Prof. K.P Hewagamage.

_____

Signature: Date:

Abstract

Colors add an extraordinary value to anything. Colors of the HTML user interfaces give them a fascinating attraction for users. But today HTML pages are written in a static way which the colors are embedded into the code. This property makes it very hard to change once the design is finalized. We have to define many CSS pages as themes, since we need web sites in many color ranges.

We purpose a solution to this issue by letting the colors of the HTML pages to be dynamic even after the design is finalized. We purpose a framework which is called “Chameleon JS” Users of the framework need to integrate it to their HTML documents. Then they can change the colors of a particular HTML document. We can choose any color form the planet for the web document.

Framework is built by experimenting the properties of color and the way they are presented in the computer screen. We purpose a way to define a color relative to another color. This relativity property is used to predict a color which is relative to the initially defined color of the HTML document. This way we keep the initial color combination of the HTML page and change the colors relatively to another color. Purposed framework is written in native CSS and JavaScript with HTML 5 standard. Hence, we assume any web technology can adapt this framework and use with their respective applications. RGB and HSL color schemes are considered to find a method to define colors relatively. HSL color scheme is selected with its unique features of dividing color into several aspects. Hue, Saturation and Lightness are compared when predicting a new color.

This work achieves the state of dynamic color theming for the HTML documents using HSL color scheme. Furthermore, the framework is able to keep the color consistency and predict new colors relative to a previously defined color while keeping the initial color scheme of the original HTML document. CSS variables are used to apply the relevant colors into the HTML document. With CSS variables colors which are predicted can be applied right away to the HTML document. Dynamic color theming and relative color predicting framework is tested against many technologies and succeeded over them. A website called “chameleonjs.xyz” is created for the convenience of the users. Download bundles are available in this website. Design, implementation and the evaluation of this work is included in this report.

Acknowledgement

First and foremost, I owe my deepest gratitude to my supervisor, Prof K.P. Hewagamage for his valuable guidance and advice which continually and convincingly conveyed a spirit of adventure in regard to the research. This project would not have been possible without him because he has always shown the way of research and helped me to overcome most of the obstacles that I had encountered.

I offer my sincere gratitude to our final year project coordinator for his great commitment and encouragement given to maintain the flow of the project from beginning to end. And make this opportunity to thank all the lecturers at the University of Colombo School of Computing, for their valuable advice.

I would also thank my parents and my friends who encouraged me to complete this research. Finally, I thank all of my colleagues, who supported me in many ways during the completion of the research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

- **CSS** - Cascading Style Sheets
- **CMYK** - Cyan Magenta Yellow Key
- **DOM** - Document Object Model
- **HTML** - Hyper Text Markup Language
- **HSL** - Hue Saturation Lightness.
- **ICT** - Information Communication Technology
- **RGB** - Red Green Blue
- **UCSC** - University of Colombo School of Computing
- **UI** - User Interface
- **URL** - Universal Resource Locator
- **W3C** - World Wide Web Consortium
- **XML** - Extensible Markup Language

Chapter 1

Introduction

1.1 Motivation

HTML is the publishing language in World Wide Web [1]. Structure of the HTML page is consisting with HTML elements, and elements are represented by tags. Browsers use these tags to render the content of the page. User interface of the HTML pages are designed with these HTML tags and CSS is used to style them for a better user experience. Dynamic user interface design is more challenging than a static traditional HTML structure. Designing and styling these user interfaces faces many complications when changing colors of the html elements dynamically. Creating a dynamic view which can give a dynamic look and feel for the user is much better than having a static design.

Fundamental standard of W3C to style HTML pages is Cascading Style Sheets (CSS) [1]. Today industry standard frameworks like CSS are not directly capable of handling complex dynamic color theming of HTML documents [2].

This research is focused on defining CSS on HTML pages which will give dynamic color changing facility after declaring the CSS on the document. Colors of sections of the page will be able to control separately and will be changed relative to the other colors defined.

As internet and interconnectivity of the world increases web technologies has become most extremely popular and useful in information technology sector. Good user interface of HTML document will play an energetic role in any web application. This can be the key point for a better human computer interaction. This is why most of software vendors' main focus is to have a better user interface, as it is the main dealing medium with the end users of their valuable applications. Using CSS web applications and mobile applications can be styled

easily. These user interfaces should be user-friendly and attractive to gain users attraction in the competitive web industry.

Theme of a user interface is created at the design time of the html page. Designers work hard to create the theme of the page without color conflictions. They decide the colors which are matching with other used colors and build up the theme carefully. Once this theme is written in CSS language it is hard to change it again. Because HTML tags are usually, tightly bind to the CSS which is defined at the design time. If we try to change the colors after that, many conflictions and mismatching situations can occur.

Having a technical situation like that will not stop the requirements of the real-world applications and it will only reduce the quality of the designed application. It is required to do dynamic color changes when we have the users who needs to change the colors of the interface eventually or when applications need to change the color for a better look and feel.

Today over 90% of web developers are using CSS [5] in 90% of the web sites and over 93% of external CSS sheets are used in them [6]. Most of these developers are facing dynamic color changing problems and it is easier if they can have a way to handle the complexity of the dynamic user interface color changing, color grouping and color conflicts. It will address a major industrial problem. It is an extra cost and will be time consuming when there is a requirement to change the colors in the user interfaces even after the development. Giving user to change the colors of the application they use is hard to give as a feature, once the development is done only for a specific color range. Well defined grouping mechanism as a framework used in development time might be able to handle this complexity.

As user interfaces are very important in any web site or an application it is always better to have a dynamic theming feature. This research states a mechanism to implement dynamic color theming in HTML documents. This will introduce an extended framework of CSS and set of rules to govern the framework in a way that dynamic theming implementation will not have color conflictions or reduce color conflictions to an acceptable/manageable level.

1.2 Problem Statement

How to define and structure CSS in HTML documents in a way that it will give the ability to change the colors of the document in a dynamic manner without conflictions.

1.3 Technological Background

Cascading style sheets are extensively used to style HTML documents. It is a simple styling language which has the ability to divide the presentation of document from its structure. Properties of HTML elements can be decorated with CSS. It describes how HTML elements are displayed in browsers and other documents. Position, size, colors and other attributes of HTML elements are controlled by CSS and it can save much work as it has characteristics like inheritance which helps multiple documents to be styled at once. Any HTML document can be styled by using CSS [1].

CSS can be applied to HTML elements in several ways.

I. Inline – Defining style on the HTML element.
II. Internal – Defining style on the header section of the HTML document.
III. External – Defining a separate CSS document and linking the document in to HTML page.

CSS properties inherit from one to another, as an example; the closest/top most definition specified for an element will be the one which will override other definitions which were written for the same element and apply to the respective element. More than 93% of the internet uses external CSS. [6]. These CSS sheets define colors of the HTML elements which will directly effect on the look and feel of the web document. This work addresses this property of the Cascading Style Sheets to dynamically change the colors of the HTML page. Predicted outputs for the HTML document will by changed on the fly with less overhead for the HTML DOM element.

1.3.1 Implementation of dynamic color theming with CSS

As we are going to build a new feature for dynamic color theming we should consider the current status and implementation of CSS in the HTML documents.

CSS is the way defined by W3C to style HTML documents [1]. It is a framework with many features which are very useful for styling HTML pages. These HTML tags builds the structure and creates as Document Object Model of the HTML page which is rendered to the browser by JavaScript.

The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. [2]

This dynamic nature of the DOM gives the ability to change the HTML elements dynamically. We can change the properties of the HTML elements even after the HTML DOM is created. We can do it by calling the DOM and changing the properties of the HTML element which is created on the DOM.

Technologies today like AngularJS, Angular2, React, jQuery, and Bootstrap are most popular frameworks in frontend designing which can directly access the DOM by using JavaScript. With these frameworks or even with native JavaScript we can implement dynamic color theming for the applications or HTML documents.

These solutions cannot be used as a standard way because one technical solution given by one framework will not directly implementable in other framework due to technical dependencies. Also, these technical stack dependent frameworks only allow to implement solutions in a programmatic way, at the end every technology will call the DOM by using pure JavaScript and change the colors of the HTML elements. Hence, we cannot use such simple solutions as a standard way in every development project. If we can implement a solution which will directly work from CSS and native JavaScript with conflict less design, it can be used in any project without depending on the technology. This research is trying to design something which will address these problems.

1.4 Goal and Objectives

Goal of this research is to find a method to dynamically change the colors of the HTML pages. These colors should be relevant and matching to the initial colors of the HTML page. This will keep the consistency of page design. To achieve the goal there are many objectives in this work.

1. Research for a methodology to define one color relative to another color.

   There we need to find be a mechanism to define a one color relative to another color. The purpose of this is to get the color of one HTML element relative to another element. With this color relativity mechanism, we can define the initial color matrix of the HTML page. Once this matrix is defined we can protect the initial color matrix and create another color matrix which is matching to the initial colors. New color matrix can be used as the new theme of the HTML page.

2. Find a suitable and effective method to change the color properties of the HTML documents on the fly.

   The colors which are predicted by the framework should apply to the HTML page instantly. This will change the colors of the HTML page on the fly. This is the way purposed to get the dynamic color changing capability to the HTML pages. As of the background search HTML 5 defines a mechanism to change the colors of HTML elements with CSS variables. There are many other ways to define colors of the HTML page such as using jQuery or native JavaScript. We need to find a suitable method to dynamically change the colors of the HTML pages.

3. Research for a methodology to predict new colors relative to another color.

   When users need to change the color of the HTML page to another color. There should be a mechanism to implement the required work. The purpose of this objective is implementing the work using color relativity and dynamic color changing

mechanism of HTML pages, we are aiming to find. This will be designed as an algorithm so any input/output to the algorithm will be handled properly.

4. Design a framework for users to use this work in their HTML pages.

   It is highly aimed this solution to use by developers and end users. Hence there should be a suitable mechanism and set of instructions for uses. This will be achieved by designing this solution as a framework. Developers should implement this framework for their HTML pages for users to use. Framework is aimed to design as easy as possible for developers to use, hence many of the aspects such as color definition methods will be automated for easiness of users.

5. Possible color conflicts that can occur with the framework should be minimized or prevented.

   There can be many color conflicts that can occur when applying the new colors to the existing HTML page. These color conflicts should be minimized as possible or prevented by using a suitable mechanism. A rule set to prevent color conflicts or other mechanism should be implemented on the framework. Ideally any color which was not conflicting in initial design should not be conflicted after the color change.

6. Design the framework independent of other technologies, so any standard website can use this solution.

   It is aimed to design this framework without using cross technologies so the final framework will not be depending on any of the other external technologies or frameworks other than CSS and JavaScript which is supported by all standard browsers as of March 2018.

1.5 Scope

Scope of the research work is important as it should clearly describe the boundaries of the work to be achieved. Scope of work of this research work is as below.

- Scope of research for a methodology to define one color relative to another color. Colors can be defined with many mechanisms. But here our scope is to find a mechanism which is suitable for our goal of the research. It is purposed to compare RGB and HSL color schemes and understand the color theory to implement a mechanism. Main aim of this is to calculate the difference between colors and the scope of this objective limits to that.

- Scope of finding a suitable and effective method to change the color properties of the HTML documents on the fly. In the technological world there are many mechanisms to change the colors on single HTML element on the fly. Scope of this objective is to test and find the best suitable mechanism for this work. A website may consist of many web pages. But scope of this work is limits to a single.

- Scope of researching for a methodology to predict new colors relative to another color. When one color of the HTML element is changing to another color, some attributes of the color may change. As an example, hue and saturation may change but lightness may not change. Scope of this objective is to find a good mechanism suitable for our work. The inputs of this prediction are limit to single color and other colors will be predicted relative to this color.

- Scope of the purposed framework. Frameworks scope is limited to implementing CSS files and JavaScript files. Set of instructions will be given for the users to use this solution in their THML documents.

- Possible color conflicts that can occur with the framework should be minimized or prevented. Scope of the color conflict prevention is limited only to the aspects of visibility and look and feel. It is purposed to achieve a good level of prevention but the scope of it is limited to prevention and not to be stopped completely. Color conflict prevention mechanism limits only to handle manually and not fully automated.

- Design the framework independent of other technologies, so any standard website can use this solution. Scope of this objective is to be stick into HTML 5 standard. Standard

HTML 5 features and JavaScript ES2016 standard [20] functions will be used by the framework. This solution is only compatible with the web browsers which supports CSS variables.

- Scope of this project is limited to address the discussed problem domain and it will not address the technical dependent problems that can occur when using the purposed framework.

- Scope is limited to control the colors of the html document and it will not address the dynamic object creation or HTML component creation. Solution will address the color changes of the document when it is applied correctly.

- Inline CSS are not allowed in the html documents due to the complexity. Solution will only accept for style sheets.

- HTML color property rgba(x,x,x,y) usage will not be feasible with the current framework. The scope is limited only to define colors with hexadecimal codes. Defining colors with word in HTML document may take into consideration but scope is limited only to hexadecimal codes.

1.6 Thesis Outline

- Chapter 2 contains the literature review of this project. It will describe the similar research efforts, technological background and how this research work deviates from them will be described.

- Chapter 3 describes the design of the purposed solution and the design principals of the work. The design and assumptions.

- Chapter 4 presents the implementation details of the proposed model and It describes system architectural level implementation as well as infrastructural level details.

- Chapter 5 discusses the evaluation methods and evaluation results of this research.

- Chapter 6 conclusion and the possible future enhancements is proposed.

Chapter 2

Background and Related Work

Cascading style sheets are an area which is not researched much. “Style sheet languages are terribly under-researched” [10] [11]. World Wide Web Consortium (W3C) has standardized the CSS language, its selectors and other specifications. CSS design principles [12] are defined by this organization. Current version for CSS on the time of this writing document is version 4 which is released on 24th March 2017. This versioning is also handled by World Wide Web Consortium.

2.1 Constraint Cascading Style Sheets for the Web

Constraint Cascading Style Sheets for the Web [13] is a work sometime back from today. But still these are the rules which are subjected to identify CSS conflicting rules. Our work is going to identify rules which can defend the CSS color conflictions which makes us to work on the generalized set of rules to apply for CSS.

This study demonstrates how constraints provide a powerful unifying formalism for declaratively understanding and specifying style sheets for web documents. [13]

This study is not directly a framework but an aim to reflect the way of braking down structure when constraints are added to the current CSS framework. Our work is breaking the structure of HTML DOM elements and trying to create a mechanism to implement dynamic changes. Hence this work relates to our work in that way. Identifying a set of rules which will reduce the color conflictions within the HTML elements will lead to generalize them before applying to the framework.

2.2 Automated Analysis of CSS Rules to Support Style Maintenance

Furthermore, style sheets of a web document should be easily maintainable. When the dynamic color change feature is implemented on users' CSS documents, they should be easily maintainable. Otherwise little portion of our problem domain will not be addressed and remain unsolved. Work done by Ali Mesbah and Shabnam Mirshokraie which is Automated Analysis of CSS Rules to Support Style Maintenance [14] is addressing the issue of maintaining Cascading Style Sheets. As of them 60% unused CSS selectors in deployed applications. [14]

When unused style rules and style definitions are validated in the user's browsers it leads to the heavy duty which can be avoided and optimized.

They have used few algorithms which can analyze the HTML DOM elements and identify style is matched with DOM element or not. Work demonstrates the ability for a browser to avoid unnecessary style overhead. Usually this unnecessary overhead is avoidable. Our work which is going to catch the HTML DOM elements and identify the color conflicting HTML elements will also use an algorithm which is close to this work. This should be able to go through all current HTML DOM elements of the document and validate the colors over the researched and identified rules.

W3 Web Consortium has defined new variables and operators as a standard and this research will use these new features of CSS defined by the w3 web consortium. [14] As we have mentioned in the introduction chapter these new variables which can be defined in the Cascading Style Sheets are able to control the property values of the CSS definitions.

While we can control the variables of HTML element properties we should be able to create a mechanism to define CSS for dynamic features.

By observation the current problem it can be addressed by creating a framework which will force the users to design the color combinations of html page at the development time.

They will first initialize the colors of the page. Once the page is finished we can calculate the color relativity of the html elements / divisions and it will be used as the parameters to do the dynamic color theming. Idea is not to destroy the color pattern of the page which is defined by the user (developer / designer) and just to make changes according same pattern/style but to give a different look and feel.

Proposed solution is not a straightforward technical development. New CSS4 and HTML5 standards can be used to implement this. Also, JavaScript DOM methods can be used to control the behavior of the framework. Framework will give CSS classes / variables for users to use. It will be the control structure of the html page of user by the framework.

Color relativity of the elements should be calculated and I will have to practically find a way to define color relativity. Initial idea is to use the RGB color codes and create a schema or a pattern in a way that it will represent the relativity of two colors.

With a background where CSS is not researched much for dynamic color theming here we are trying to find a solution for that. Dynamic color predicting will be tightly bind to this solution as colors of the html elements should not conflicted with each other.

2.3 Color Relativity Work form Adobe CC

Adobe CC have done a color wheel to generate many colors. [16] They have color rules called Analogous, Monochromic, Triad, Complementary, Compound, Shades and Custom. These rules change the way the colors are mixed in the color wheel to generate a new color. Color relativity mechanism they have used with rule of complementary is giving a similar experience as from our work. In that what they are trying to achieve is creating a new color and a good experience for users to pick the colors. Our work is relating to change the colors of HTML elements and our algorithm is predicting a color to an element according to a given color, hence this is purely not our work but similar work which focuses on giving a good user experience for the users who are looking for colors.

2.4 W3C Standard for Colors in HTML documents

CSS style sheets are used to style the HTML pages. [12] Color of them is defined by using many standard methods described in CSS standard by W3C organization. [21] Following are methods to define a color to a HTML element.

```
em { color: lime }                  /* color keyword */
em { color: rgb(0,255,0) }          /* RGB range 0-255   */
body {color: black; background: white }
h1 { color: maroon }
h2 { color: olive }
em { color: #f00 }                  /* #rgb */
em { color: #ff0000 }               /* #rrggbb */
em { color: rgb(255,0,0) }
em { color: rgb(100%, 0%, 0%) }
em { color: hsl(120, 100%, 25%) } /* dark green */
```

Figure 2.1: Methods to define color in CSS

Users can use any of these methods depending on the situation on their HTML documents. In our work we can consider these methods when we create the algorithm. Usual format of the color definition in hexadecimal format followed by a “#” symbol.

2.4.1 RGB and HSL color schemes in HTML

As of the W3C standard RGB and HSL color schemes can be used in HTML elements. The properties of the color are divided into several aspects in these two color schemes.

2.4.1.1 RGB Color Scheme

Color is created by using the three main elements which are red, green and blue. Any color can be made by combining these three colors. They are the primary colors from the color wheel.

In HTML red color can be defined from 0 to 255. Green and blue colors are also ranged from 0 to 255. These numbers can be mixed together to implement the final color for the HTML element. Following is the way it is being done in the standard HTML elements.

```
h1 {
   color : rgb(255,0,0)
}
```

Figure 2.2: RGB definition in HTML page

Since we have 256 different colors in each primary color we can produce 16 581 375 colors theoretically. Roughly we can produce 16 million different colors with this 8-bit RGB color scheme.

2.4.1.2 HSL Color Scheme

Three major factors of the colors can be described as hue, saturation and lightness. This is taken into consideration by W3C and HSL color scheme is define as another standard way to define colors as mentioned above.

Hue defines the color variance form the color wheel saturation defines the color density and lightness defines the shade from black to white. Following figure [22] from the "nixsensor.com" describes this connection very clearly.

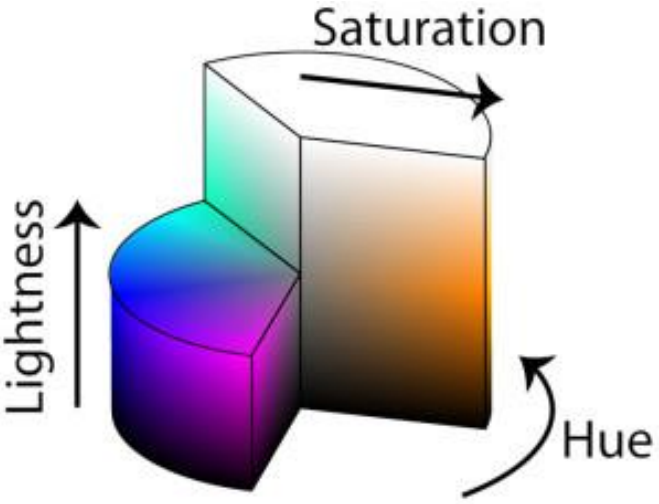


Figure 2.3: Hue, Saturation and Lightness

HTML documents define this property in the following way. As we can see we mention the hue value from 0 to 360 as degrees, saturation and lightness values from 0% to 100%. Below example outputs the dark green color by using HSL properties.

```
h1 {
color: hsl(120, 100%, 25%)  /* dark green */
}
```

Figure 2.4: HSL Color definition in HTML pages

2.5 Color theory

Beauty comes from the bright match of right colors. Color theory explains the mixing of colors and the way they are interacting with each other. In visual arts color theory plays a vital role to impress the audience of the art. Color mixing highly depends with the right color matching. There are primary colors, secondary colors and tertiary colors. Mix of them can make any color from the color wheel. With respect to primary colors there are three color system as follows.

- RGB – Red, Green, Blue

  RGB color wheel uses three light sources and mix them as needed. This is used in computer and television screens.

- CMY – Cyan, Magenta, Yellow

  CMY color scheme is sometimes referred as CMYK where K stands for black. This color scheme is mostly used to represent colors in the papers. Most modern printers use this color scheme to generate the colors. Here K is considered externally since true black is not possible with CMY colors alone.

- RYB – Red, Yellow, Blue

  RYB color scheme is used by painters and artists. This is the most natural color pattern and hence it makes more natural feeling with white background.

Each and every system is able to implement the full color wheel by mixing the three colors it contains. There are many predefined combinations of colors where monochromatic color schemes, complementary color schemes and analogous color schemes.

Color can be divided into several parts with its properties. Following are the few main properties which are important when considering about color.

2.5.1 Hue

Hue is the mixture of two colors like red and green, it is also the pure color within the color properties. Hue ranges from 0° to 359° degrees in the color wheel. Within the color space hue can be considers as in the set of pure colors. Hue can tropically be represented as a number. Hence the measurement of hue is quantitative.

2.5.2 Saturation

Saturation is the range from pure color to gray (or the lightness). This can also be described as purity of the hue. High saturation makes the colors bright and low saturation makes the color

washed out or grayish. Saturation also defines the brilliance and intensity of color. Three main properties can be considered with saturation, they are tint, tone and shade, figure 2.5 explains this concept in a straightforward way. This figure speaks more than 1000 words described about the saturation.

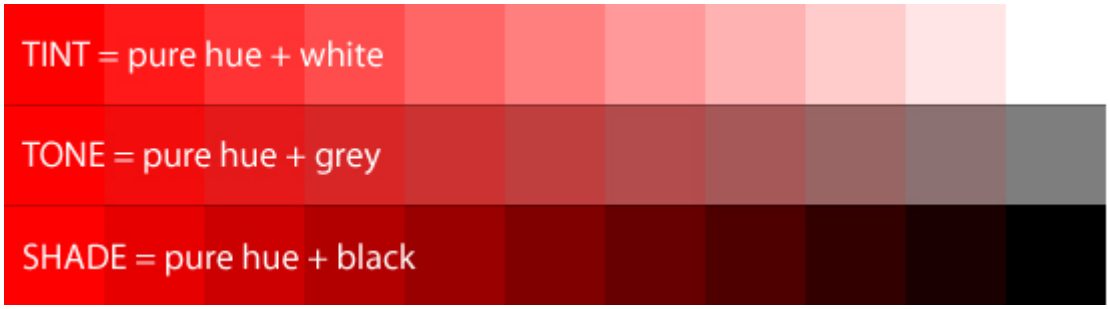


Figure 2.5: Different saturations with white gray and black

As we can clearly see when white is added to the pure hue of the white, verity colors can be generated. Only specific values are mentioned here but there are millions of colors within this range. This property is called tint.

When mixing the gray or the washed-out color to the pure red there another variance that is created, this property is called tone of the color. Pure color is being washed out at the edge of the color pattern. Grayscale pictures can be generated with this property.

When mixing pure black the last set of colors are generated. This property is called shade. At the edge of the color pattern it is purely black, but what we actually have is absence of color. Many color patterns can be created with mixing the black.

2.5.3 Lightness

Lightness is the range from dark to illumination. When the color's lightness is set to 0% it is called dark and only black color can be seen, when it I set to 100% it is too light and only white color can be seen. 50% is the normal value and there is no disturbance to the pure color by that value.

2.5.4 Intensity (Luma)

Number of photons a light source eliminates is called intensity or luma. There are many meanings for this in context of physicians and painters. On the other hand, intensity can be defined as the amount of purity in the hue itself. This property can be seen when gray is mixed to the color. That would make the color dull or washed-out. When more gray is added the color can be seen as low intensive.

2.6 Color and the Human Eye

Color is essential for human perception, even animals can identify colors. Good colors can bring more value to the human life while some objects become extraordinary for many situations with the right colors. Below is a good example of using shades of some colors in photography.

"With the human brain able to distinguish over two hundred shades of white, able to see the same color no matter the light source, saying color is essential to our perception is no slight exaggeration. Viewing a black and white scenic full of all the shades of gray that a good paper and photographer can bring to light, the emotions just those shades of gray can evoke is tremendous!" – Vol 5-1 BT Journal [17]

Humans are able to see all these colors with the native eyes due to sensitivity of the eye to very narrow band of frequencies within the wide range of frequencies of electromagnetic spectrum. Visible range of the color starts from the wavelength $\mathbf{7.80 \times 10^{-7} m}$ and ends nearly at $\mathbf{3.90 \times 10^{-7} m}$ [18] which refers to "Red" and "Violet" in terms of the color wheel.

As the colors which are visible to the human eye is made out of electromagnetic rays even a slight change of the ray makes a difference in visible color to the eye. Which leads the human eye to identify millions of different colors. Computers can display more than 16.8 million colors. [19] This work is to identify a way to define the colors which can be displayed on screen into another color and make the HTML user interfaces changeable even after creating them. While there are millions of colors which the human eye can catch, all these colors are made

from three basic colors. We mix them as necessary to reproduce other colors. When we consider about the colors in real life and computer screen there is a major difference. To understand the difference of color reproduction in these two domains, we need to know about the addictive and subtractive colors.

2.6.1 Additive and Subtractive Colors

Computer screens, television screens and may other screens of various devices be using light rays to display colors in the screen. This is achieved by mixing spectral light in varying combinations. Background is a black screen with three primary addictive color rays which are red, green, and blue. By mixing these colors exactly one time we can produce subtractive primary colors, which are cyan, magenta and yellow. Since Black absorbs all the color rays mixing RGB gives us the white color, this technique is used in computer screens where there is a black light in in the background. Mix of CMY which stands for cyan magenta and yellow gives us white and we use them in the backgrounds of white.

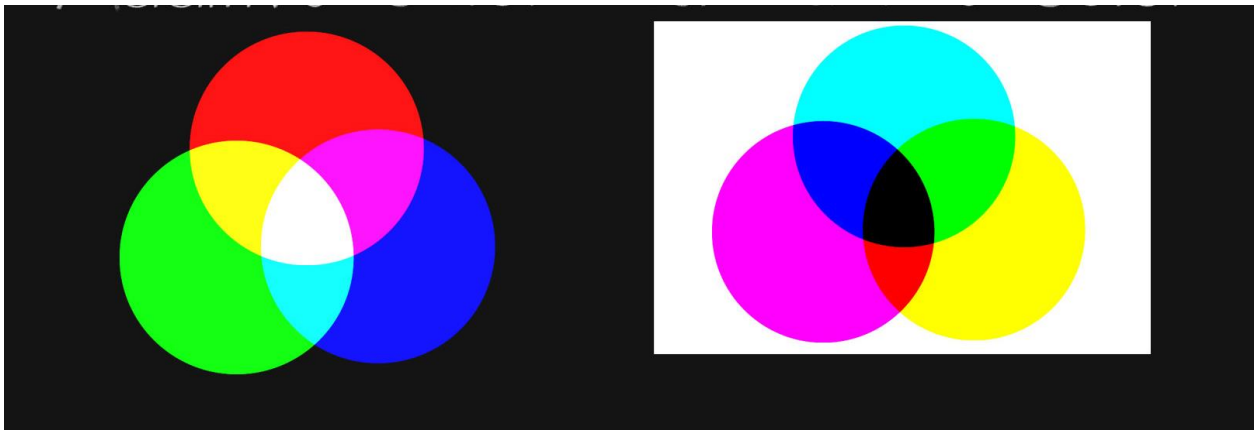


Figure 2.6: Difference of colors in light and colors in ink, left demonstrates the mixture of additive color and right side white background demonstrated subtractive color

2.7 Way to define CSS with Variables

W3C defines a way to use variables in the CSS documents [7]. This new feature gives us ability to define colors of the CSS page and we can use it to implement dynamic features to change the colors of the documents. Traditionally CSS uses the following way to define color of the HTML element in CSS document. There are three ways to access the HTML elements in CSS. HTML element can be accessed by element id property, class property or HTML tag name. For the simplicity we will use the class property of the HTML to demonstrate the way to define CSS.

```
<!DOCTYPE html>
<html>
<head>
	<style type="text/css">
		.class_1{
			color: blue;
		}
	</style>
</head>
<body>
	<p class="class_1">This is a sample paragraph</p>
</body>
</html>
```

Figure 2.7: Sample HTML code to define CSS

In figure 2.8 HTML paragraph tag is used to demonstrate the CSS definition. As we can see header section of the HTML page is used to define a class called “class_1” which makes color of the element to the blue color. This is applied to paragraph tag and it will make the specific element of the document to blue color.

Below is a demonstration of using variables in the CSS code. It will not directly bind the color or the value of the property to the html element. Rather it will define a variable in root location and that will be used to assign the value to the element.

```
<!DOCTYPE html>
<html>

<head>
      <title></title>
      <style type="text/css">

            :root {
            --main-color: blue;
            }

            .class_2{
            color: var(--main-color);
            }
      </style>
</head>
<body>
      <p class="class_2">This is a sample paragraph</p>
</body>
</html>
```

Figure 2.8: Sample HTML code to define CSS

From the demonstration above we can see new variables give us the ability to define CSS in a relative way. After the definition of the CSS we still can change the color in one place and then everywhere it will be changed respectively.

Using this feature and by defining color levels we can start creating a framework to handle dynamic color theming.

We will be going to use some base colors which will be defended in the framework. It will control the color relativity. Another mechanism which can identify colors of the objects and resolving color conflicts will be used to control the behavior of the respective document or the application.

Chapter 3

Analysis and System Design

This chapter will be giving an introduction of the system design and the principles behind the design. It will also give a brief overview to analysis of the framework. There are few feasible approaches to achieve the problem solution, here these approaches and problems faced are discussed.

3.1 Introduction to Design Principles

When it comes to the design of this framework it is highly depending on the key features and properties of the colors. Success of the color relativity is highly depending on the understanding to these concepts. Key concepts related to the design is discussed below in this section.

3.1.1 Color Relativity

To observe the natural colors and their behavior, we should consider the color relativity. Naturally color is relative. Relative to the light and to the other colors surrounded by that color. Human eye makes that relativity using the way human brain work with the eye, to determine the color of a particular object with the colors surrounded by that object. This relativity can be expressed by using the figure below.

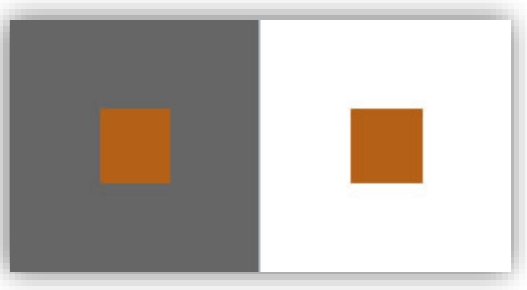


Figure 3.1 – Color relativity

When we see the figure 3.1 it appears to be four colors with the background colors gray and white respectively. But same orange color is there inside two boxes where the actual appearance is deferent with background colors. Orange in the gray color background is appears to be lighter than the orange in the white background. This refers to the color relativity in the color theory.

But in other hand the color relativity we are trying to achieve here is quite different then this color relativity in the color theory. Rather than considering the way colors appear with the surrounding we are trying to define or value the color, relative to another color which exist in the HTML page.

With the colors defined one related to another we can ask the designers to finish their design and even after that we can change the colors of the design without losing the color matching and details of color shades in the design. There are many ways to define colors but this method needs to follow the below conditions to be useful for our framework.

I. Should be mathematically calculable and answer or the color given by the answer should be in the visible color range.

This ensures that we can mathematically calculate the color values and apply them for the page or sections that we need. Furthermore, after the calculation answer of the calculation

which will be the color, should be in the visible color range or the so-called color wheel for us to apply it in the object that we need.

II. Definition of the color should be easy for the designers.

As no designers are willing to use a framework which is really hard to use and complicated out solution should be easy to use for the designers of the HTML pages. Therefore, initial color definition should be easy as possible.

III. Details of the colors or the shades should not be destroyed by the solution.

As the designers carefully design the HTML page many details will be there in the design, such as shades, highlight colors or may be lighter or darker color matchings. Our solution should be smart enough to keep this constancy and the accuracy when it changes the color of the page. This is the most important aspect with respect to usability of the whole framework.

Color relativity is not something straightforward to find and calculate. In colors there is no basement or strict place to define it relative to another color. Even if we do it is hard to generalize the solution as the boundary which is visible for human eye can be exceeded very easily. But still relativity of colors can be achieved with the properties of colors such as hue, structuration, lightness and intensity. We have considered the color schemes RGB and HSL with its internal properties to divide the color into few parts and understand the way to define colors relatively.

3.2 Approach to find relative Colors

3.2.1 Approach 1 – Using RGB Color Scheme

As the color can be represented with the RGB values, we define a base color and calculate the different of red, green and blue with the color of the element. After that when a new color is

chosen as the base color defined, relativity of RGB to that color is calculated and added with the previous base color. This will result a new color for the element relative to the original color of the element.

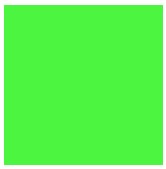
As an example of our base color is #60cf47 then the calculation goes as follows.

RGB values of #60cf47 – a green shade.

- Red – 96
- Green – 207
- Blue – 71

Now if we need to define the relativity of #4cf540 a light green color, values are as follows.

- Red – 76
- Green – 245
- Blue – 64



Now we will consider changing this green color to a blue color. Then the light green color is expected to go to a light blue color.

Base blue color: #4760cf

- Red – 71
- Green – 96
- Blue – 207

Let us see the calculation with green color we considered earlier.

- Red difference: 96 – 71 = 25
- Green difference: 207 – 96 = 111
- Blue difference: 71 – 207 = -136

New expected blue color is can be calculated with the light green we considered earlier.

- Red – 76 + 25 = 101
- Green – 245 + 111 = 356 = 365 – 256 = 109
- Blue – 64 + |-135| = 199

This color is #656dc7. As we can see the blue color which is predicted is lighter than the previous blue color and our work is successful to some extent. But there are issues with this mechanism and they are mentioned below.

3.2.2. Problems Faced in RGB Color Schema

This method is good for defining the relativity of colors only in few color range. Because as we know, RGB values are ranging from 0 to 255 and sometimes there can be minus values as the answer for calculation, for the relativity in the above method. Practically these values may not be in the visible color range or just some false values. Similarly, there can be situations where relativity value exceeds 255 with the calculation. This is a major problem when finding the relative color because there might be no color defined for the predicted color value if RGB relativity values exceeds the range.

One way to avoid this is recalculating the value again from the next as a color cycle. But when testing practically output color became less applicable for the real color which was originally defined. In other words, the predicted color is not matching with the color defined in the HTML element, and our goal is not achievable with that approach practically.

Another way to avoid this problem is to keep the maximum and minimum values at the upper bound and the lower bound of the range. It was solved the problem to a certain extent but, it was also not the best solution since after a certain extent the algorithm or the framework itself is not predicting a good matching color to the defined color of the HTML element. This was a good improvement for the current problem but a better solution was needed to achieve research goals.

This method was not selected due to the discussed issues above and the approach two discussed below was much better for the color relativity with practical applications.

3.2.3 Approach 2 – Using HSL Values of the Color

Properties in the color like hue, saturation and intensity is able to give a better solution to this problem. Color mixed by the HSL values can be used to generate the relative value and this can be used to predict the relative color.

Value of the Hue can go from 0 to 360 degrees. Hue denote the main color in the color mix with Saturation and Lightness. Since that hue can be used to define the colors relatively and saturation and lightness with define the shades of it.

This approach is using the HSL values of one color and calculates the saturation and lightness values of the new color related to previous color using simple mathematical calculation. Later in implementation section these calculations and the improvements made to this approach will be discussed.

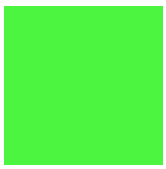
Here is the mechanism to find the color relativity with this approach and below is an example of our base color is #60cf47 then the calculation goes as follows.

HSL values of #60cf47 – a green shade.

- Hue – 108.97°
- Saturation – 58.62
- Lightness – 54.51

Now if we need to define the relativity of #4cf540 a light green color, values are as follows.

- Hue – 116.02°
- Saturation – 90.05
- Lightness – 60.59



Now we will consider changing this green color to a blue color. Then the light green color is expected to go to a light blue color.

Base blue color: #4760cf

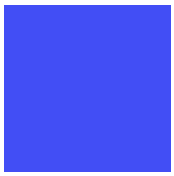
- Hue – 228.97°
- Saturation – 58.62
- Lightness – 54.51

Let us see the calculation with green color we considered earlier.

- Hue difference: 116.02°– 108.97° = 7.05°
- Saturation difference: 90.05– 58.62 = 31.43
- Lightness difference: 60.59– 54.51= 6.08 But we do not use this value always. We usually use the lightness value of light green and if required we change use this value.

New expected blue color is can be calculated with the light green we considered earlier.

- Hue – 228.97°+ 7.05° = 236.02°
- Saturation – 58.62 + 31.43= 90.05
- Lightness – 54.51+ 6.08 = 60.59 or



As we can see this blue color is more matching with the light green given previously. This mechanism does not exceed the values as the hue is cyclic and saturation, lightness has its own limitations as they are percentages. Hence, we have used this mechanism in the framework.

3.2.4 Problems in HSL Color Schema

HSL color scheme was quite good to handle the complexity of the relative color definition used in this work. But even after still there are limitations with the current application of this method in our algorithm. Major issue is to be discussed here would be lightness and saturation problem. Which is quite challenging to keep the accuracy of the HTML page when saturation and lightness changes with the newly applied colors. This problem is addressed to some extent where someone is able to use this solution with reasonable limitations. The solution we have given is users to choose whether to apply the lightness changes or to go with the initial

lightness values. The way this problem addressed and that solution is discussed later in implementation section.

3.3 Color Conflictions

When colors are mixing together some colors may not be able to see properly, this is mainly a mistake that can happen by the user when they are changing colors in multiple layers simultaneously. From the framework we can handle these kinds of issues and let the users to work without issues. This is focused when users are working with layered styles which will be discussed later in this document.

With the design of the solution we have layers (sections) that the user interface can be divided into. These areas can be used as separate color schemas so that user (or the UI developer/ engineer) will have a good flexibility for color theming in a one page of their interface. They can have many pages for their purposes. When users are defining multiple colors for the layers at the same time without paying attention to the output these color conflicts can occur.

With our solution initial color schema of the user is not destroyed. Hence if initial color designer has not done any serious mistake with color conflictions this framework's color predicting algorithm will never make a chance for a color confliction to be happen in single layer. But when it comes to multiple layers users are notified about the color conflictions where it is applicable. Detailed explanation of this will be in implementation section of this document.

3.3.1 Preventing Color Confections and its Limitations

Initial understanding was to prevent color conflicts by using color validation rules. Later it was realized a rule set is not able to prevent possible color conflictions. Color conflictions which may arise within the closer elements cannot be identified easily with the way HTML pages work. Even if we make a rough sketch to identify that it will not be a general solution since many HTML pages are dynamic and HTML elements are not consistently static. When

we try to give user a warning the warning may be a false one if the element is changed on the fly dynamically. This identifies issue lead to stop developing the color conflicting rule set and we had to look for another solution to prevent the color conflictions.

With many thoughts and practical tests, we came to a point that these conflictions can be prevented with manual layered approach used for the HTML documents. This was further researched practically and the solution was fascinating. Finally, we decided to use this as the color conflict prevention mechanism.

Web page is layered for the different color areas by the naming convention of the CSS variables. This way different areas of the web document are identified and treated separately. Since users can change the colors of the different layers independently possible color confections are easy to see in the HTML document at the time of changing the color. Furthermore, in the control panel of the framework there we have created a button to control the lightness and with this change, users can choose to use the initial lightness or to change the lightness value to a new one. This led to pass the control more to the side of user but it led for users to have a wide range of color designs.

3.4 Usage of the framework

Dynamic color relativity easy to achieve with our proposed solution. Most important fact is that our solution will be able to use by almost every technical stack which uses HTML and CSS for their user interface designing and where JavaScript can be used. Every browser today is compatible with these three technologies since it is the standard for every browser which is deigned to view web using HTTP traffic. But users should follow few steps in-order to get the benefits from our new proposed solution. Detailed user manual will be appended into appendix 1.

1. Users must create HTML pages by using standard HTML, CSS and they need to include a JavaScript file in their solution.

2. Our solution uses CSS variables to achieve the color relativity. Hence users will need to define CSS variables in their CSS sheets and define colors to the elements by using that CSS variables.
3. Users need to define the sections and the main color for these sections respectively.
4. Once everything is done, users can change the color of their design by changing only the main color of the section or the page. This will result a whole color change in every HTML element in their user interface.

3.4.1 Steps to use the solution

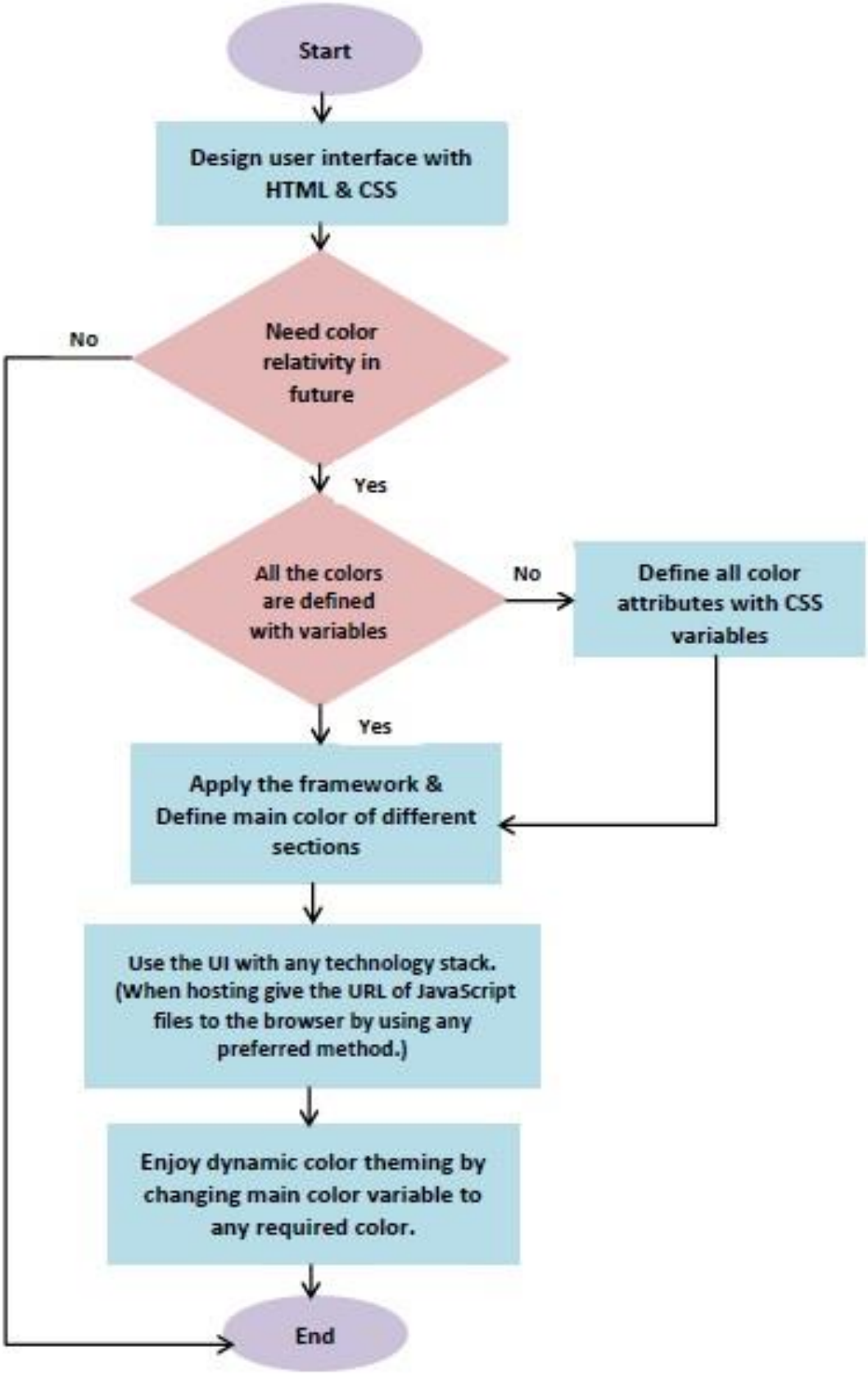


Figure 3.2: Illustrates the steps to use the solution

Chapter 4

Implementation

4.1 Overview

Purposed solution for the color relativity algorithm and relative color predicting was implemented in few stages. Development of the algorithm and framework was not straightforward but methodology was selected while doing the researching and testing for a suitable approach. This chapter will explore the used concepts and technologies with reasons, and the flow of the solution in a practical application to a general HTML page.

4.2 Technologies, Libraries and Frameworks

4.2.1 Technologies

- Hyper Text Markup Language (HTML) – Version 5
- Cascading Style Sheets (CSS) – Version 4
- JavaScript (Native version without any libraries) – Version ECMAScript 7

As mentioned above this work is achieved the state of working with standard browsers without any external installation. This was a key goal that wanted achieve since it helps the framework to work with any technology without conflictions.

Standard browsers support for HTML, CSS and JavaScript natively. All the websites are either natively written in these technologies or converted into these technologies for reliability. This framework should work with any current technology or framework due to this

reason. Apart from the technologies mentioned above there are no any other library or framework used to implement this solution.

4.2.2 Tools and the Environments

- WebStorm 2016.3.1 as the development IDE
- Google Chrome for the main testing browser.
- Mozilla Firefox browser for testing
- Safari browser for testing
- Opera browser for testing
- Edge browser for testing
- Internet Explorer browser for testing
- Windows 10 as the environment.

Many standard browsers are used for the testing. But as we can see there are no much advanced tools used as we wanted our solution to be technological independent from many of the technologies available.

4.3 Implementation

Framework is named as “Chameleon JS” this framework contains few CSS and JavaScript files. Following figure show the way these files are organized into the framework.

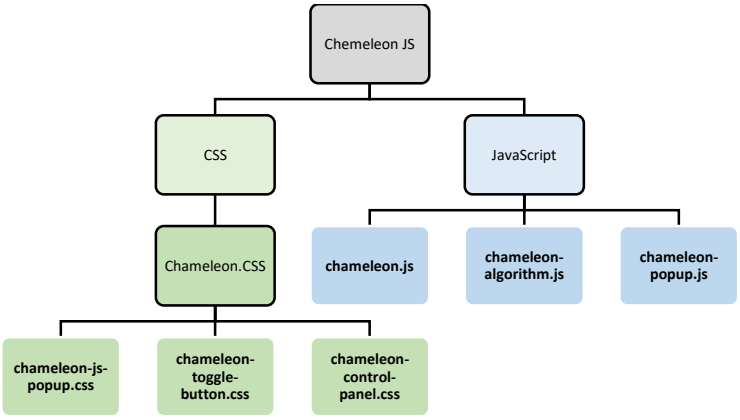


Figure 4.1 - Illustrate organization of the framework

4.3.1 CSS

Framework contains four CSS files as described below. They are categorized with the purpose of their usage.

- chameleon.css – Used to define the root CSS element of the document. Other three CSS files are included into this file. Hence this file is sufficient to include in any HTML document.
- chameleon-js-popup.css – CSS definitions of the popup panel is defined here.
- chameleon-toggle-button.css – CSS definitions of the toggle button which is used in control panel is defined here.
- chameleon-control-panel.css – CSS definitions of the control panel is defined here

4.3.2 JavaScript

Framework contains three JavaScript files described as below. They are categorized with the purpose of their usages.

- chameleon.js – Main functional definitions and area for the users to include their CSS variable names is included in this script file. Should be added to the end of the HTML document.
- chameleon-algorithm.js – Main Algorithm and the works are defined here in this file. Should be added to the end of the HTML document.
- chameleon-popup.js – Script definitions for the popup to work is included in this file. Should be added to the end of the HTML document.

4.4 Flow and the Design of the Framework

Following is the main flow of the framework. This includes user interaction only in the situation where they change the color of layers. Flow will be started when the page is fully loaded in the client browser. After successful loading framework is doing many works underneath by using JavaScript. Below figure explains the way they are interacting at a particular stage.

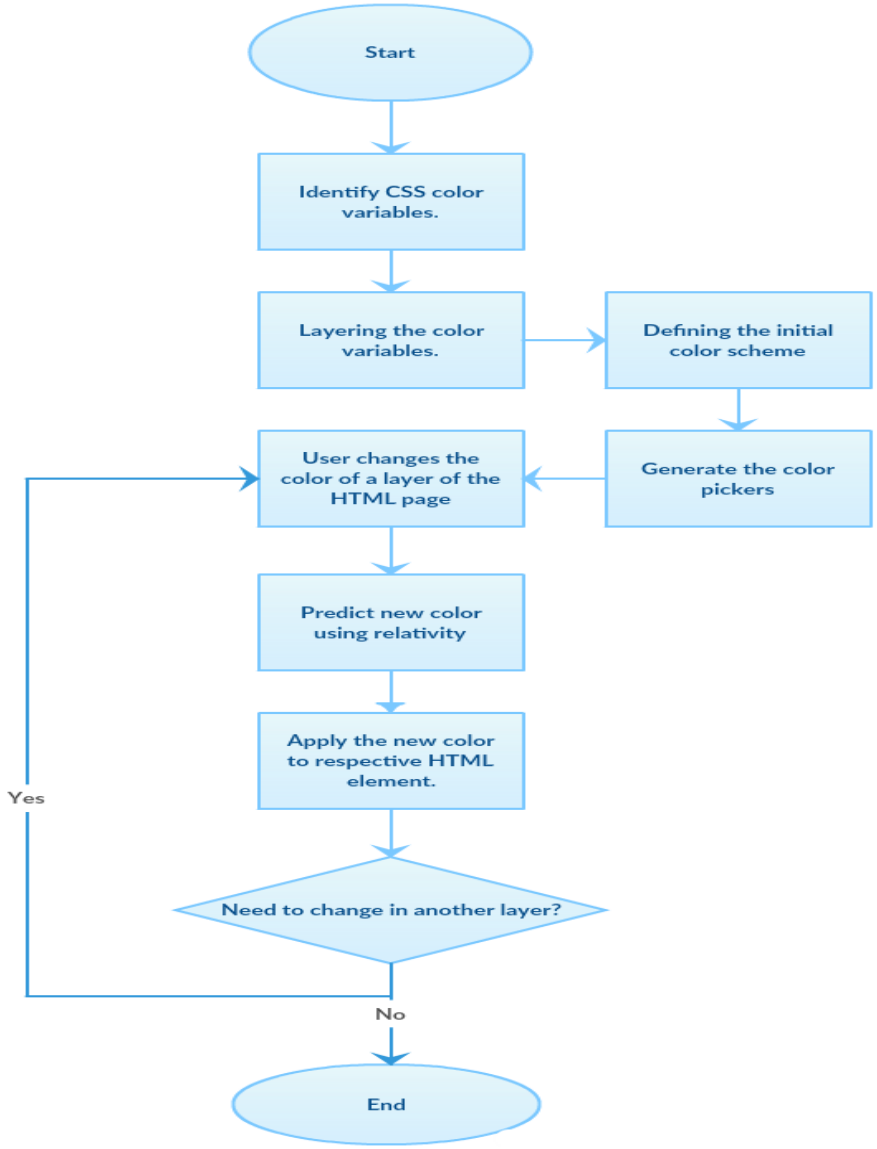


Figure 4.2 - Illustrate the flow of framework and design

As explained in figure 4.2 following steps will be the main key points in the framework.

I. Identify CSS color variables.

II. Layering the color variables.
III. Defining the initial color scheme.
IV. Generate the color pickers.
V. User changes the color of a layer of the HTML page.
VI. Predict new color using relativity
VII. Apply the new color to respective HTML element.

Execution and interactions of these steps will be explained here in detail with the code snips used in the framework to understand the framework and its behavior.

4.4.1 Identify CSS color variables

First step would be to identify the CSS color variables which are defined by the user. These variables contain the initial colors for the HTML page in many forms. Figure 4.3 is an example of color variables defined in the HTML page.

```
:root {
    --btn-color : #60cf47;
    --btn-hover-color : #4cf540;
    --text-background-color : #d3f7d1;
    --input-hover-border-color : #4cf540;
    --background-main-color : whitesmoke;

    --L1-bg-color  : #16CF6D;

    --L2-bg-color-1  : #4ccf36;
    --L2-bg-color-2 : #85cf4e;

    --L3-color : #85cf4e;
}
```

Figure 4.3 – CSS variables with colors in HTML page

These variables are stored in an array called “variablesArr” and used within the framework to access the initial colors and to apply the relevant color to the particular HTML element. Name of the variable will be starting with “--” according to the standard way in defining CSS variables. [8]

4.4.2 Variable naming convention

Variables defined in this framework follows a naming convention. Users of the framework should follow this naming convention in order to define the layer of the variable which they are referring to. Naming convention should follow the following rules in order to achieve the color relativity in HTL documents.

I. All the names should be declared as CSS variables and assigned to a color.

II. Names must be unique, but one unique variable can be used in many places in the HTML page, where users need to have the same color.

III. Name should start with "--" due to the standards in CSS variables.

IV. Names which needs to go to particular layer should mention the layer name starting with "L" followed by layer number. Ex: "--L1-my-background-color"

V. Variables which does not need to go to a particular layer can be declared without "L" but these variables will fall into the category of default layer which is the "0" layer.

4.4.3. Layering the color variables.

Framework support for users to define variables in separate layers. These layers can be used to define the similar colors in the same layer or to differentiate the sections in the HTML page such as let navigation sections or header sections which may have a different color from the other sections of the HTML page. Following are the special things to notice with layers and what the users can do with layers. Layering is important as it acts as the color conflicts preventing mechanism of this framework.

- Each layer will be able to control the colors separately. For this purpose, there will be a color picker automatically generated by the framework with the HTML page.

- Users can define any number of layers but defining large number of layers may hard to handle for users due to complexity.
- Users can use color variables to define new layers.
- Layers can be used with different colors for different layers or different sections for different layers, depending on the users wish.
- It is recommended to use different color variations as different layers since it will be easy to change the colors of the HTML document.

Figure 4.4 explains the function used by the framework to layer the variables used in the HTML page. As we can see function look for the each and every variable in "variablesArr" array. Further it stores the variable in a relevant array and later combine these arrays to finalize the CSS color variables in the HTML page into layers.

```
function runForLayerVariables(){
    //to store arrays of layer variables. 0 index will be the default page variables.
    // otherwise exact layer name will be used as the index of this array
    var layerVariableArrayStore = [];

    for(var i=0; i< variablesArr.length; i++){
        var varName = variablesArr[i];
        var nameSlices = varName.split("-");
        var layerName = nameSlices[2].replace('"',"");

        if(!layerName.match("[/(L)(0-9)/]")){
            layerNames[0] = "L0"; //default layer is zero layer
            /* Inserting new css variable name to the array */
            var currentLayer = layerVariableArrayStore[0];
            if(!currentLayer){
                currentLayer = [];
            }
            currentLayer[currentLayer.length] = varName;
            layerVariableArrayStore[0] = currentLayer;
        }else{
            if(!layerNames.includes(layerName)){
                //inserting the layer in the array
                layerNames[layerNames.length] = layerName;
            } // else Layer is already in the array

            var num = layerName.replace("L","").replace('"',"");
            var currentLayer = layerVariableArrayStore[num];
            if(!currentLayer){
                currentLayer = [];
            }
            currentLayer[currentLayer.length] = varName;
            layerVariableArrayStore[layerName.replace("L","")] = currentLayer;
        }
    }
    return layerVariableArrayStore;
}
```

Figure 4.4 – Run for layers function.

4.4.4 Initial color scheme.

Initial user color scheme is taken into variable array called “layerVariableAndColorsArray”. This stores all the initial colors which the user has defined at the design time. In other words, this array will store all the original colors of the HTML page.

When changing colors of the HTML page dynamically this initial color scheme is used to calculate the relativity of the colors without losing the important details of the colors in HTM page.

Defined CSS variables will be layered according to a naming convention recommended by the framework. Layer system is important since it give the users the ability to control the colors of a section of the HTML page separately. This way users can control the color of the HTML page very sensitively. Correct layering of the HTML page may overcome the limitations of the color relativity algorithm.

Figure 4.5 shows the function used by the framework to define the initial color scheme for a particular HTML page. This function uses another function called “defineInitialElementColors” which is a supporting function described in figure 4.6. This function takes a particular CSS color variable and look for the color defined with the variable. This work is repeated for a particular layer and resulting colors are saved into an array.

```
function runForLayerVariableColors(layerVariablesArray) {
    var initialColors = [];
    for(var i=0; i<layerVariablesArray.length; i++){
        var array = defineInitialElementColors(layerVariablesArray[i]);
        initialColors[initialColors.length]  = array;
    }
    return initialColors;
}
```

Figure 4.5 – Run for layer variable colors function.

```
function defineInitialElementColors(layerVariables){
    var elementColorsArr = [];
    for(var i=0; i< layerVariables.length; i++){
        var styleHoverColor = getComputedStyle(document.body);
        var elementColor = styleHoverColor.getPropertyValue(layerVariables[i]);
        elementColorsArr[elementColorsArr.length] = elementColor;
    }
    return elementColorsArr;
}
```

Figure 4.6 – Define initial element colors function.

Figure 4.7 shows the final function which is run for layer variable colors function. This function is used to complete the color variable array which actually is an array of an array. As explained earlier in this section this function is used as a measurement when changing the colors relatively.

```
function runForLayerVariableColors(layerVariablesArray) {
    var initialColors = [];
    for(var i=0; i<layerVariablesArray.length; i++){
        var array = defineInitialElementColors(layerVariablesArray[i]);
        initialColors[initialColors.length]  = array;
    }
    return initialColors;
}
```

Figure 4.7 – Run for layer variable colors function.

4.4.5 Generating color pickers

Color pickers will be generated in a separate section of the HTML page. Users can define their own location in the HTML page for this. Further users need to create a “div” element or any other HTML element which can give some space for color pickers to be created in. This element should be tagged with a “id” property equal to “color-box-div”. color pickers will be generated inside this HTML element automatically by the framework. It is important to notice that separate colors pickers will be generated for each of the layer which are defined by the

color variables. Separate color pickers will give the ability for users to change the colors of the layers separately.

Figure 4.8 shows the generate color pickers function which is responsible for creating color pickers for each layer in the HTML page. This function looks for all the layers and asks the "createColorPicker" method to generate a color picker for each of the layer. Figure 4.9 shows the create color picker function which is responsible for creating particular color picker. As we can see after creating a color picker for the respective layer the function looks for the HTML element with id equaling to "color-box-div" and append the color picker as a child element of that element. Since it used HTML 5 standard color picker any browser which is compatible with HTML 5 will support for this function.

```
function generateColorPickers(layerVariableAndColorsArray) {
    for(var i=0; i <  layerVariableAndColorsArray.length; i++){
        createColorPicker(i,layerVariableAndColorsArray[i][1][0]);
    }
}
```

Figure 4.8 – Generate color pickers function.

```
function createColorPicker(id, color) {
    var colorPicker = document.createElement('input');
    colorPicker.id = 'color-picker-input-'+id;
    colorPicker.type = 'color';
    colorPicker.value = color.trim();
    colorPicker.onchange =  function () {updateFavouriteColor(this.value,id);};
    var colorBoxDiv = document.getElementById("color-box-div");
    colorBoxDiv.appendChild(colorPicker);
}
```

Figure 4.9 – Run for layer variable colors function.

4.4.6 Color of a layer

After the HTML page fully loaded, users can start changing the colors of the defined layers. At this stage color picker for the layers are created already as it has explained in the previous sections. Users may select any color out of 16.8 million colors by using the color picker.

Color predicting algorithm will predict the colors of other HTML element in the same layer with the aid of initial color scheme. Users can have look at their web pages as never before with dynamic color theming and choose the most suitable color theme by seeing it directly rather than guessing the suitable colors as they used to do in the traditional way. Figure 4.10 shows the implementation of this work with the function named as update favorite color function.

This function run after user selects a particular color within 16.8 million different colors. This function takes two arguments as the new color and the layer number. Basically, this function looks for each and every variable defined in the layer and asks generate relative color HSL function to give a matching color for the variable. It is important to notice that function uses the 0^{th} index of the initial color array as the base color for the layer. This is default behavior but if users need to change the basic color they have to edit the code lines of the framework and define a relative base color. In many situations it is very important for base color to be a good average color within the used colors of the framework since color predicting ability of the algorithm highly depends on this base color. For the simplicity of the framework for the users to use this method uses the 0^{th} index as the base color of that respective layer. This method uses a helper function called "generateRelativeColorHSL" and this is the magical part of the framework which will be discussed in the section 4.3.7

```
function updateFavouriteColor(newColor,layerNumber){
    const elementColorArray = layerVariableAndColorsArray[layerNumber];
    console.log(elementColorArray);
    for(var i=0; i< elementColorArray[1].length; i++){
        var newElementColor = generateRelativeColorHSL(elementColorArray[1][0],newColor,elementColorArray[1][i]);
        document.documentElement.style.setProperty(elementColorArray[0][i],newElementColor);
    }
}
```

Figure 4.10 – Update favorite color function

4.4.7 Prediction of new colors using relativity

After user change the color by using the color picker it will directly change the colors of the HTML page. But to achieve that there will be a process going underneath. Predicting new colors can be achieved with the relative color calculating functions written in the framework. This is referred as the color predicting algorithm and appendix B shows this important method

and its description. This method accepts for three variables called base color, new Main color and actual color. By calculating the HSL parameters of these colors by using the function shown in appendix B. This method calculates the relativity of the new color related to the color that was existing in the element before. By changing these parameters to some extent this method successfully outputs the predicted relative color for the given color of the HTML element.

4.4.8 Apply the new color to respective HTML element

Finally, the framework applies the generated color to the HTML element by using the CSS variable which is there for each and every HTML element. Users can see the applied new colors and make changes to the color as much as they need. It is important to notice that persistency for the results are not saved in the HTML pages by this framework. Users (Developers) need to implement their own way to store the newly generated colors and apply them at the start of the HTML page. There can be many methods for users (developers) to achieve the persistency and handling that complexity exceeds the limitations of this research work.

4.5 Control Panel of the Framework

Control panel contains few elements of the framework. Following points explains about the important functions of the control panel.

- Each layer will have a separate control are to change the color of the layer and a button to decide whether to change the lightness values or not.

- There is a panel at the bottom of the control panel to control the persistency of the framework and the results. Users can save the initial color matrix and / or the current color matrix. These matrixes are save as .txt files to the local computer they are using. file will be in the format of JSON. Users can load the JSON file again and apply the relevant color set again. Users can use this as a mechanism to test the color combinations and match them as they need.

- It is important to notice that this persistency is only for the convenience of users and not a permanent persistency solution. The initial understanding was persistency to be handled by the developers as per their respective applications.

- This control panel will be shown as a popup panel. This panel will work in any browser since this is created as a native CSS solution.

- Some basic styling has been provided by the framework for this control panel. But styling of the control panel should be handled by the developers as per their applications.

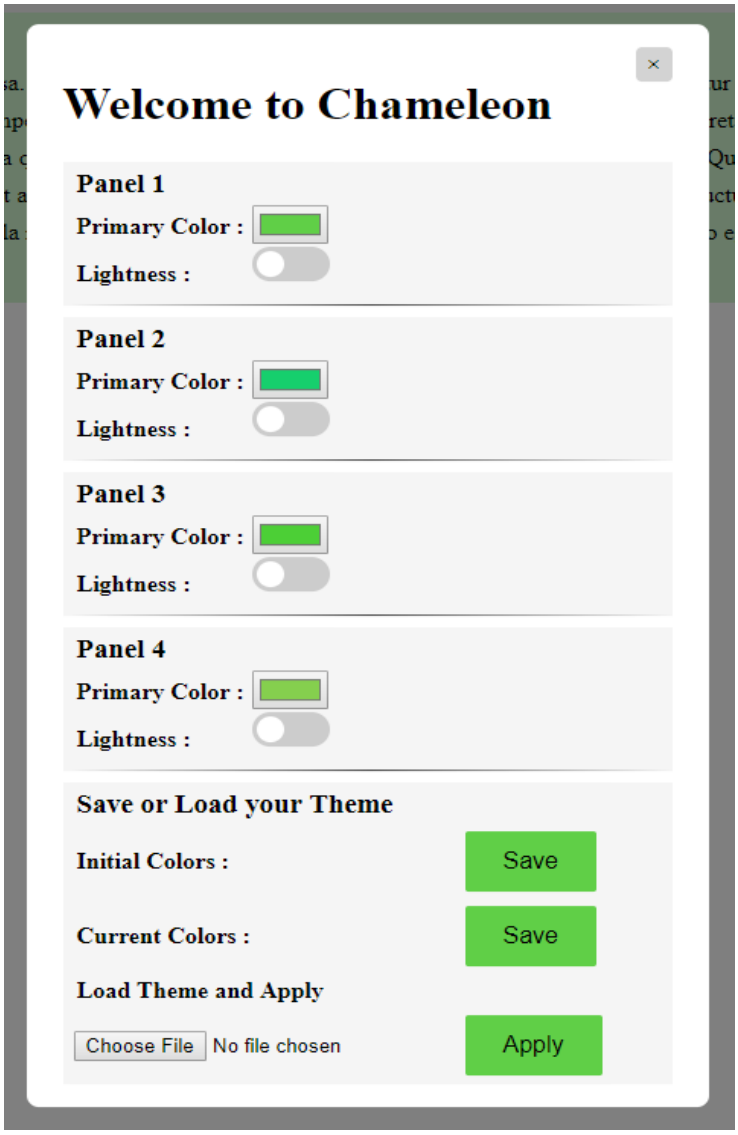


Figure 4.13 – Control Panel of the Framework

4.6 Automatic Generation of UI elements

User interface elements which are used in framework are automatically generated by using several JavaScript functions. Following figure is used to generate the main div of the framework. For other user interfaces there are similar functions used in the framework.

The id property is used little bit long to make sure developer web site's id will not be conflicting with the id of a framework element.

```
/* Interface Generation functions */
function generateChameleonMainDiv() {
    var chameMainDiv = document.getElementById("chameleon-js-main");

    /* Main Button */
    var btn = document.createElement('button');
    btn.classList.add('chameleon-js-popup-trigger');
    var btnText = document.createTextNode('Chameleon Effect');
    btn.appendChild(btnText);
    chameMainDiv.appendChild(btn);

    var chamePopupDiv = document.createElement('div');
    chamePopupDiv.classList.add('chameleon-js-popup-modal');

    var chamePopupContentDiv = document.createElement('div');
    chamePopupContentDiv.classList.add('chameleon-js-popup-modal-content');

    var span = document.createElement('span');
    span.classList.add('chameleon-js-popup-close-button');
    span.innerHTML = "&times;";

    var h1 = document.createElement('h1');
    var h1Text = document.createTextNode('Welcome to Chameleon');
    h1.appendChild(h1Text);

    var colorBoxDiv = document.createElement('div');
    colorBoxDiv.setAttribute('id', 'color-box-div');

    chamePopupContentDiv.appendChild(span);
    chamePopupContentDiv.appendChild(h1);
    chamePopupContentDiv.appendChild(colorBoxDiv);
    chamePopupDiv.appendChild(chamePopupContentDiv);
    chameMainDiv.appendChild(chamePopupDiv);
}
```

Figure 4.14 – Automatically Generated UI Elements

4.7 Distribution of the Framework

Framework should be distributed to the developers who are looking for the color changes. There should be a mechanism to spared the details and the instructions. In order to facilitate the information through the open internet there is a web site which is created for this framework. Website is hosted in an Amazon S3 bucket as a static web site. URL to the website is as www.chameleonjs.xyz

Initial framework package can be downloaded from this web site. Website is created as a single page website. Details and instructions to use the framework, technical information contact information and much more information is given on the web site. Following figures show the contents and the way the website is organized.

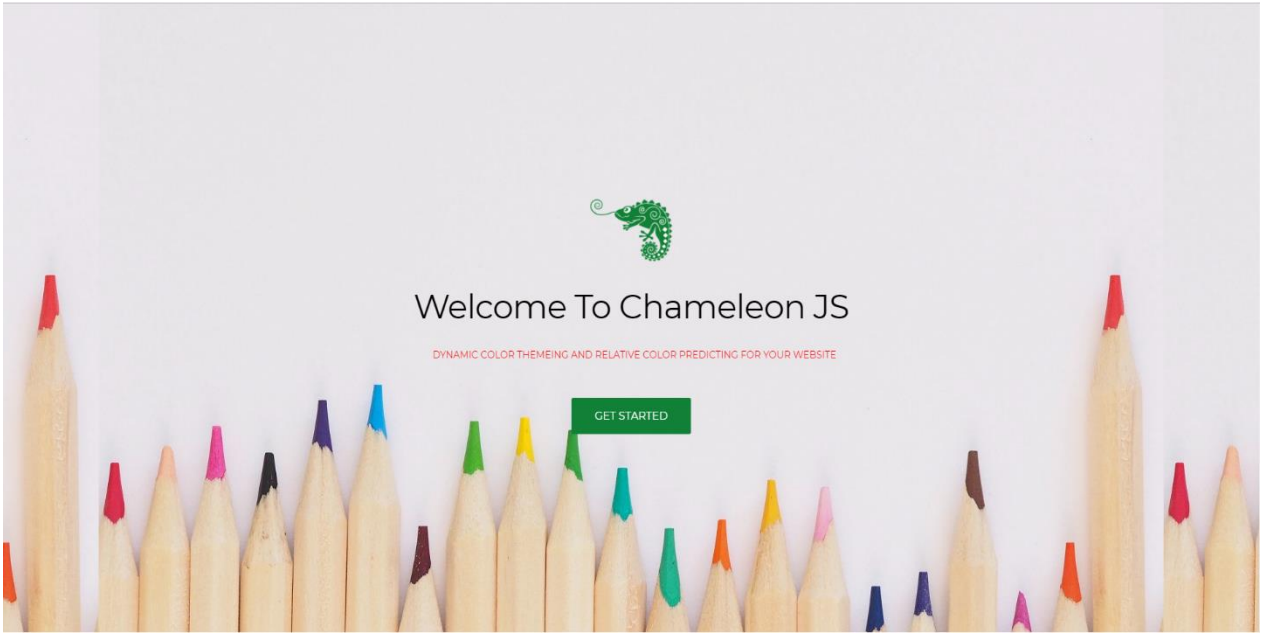


Figure 4.15– Home page of the website

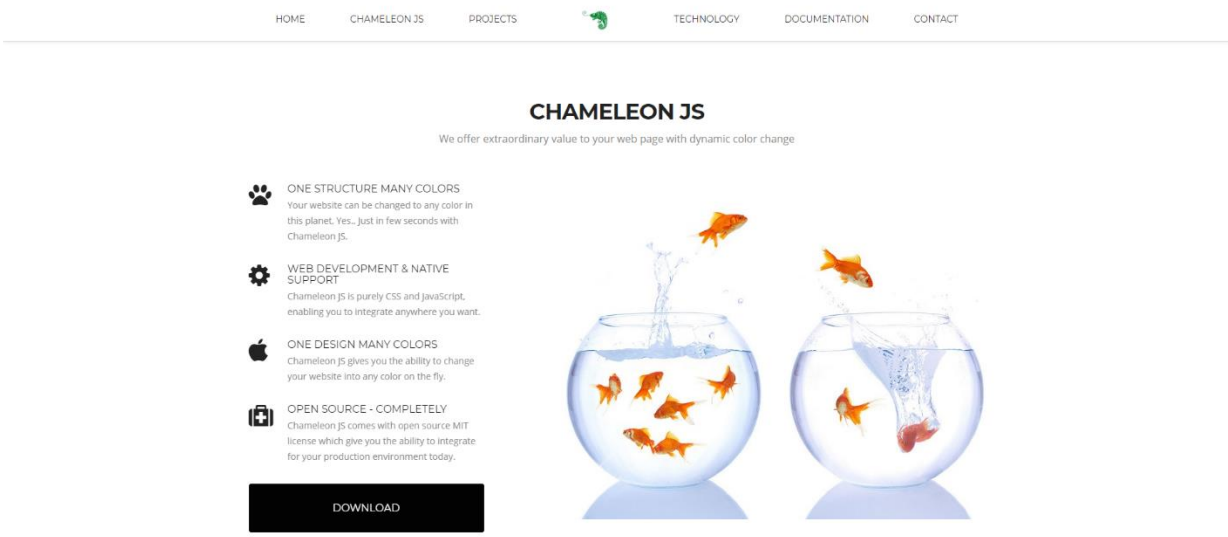


Figure 4.16 – Introduction page of the website

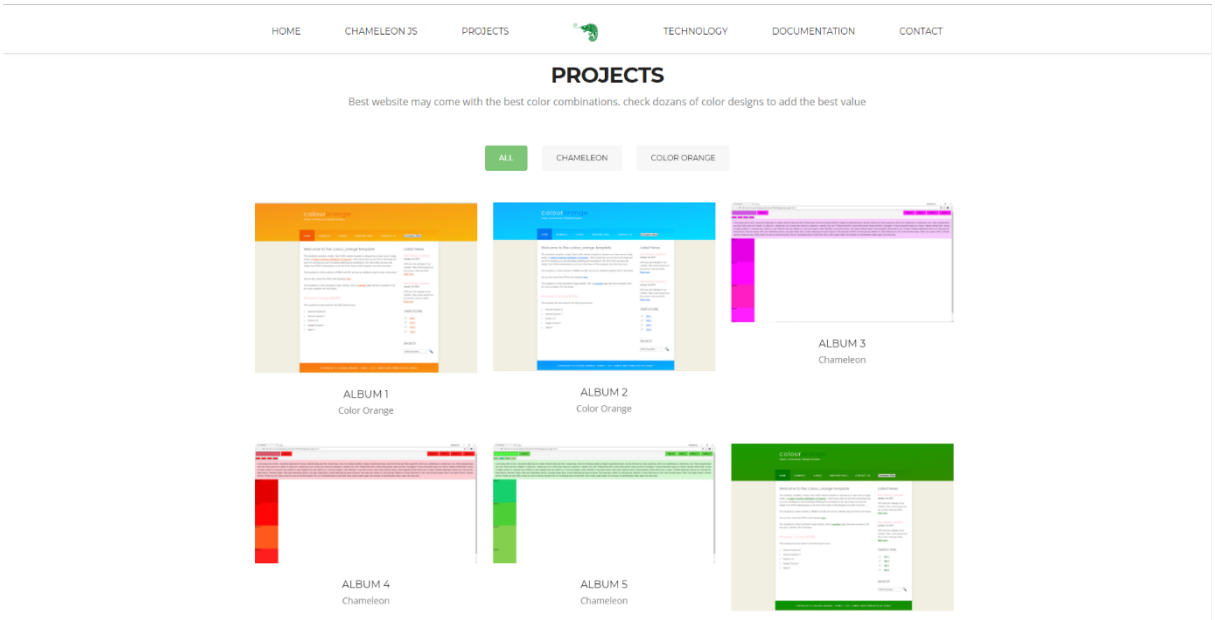


Figure 4.17– Projects page of the website

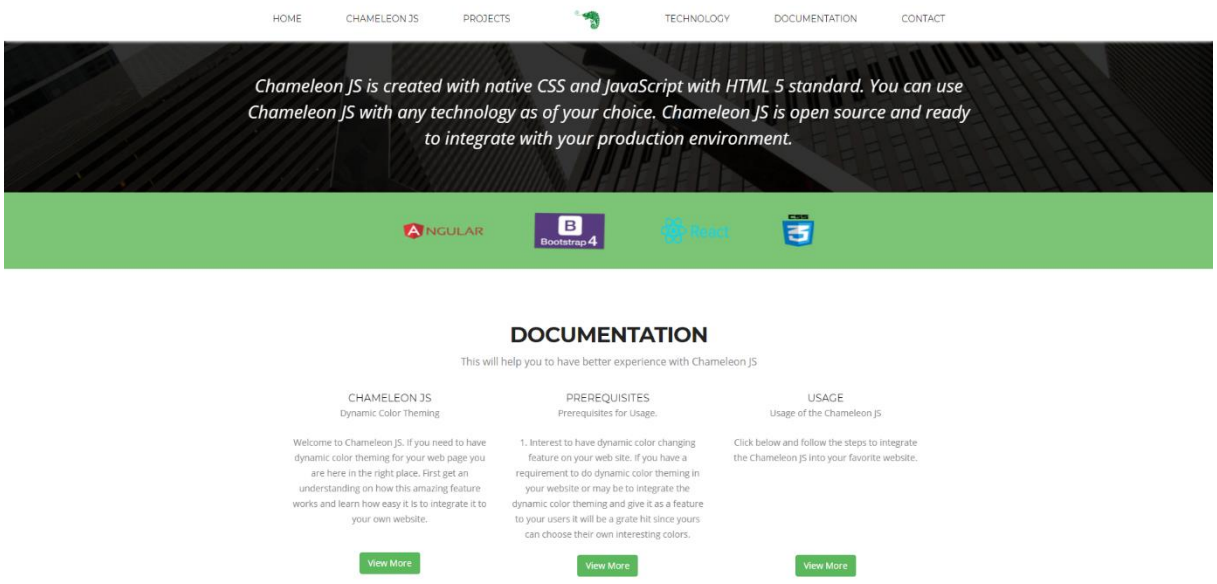


Figure 4.18 – Documentation page of the website

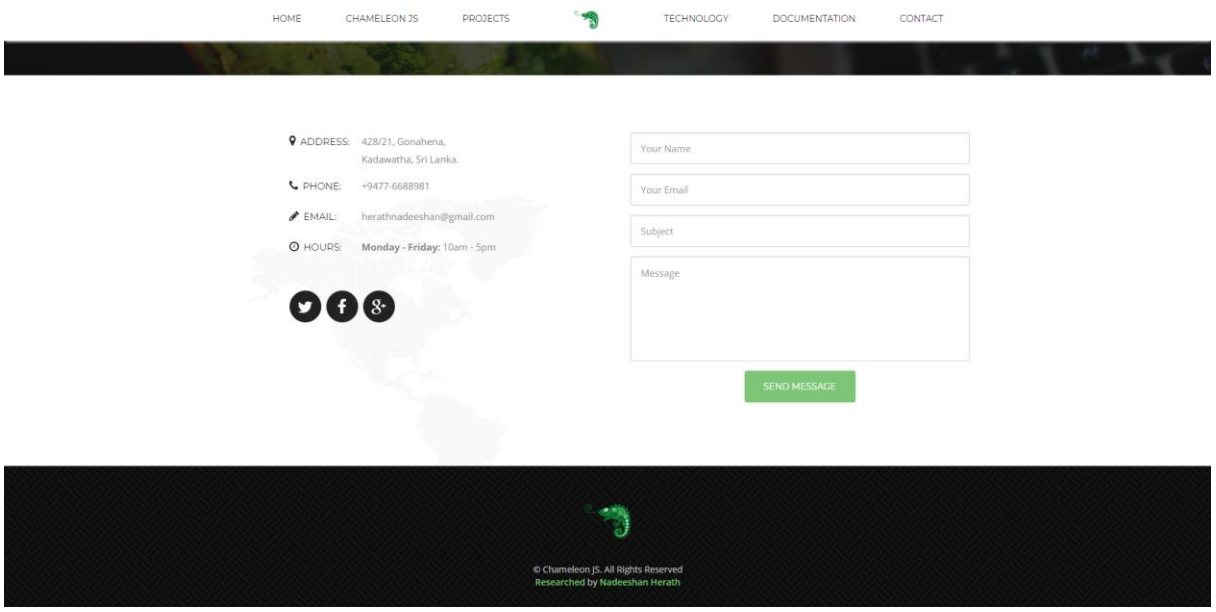


Figure 4.19 – Contacts page of the website

Chapter 5

Evaluation

The evaluation process is the one of the most important roles of the research. This chapter is devoted to describe the evaluation process carried out for our proposed Cascading Style Sheet framework for dynamic color theming and relative color predicting. With this work we should be able to implement the framework in our web applications and change the colors of a HTML page(s) into any color that can select from the color wheel.

5.1 Evaluation approach

Dynamic color theming framework is an innovation to color theming world and carried out as an experimental research. It is important to evaluate this framework correctly to measure the success and the usability of the framework.

This work defines the color relatively by using HSL color scheme. Suggested algorithm is able to predict a suitable color with respect to the existing color and the new color. This work is tested several times by using standard HTML pages and standard (famous) web front end technologies. Works carries out for this are listed below.

- Testing the solution with sample HTML pages.
- Testing the cross-browser compatibility with several standard web browsers
- Testing the solution with other standard frontend web technologies.
- Carryout a survey with experienced web developers to get their feedback.
- Evaluating the used approach and problems faced.

5.2 Testing the solution with sample HTML pages

Sample HTML page was created to test the framework by using only necessary elements in the HTML page. Following figure 5.1 shows the original design of that HTML page.

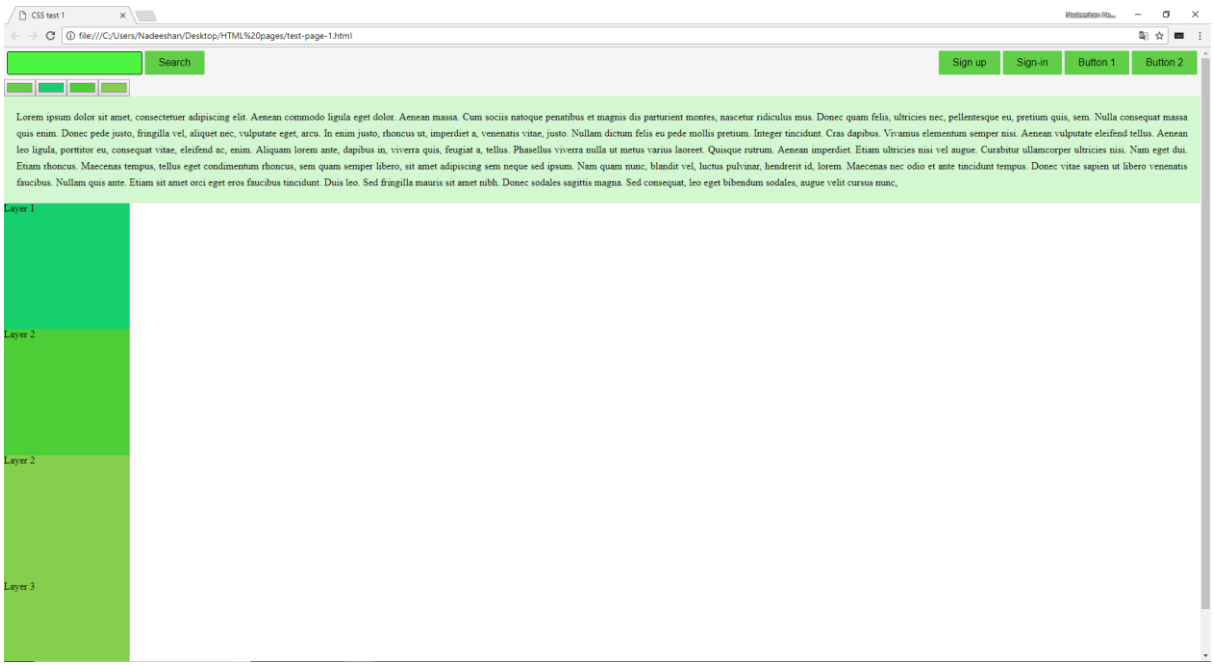


Figure 5.1 – Original HTML page

This page consists of HTML elements with several color variances. To illustrate the color pickers and the way they work color pickers are placed below the header section of the page. There are 4 layers in the page. The default layer and 3 more layers with color boxes to illustrate the color layering ability of the framework. Figure 5.2 shows the color pickers and the way users can use to change the colors dynamically in the HTML pages.

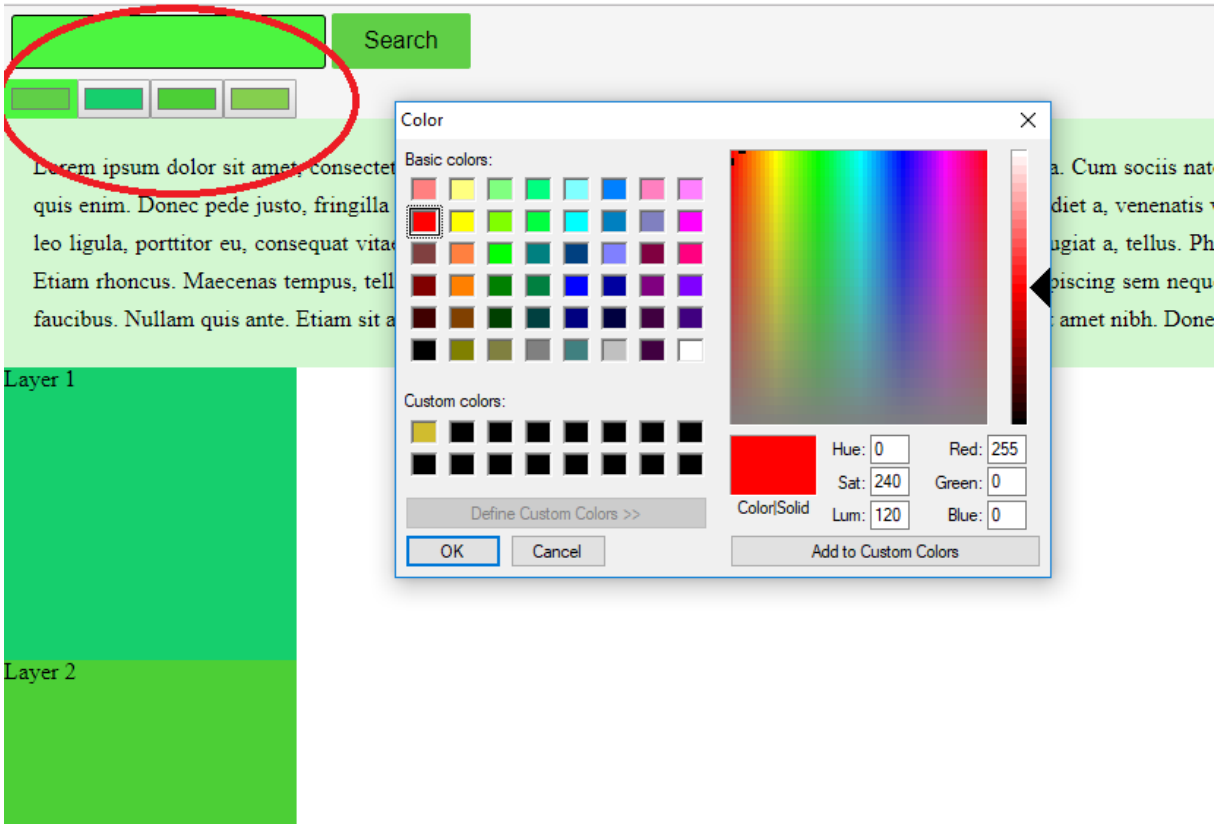


Figure 5.2 – Color pickers of the HTML page

Users may select their favorite color within 16.8 million colors of the color picker. Following figures 5.3 to 5.6 shows the step by step color selection for this HTML. In this situation we assume that user is trying to change the color to a red for all the layers.

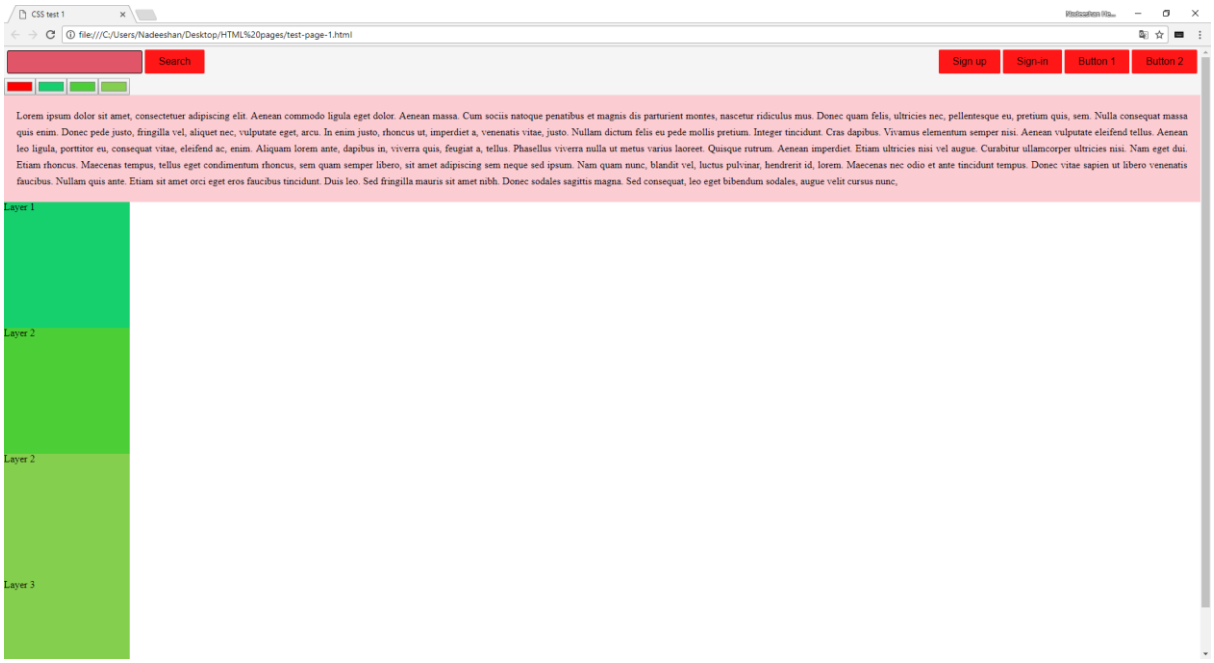


Figure 5.3 – Changing color to red in default layer of the HTML page

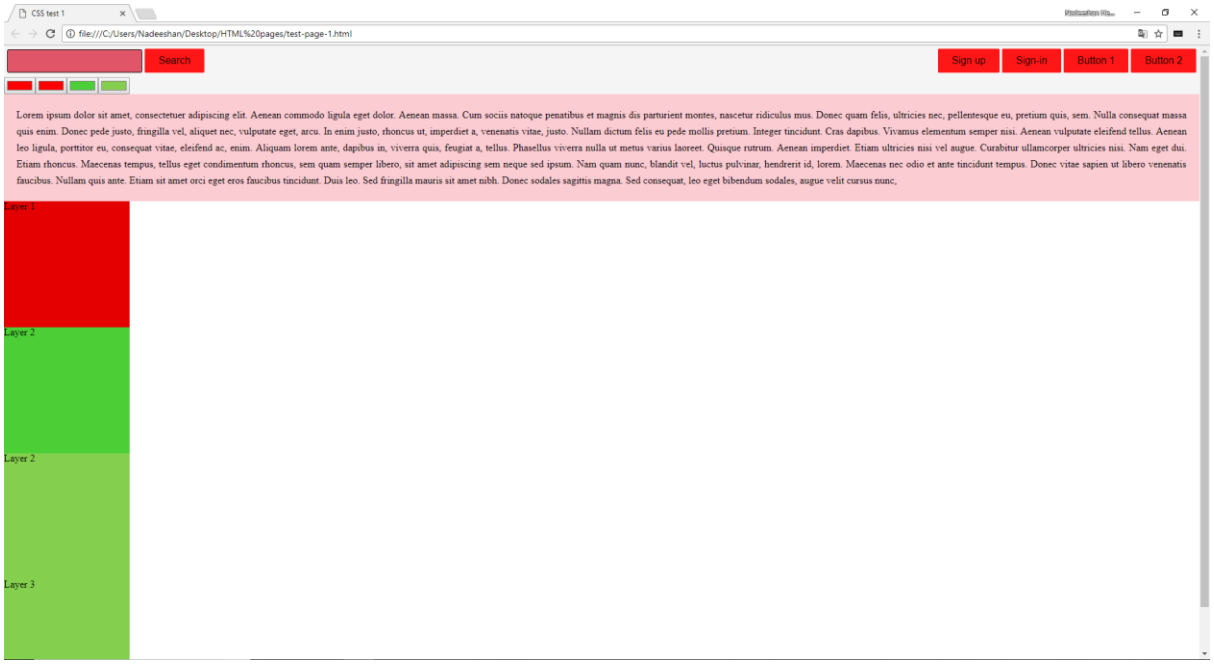


Figure 5.4 – Changing color to red in layer 1 of the HTML page

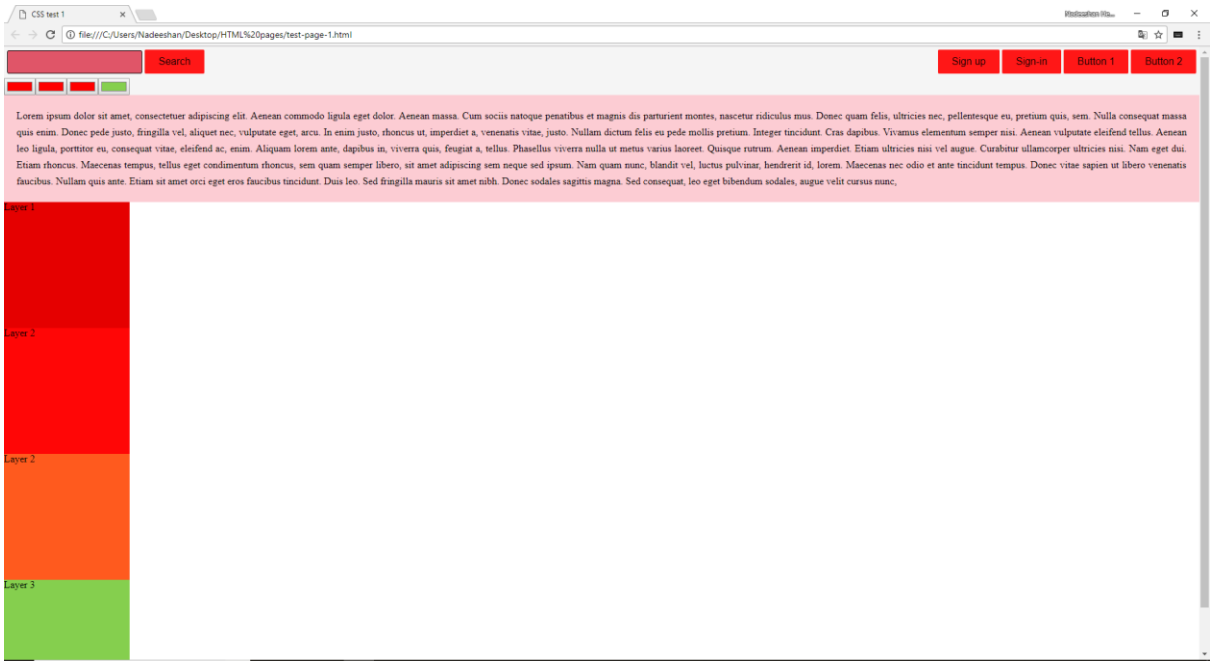


Figure 5.5 – Changing color to red in layer 2 of the HTML page

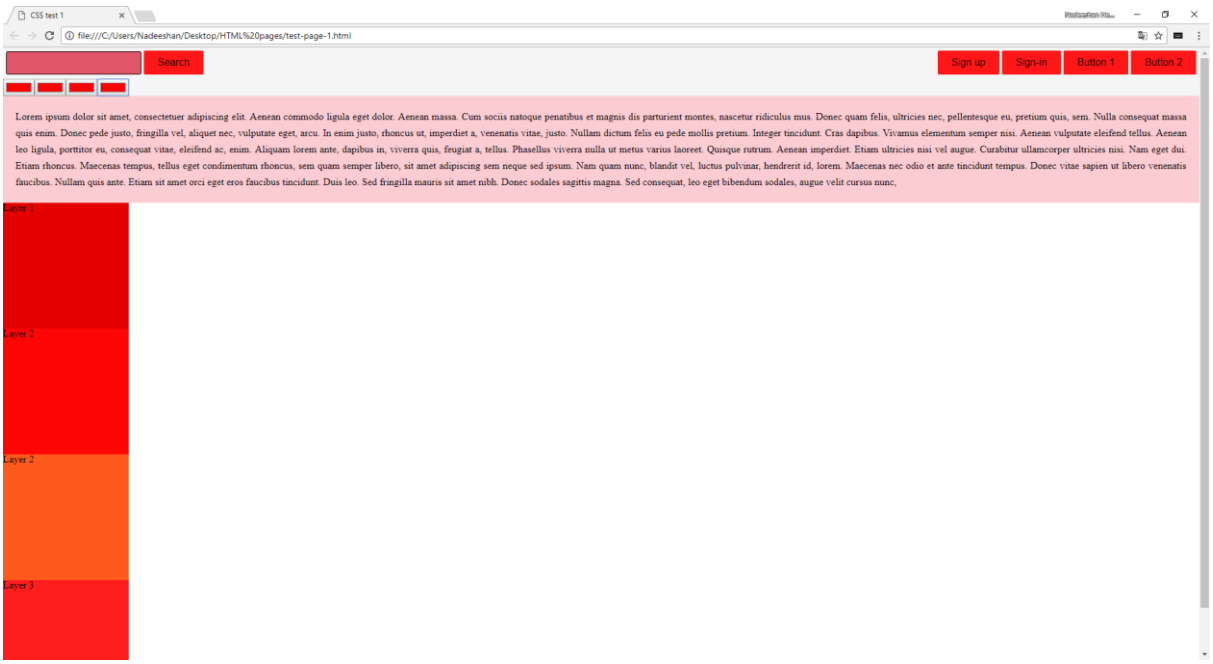


Figure 5.6 – Changing color to red in layer 1 of the HTML page

At this stage color of all the layers are changed to the red color. As we can see in the figure 5.6 all the colors and shades are successfully changed into red color.

Furthermore, users can select any color suitable for the HTML page. Following figure 5.7 to 5.11 shows the different colors which is applied to the above discussed HTML page. It should be notice that shades of the HTML page are not changed by the color change. Which was a key goal achieved with this framework.

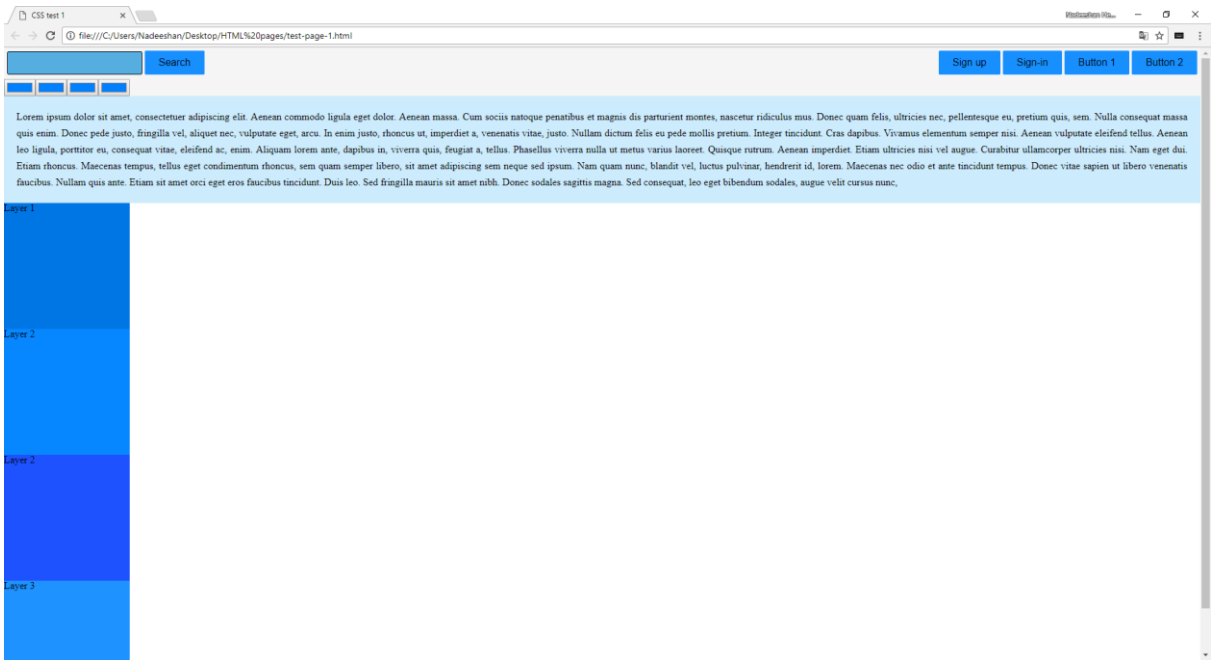


Figure 5.7 – Changing colors to blue on testing HTML page

Figure 5.8 – Changing colors to a purple shade on testing HTML page

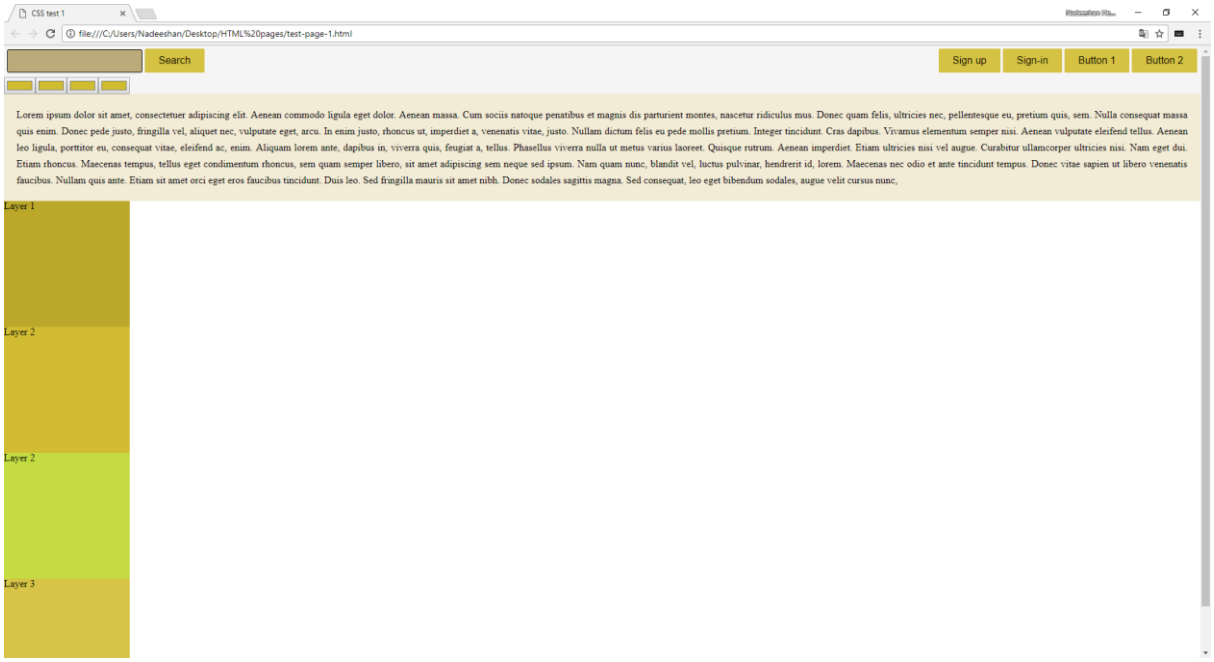


Figure 5.9 – Changing colors to a yellow-green shade on testing HTML page

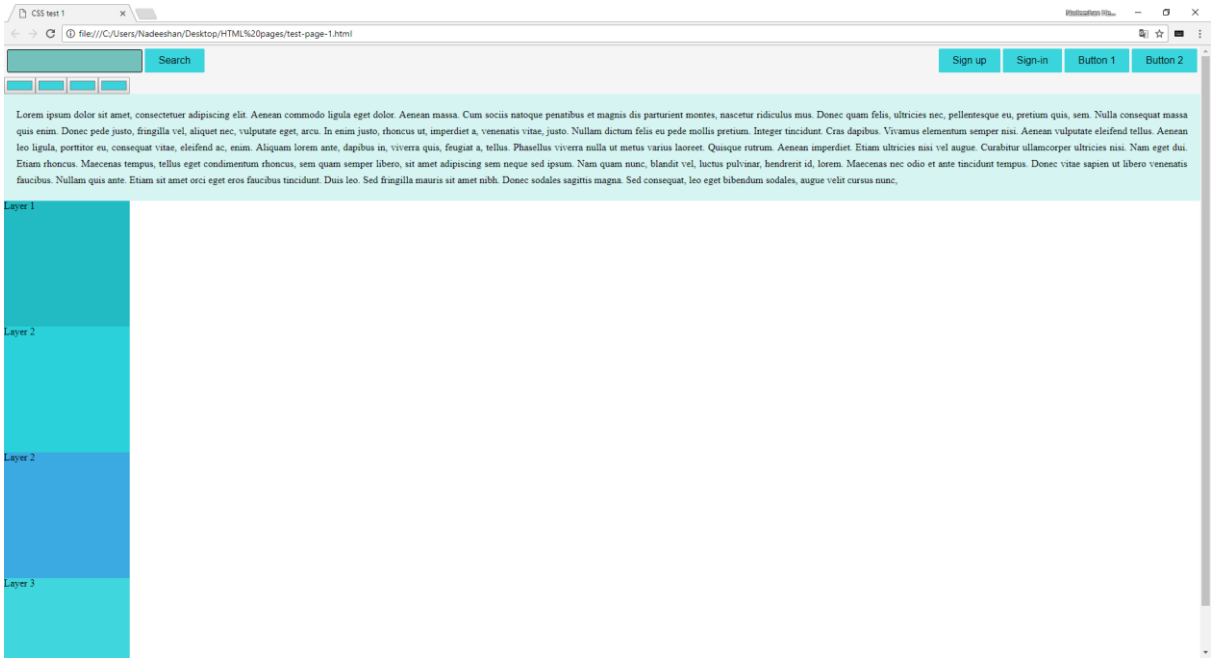


Figure 5.10 – Changing colors to a cyan shade on testing HTML page

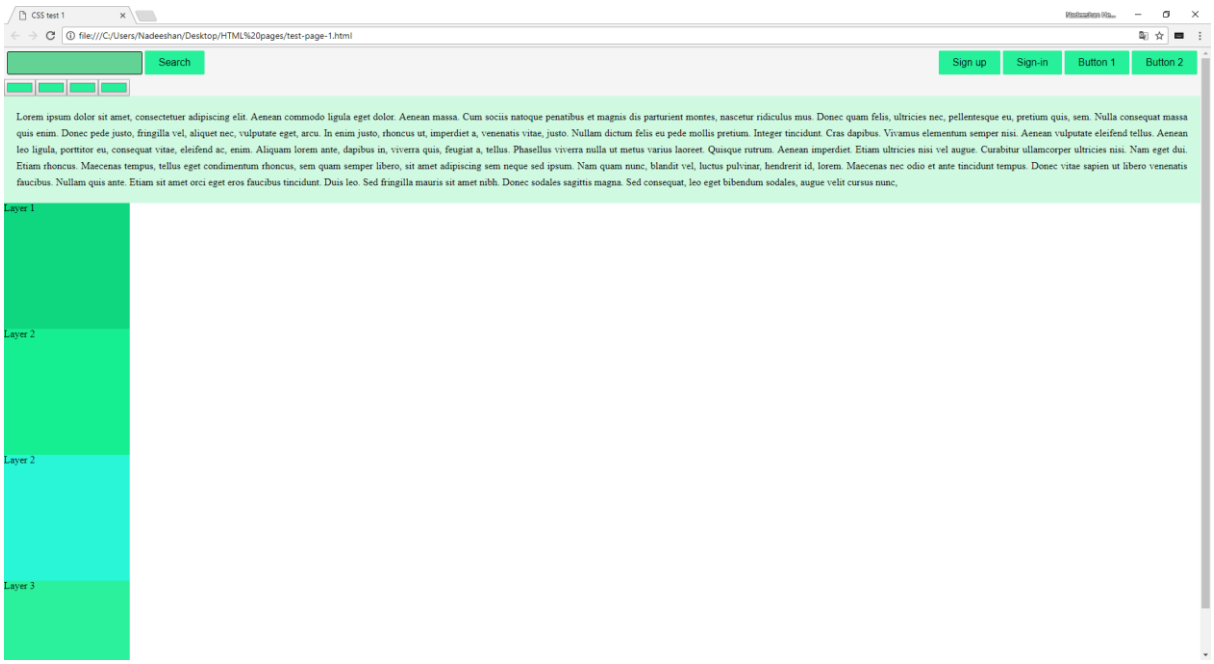


Figure 5.11 – Changing colors to a green shade on testing HTML page

As described in the implementation chapter, it is possible for the framework to change the colors of single layers separately. Figure 5.12 shows the color changed in the 1st layer into a different color. It should be noticed that color of the layer 1 is selected as a different color which may not match with other colors but only for the demonstration purposes.

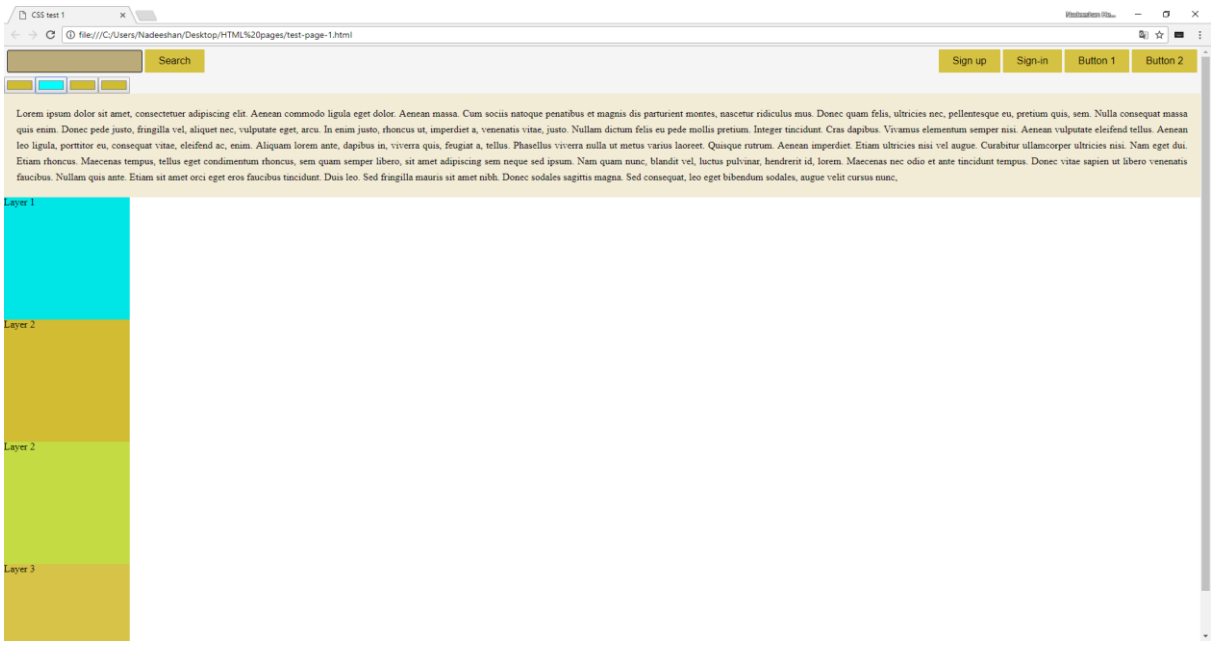


Figure 5.12 – Cyan color applied to the layer 1 of testing HTML page while other colors remains the same

5.2.1 Testing Framework with W3 CSS

A HTML document is created by using W3 CSS which is popular CSS framework on the web development world, to test the framework. We can see the framework is compatible and working with W3 CSS properly. Following is figures are some of the screenshots of the W3 CSS web document.

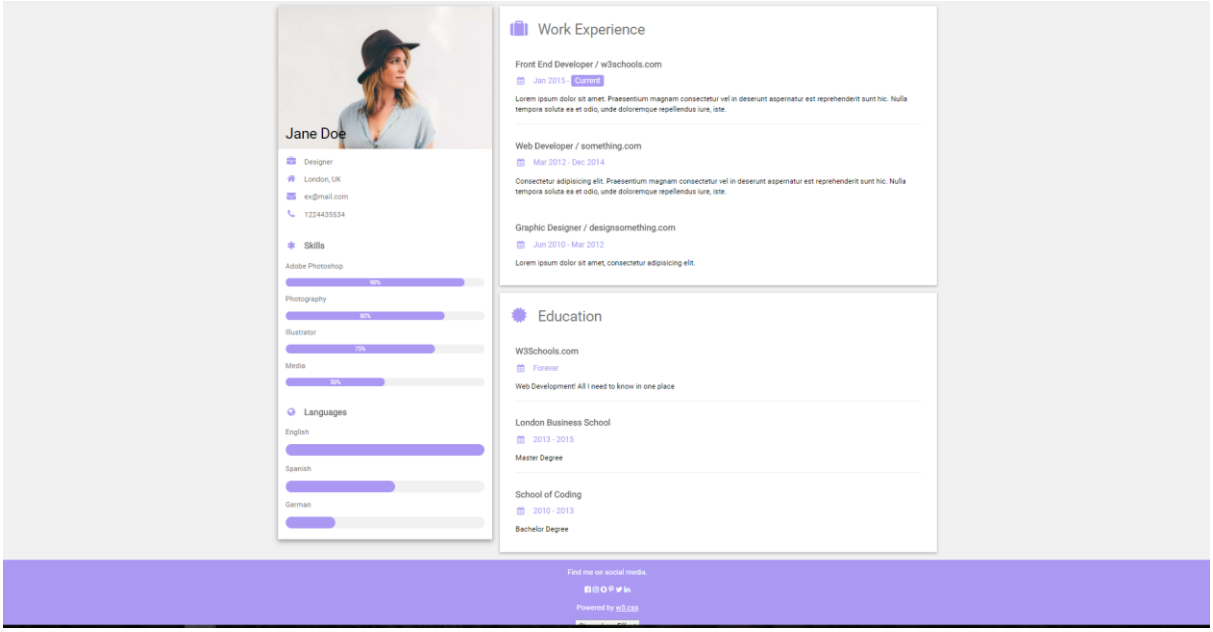


Figure 5.13 – W3 CSS HTML document, sample 1

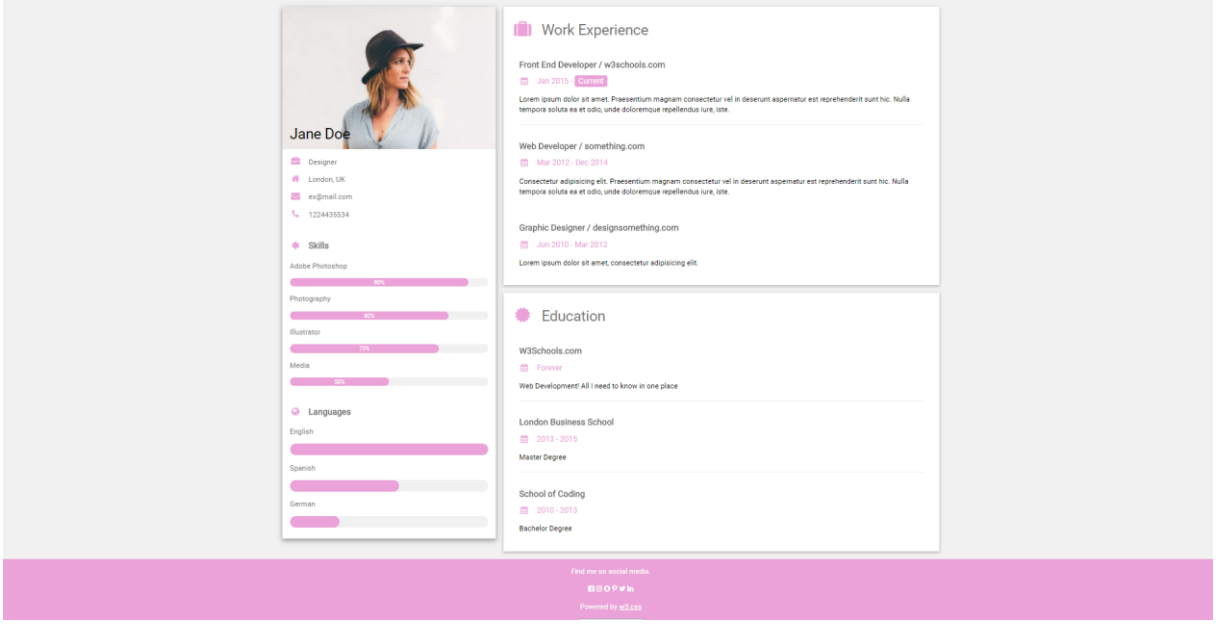


Figure 5.14 – W3 CSS HTML document, sample 2

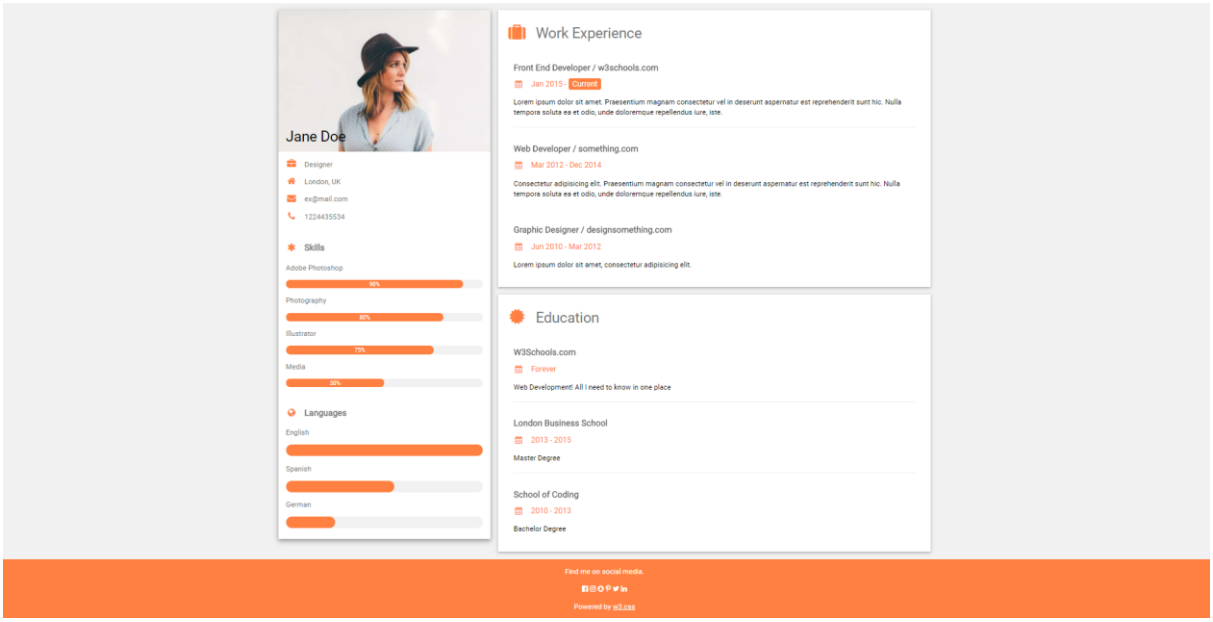


Figure 5.15 – W3 CSS HTML document, sample 3

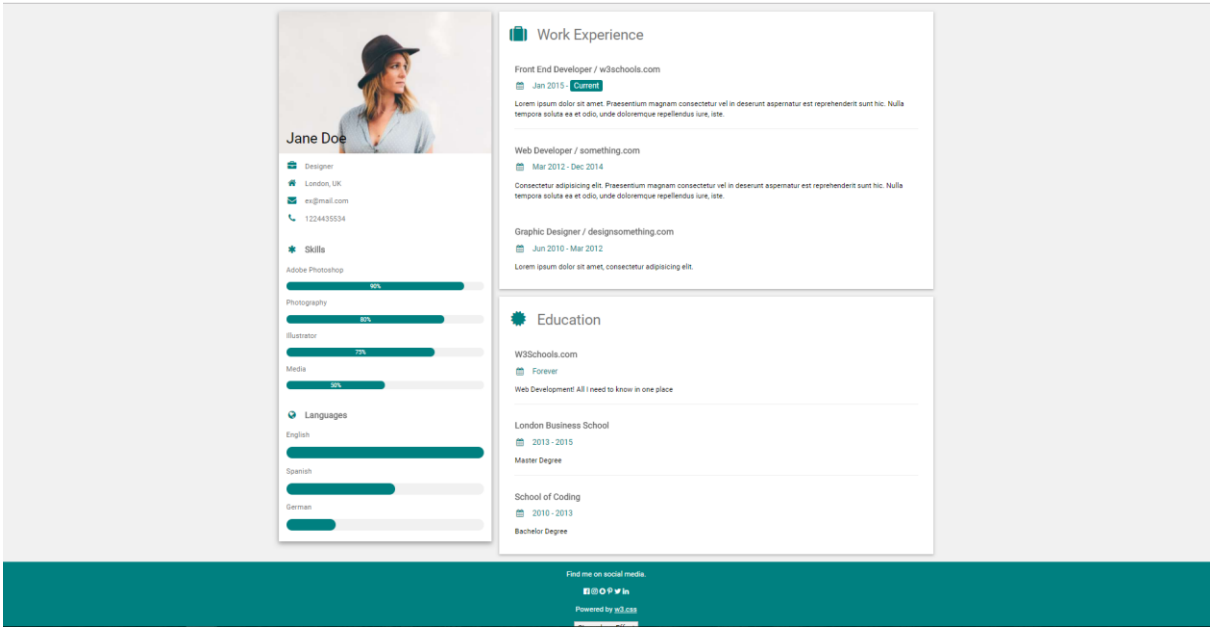


Figure 5.16 – W3 CSS HTML document, sample 4

5.2.2 Testing Framework with Native CSS and jQuery

Following figures are a document from native CSS with JavaScript and jQuery. Framework was integrated into this document for the testing. We can see the framework is compatible with web document and working properly.

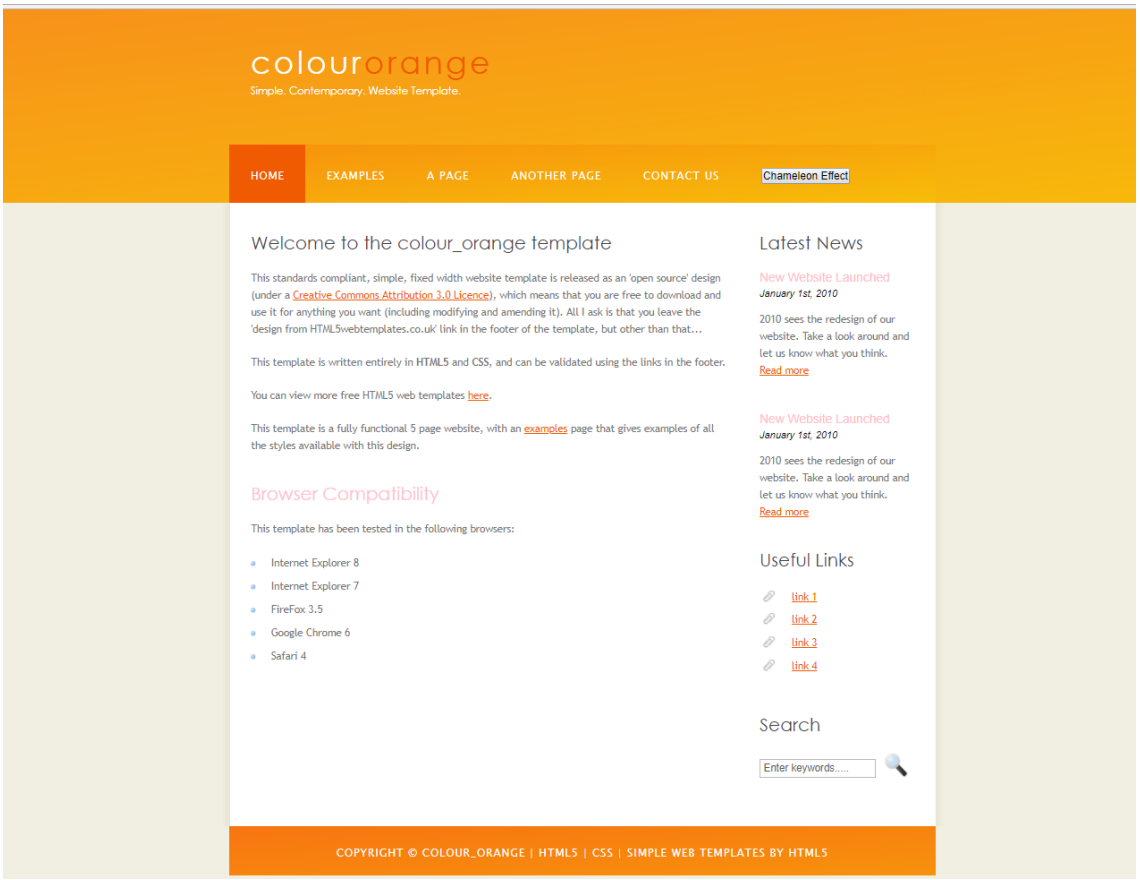


Figure 5.17 – Native CSS with jQuery - HTML document, sample 1

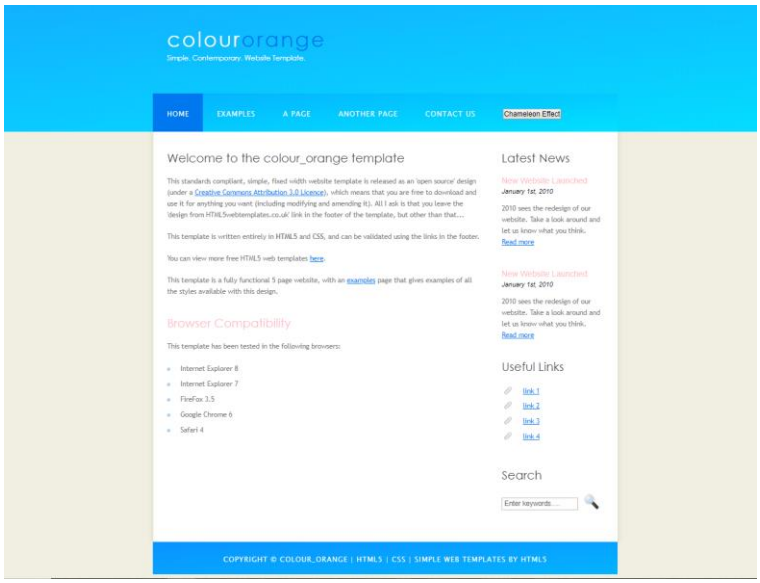


Figure 5.18 – Native CSS with jQuery - HTML document, sample 2

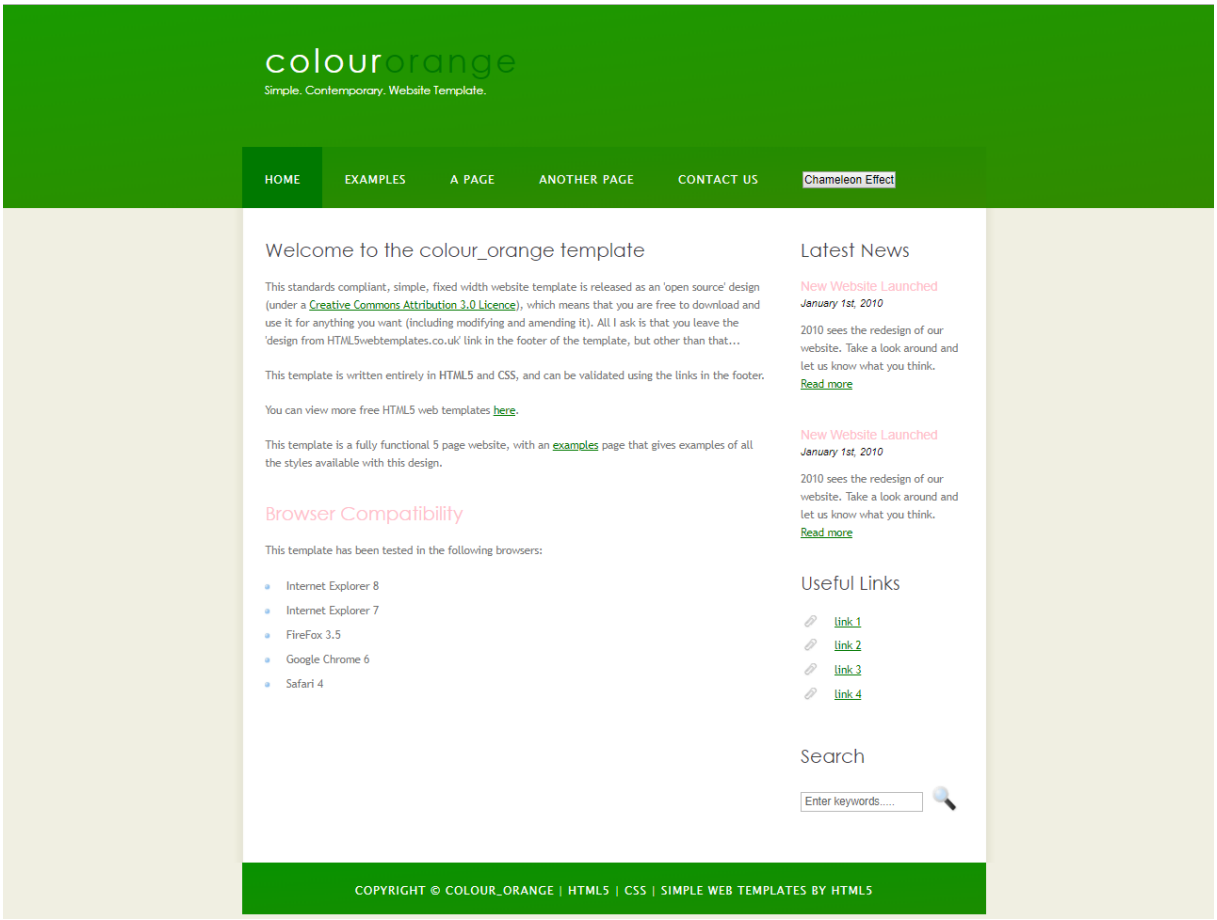


Figure 5.19 – Native CSS with jQuery - HTML document, sample 3

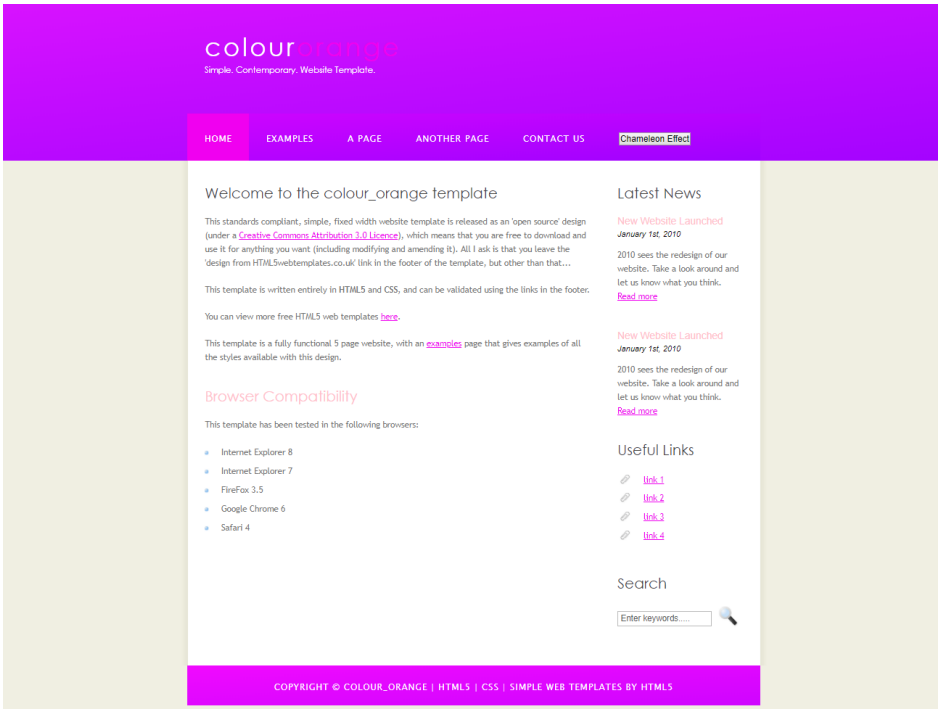


Figure 5.20 – Native CSS with jQuery - HTML document, sample 4

5.3 Testing the cross-browser compatibility with several standard web browsers

It is really important for a web application to be cross browser compatible for keeping their wide range of users who use different browsers worldwide. Best way to achieve cross browser compatibility is to use standard functions supported by standard JavaScript and use them in the correct way. This was another major goal in our work and our work was able to achieve it successfully. Figure 5.21 to figure 5.22 shows our testing HTML page working in several standard browsers available today. Furthermore, since we use standard methods to achieve dynamic color theming it is assumed to work in other standard browsers which are not mentioned here as well.

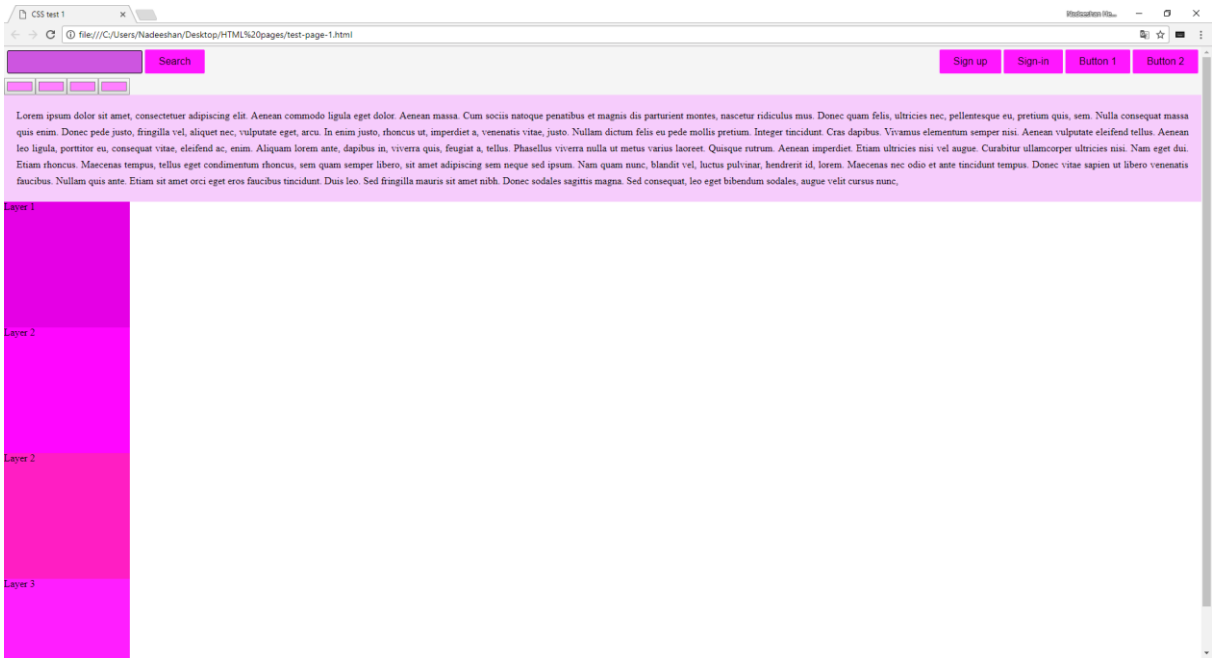


Figure 5.21 – Testing HTML page in Google Chrome

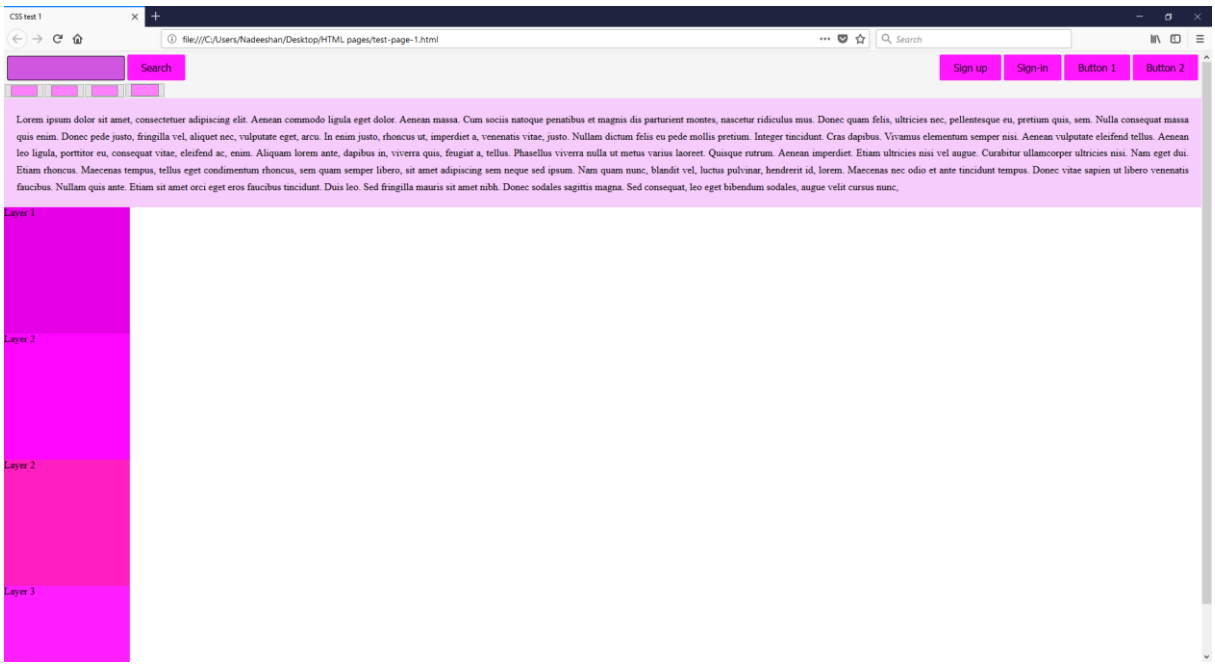


Figure 5.22 – Testing HTML page in Firefox browser

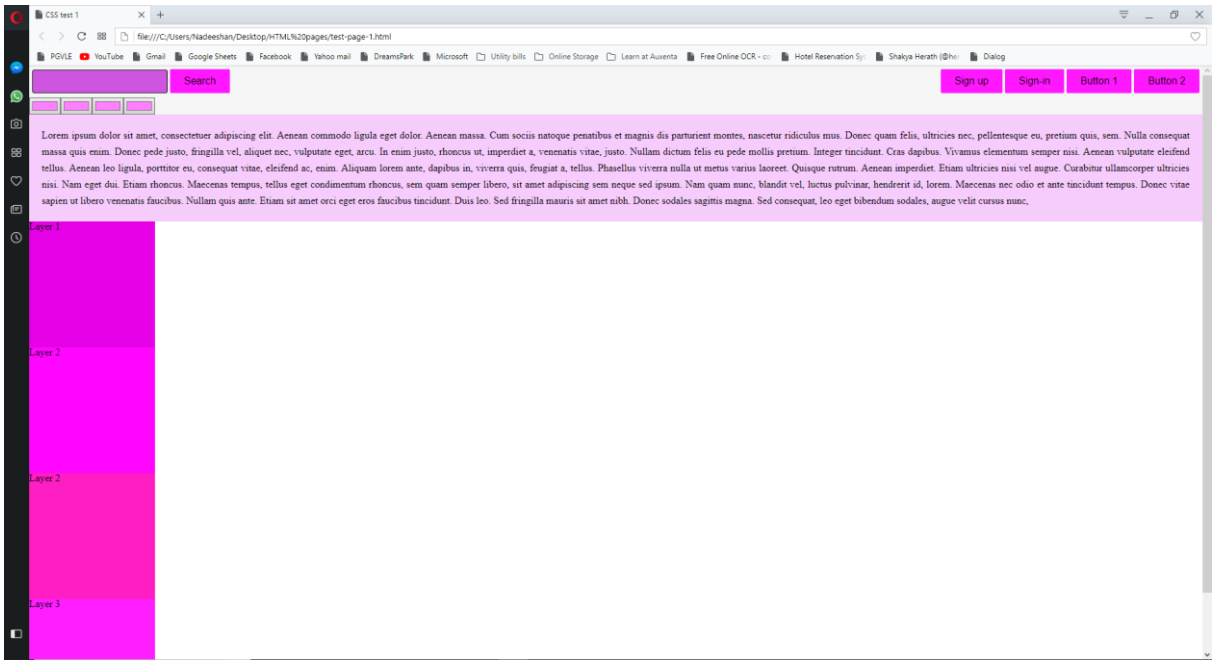


Figure 5.23 – Testing HTML page Opera browser

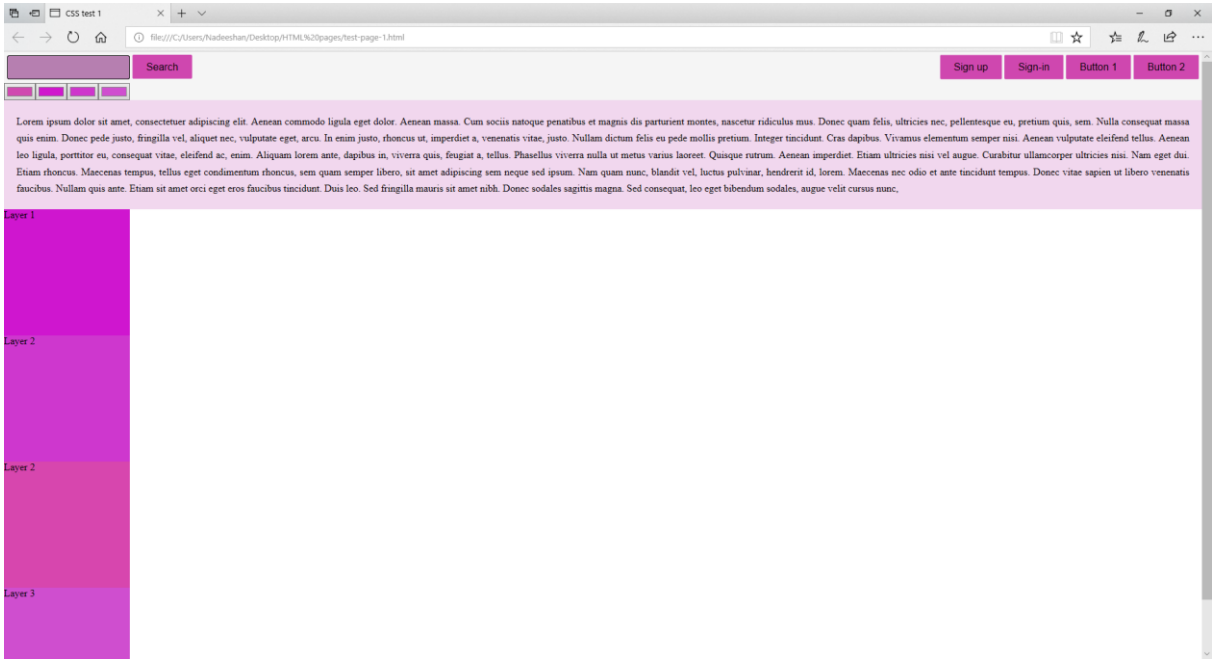


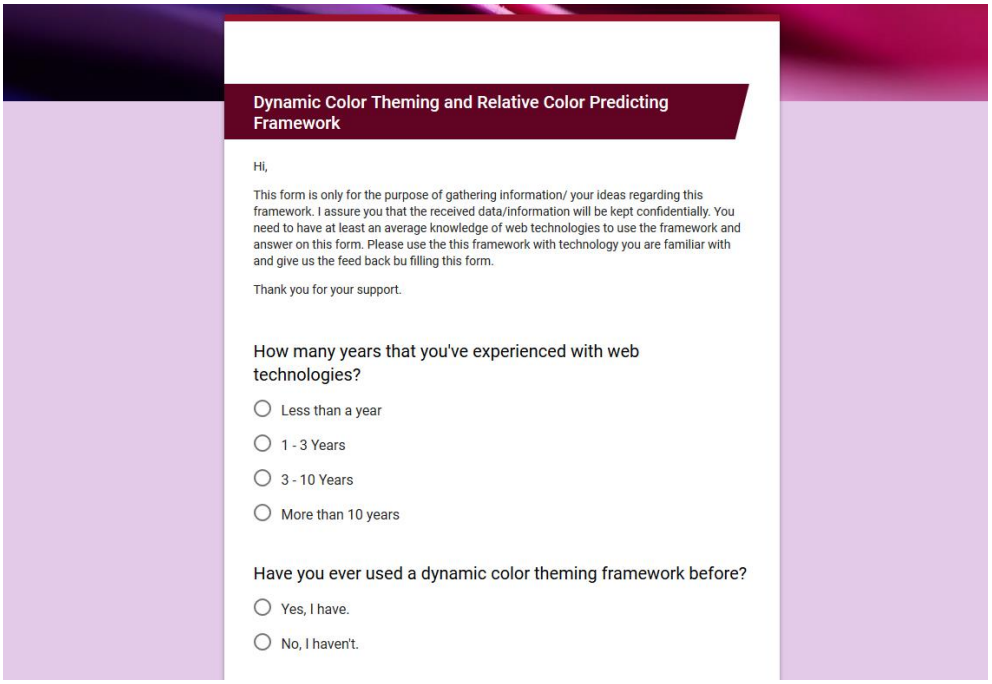
Figure 5.24 – Testing HTML page in Edge browser

5.4 Survey with experienced web developers to get their feedback.

To measure the integrality and usably of this framework there is no better way than getting user feedback. Hence a questionnaire is being created and given to few users who are experienced web developers to use the framework and give a feedback. To give the questionnaire a google form is used with few important questions. Figure 5.25 to figure 5.27 shows the google form which is used to collect the user feedback. Google from can be found in the following link by using open internet.

https://docs.google.com/forms/d/e/1FAIpQLSctVCMWWOmlwndjZawPID8zbLihVpNo9TzgZlIJhXMeNBCWrQ/viewform?entry.1702728825

(Link to the Google form with the questionnaire)



Dynamic Color Theming and Relative Color Predicting Framework

Hi,

This form is only for the purpose of gathering information/ your ideas regarding this framework. I assure you that the received data/information will be kept confidentially. You need to have at least an average knowledge of web technologies to use the framework and answer on this form. Please use the this framework with technology you are familiar with and give us the feed back bu filling this form.

Thank you for your support.

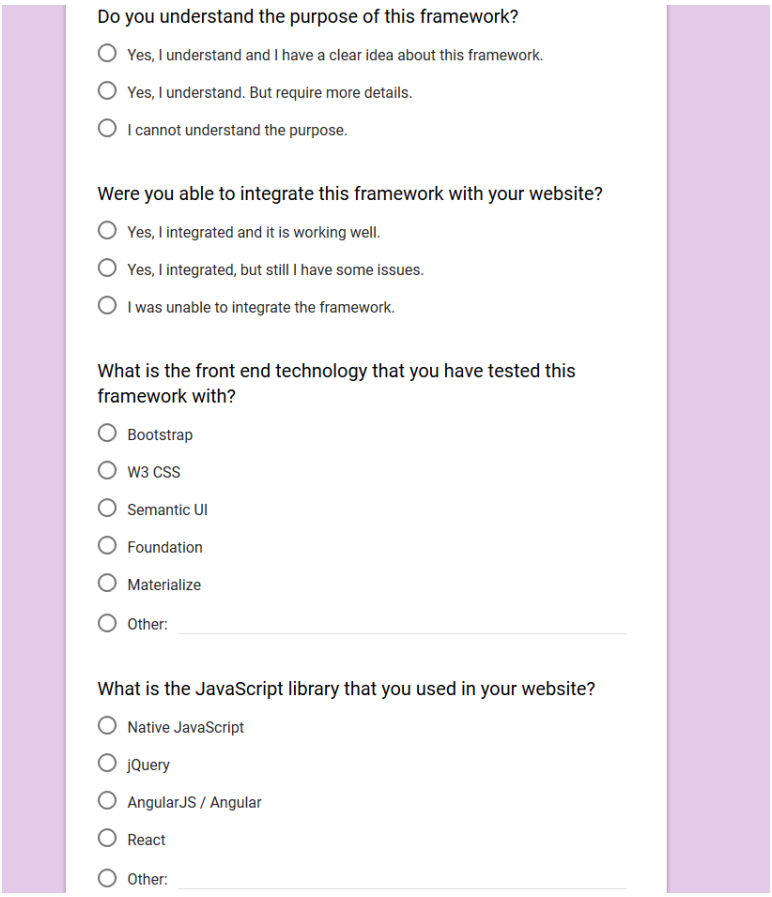
How many years that you've experienced with web technologies?

- ○ Less than a year
- ○ 1 - 3 Years
- ○ 3 - 10 Years
- ○ More than 10 years

Have you ever used a dynamic color theming framework before?

- ○ Yes, I have.
- ○ No, I haven't.

Figure 5.25 – Feedback form part 1



Do you understand the purpose of this framework?

- ○ Yes, I understand and I have a clear idea about this framework.
- ○ Yes, I understand. But require more details.
- ○ I cannot understand the purpose.

Were you able to integrate this framework with your website?

- ○ Yes, I integrated and it is working well.
- ○ Yes, I integrated, but still I have some issues.
- ○ I was unable to integrate the framework.

What is the front end technology that you have tested this framework with?

- ○ Bootstrap
- ○ W3 CSS
- ○ Semantic UI
- ○ Foundation
- ○ Materialize
- ○ Other: _____

What is the JavaScript library that you used in your website?

- ○ Native JavaScript
- ○ jQuery
- ○ AngularJS / Angular
- ○ React
- ○ Other: _____

Figure 5.26 – Feedback form part 2

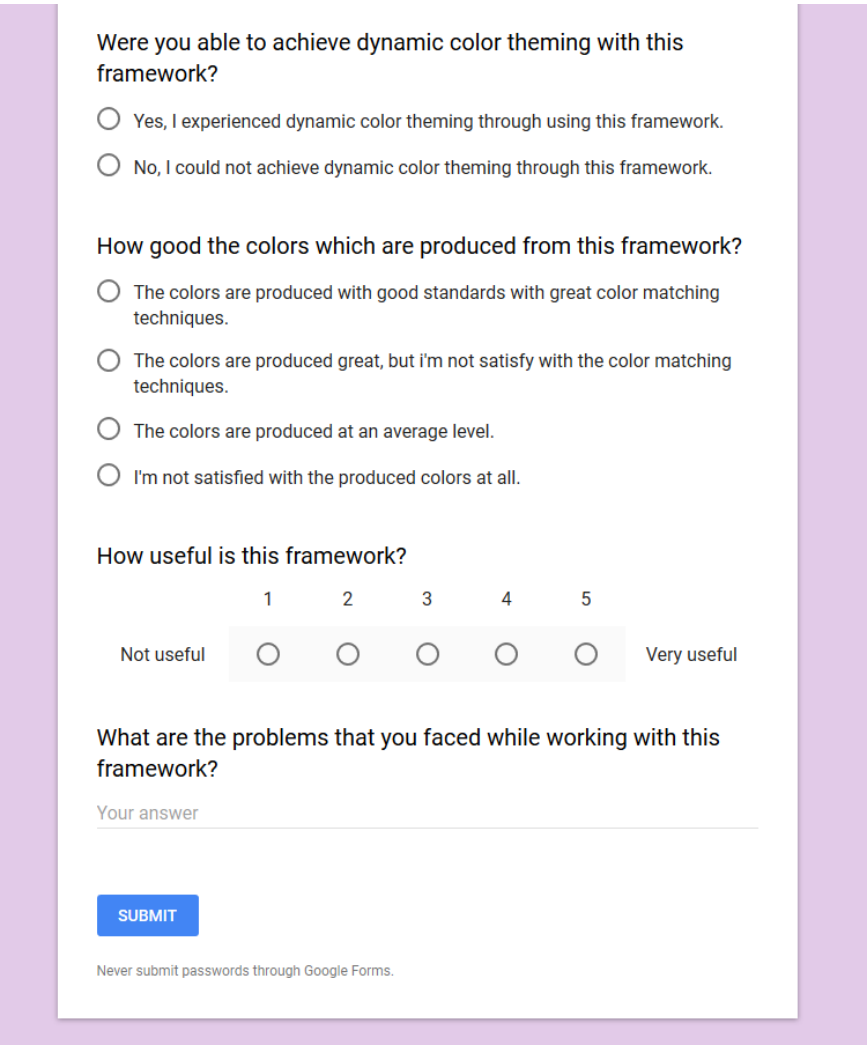


Figure 5.27 – Feedback form part 3

Following is the responses from the users for this framework. Their ideas and problems they faced will be discussed in the next section of this report. Figure 5.28 – figure 5.31 shows the summary of the responses of the users.

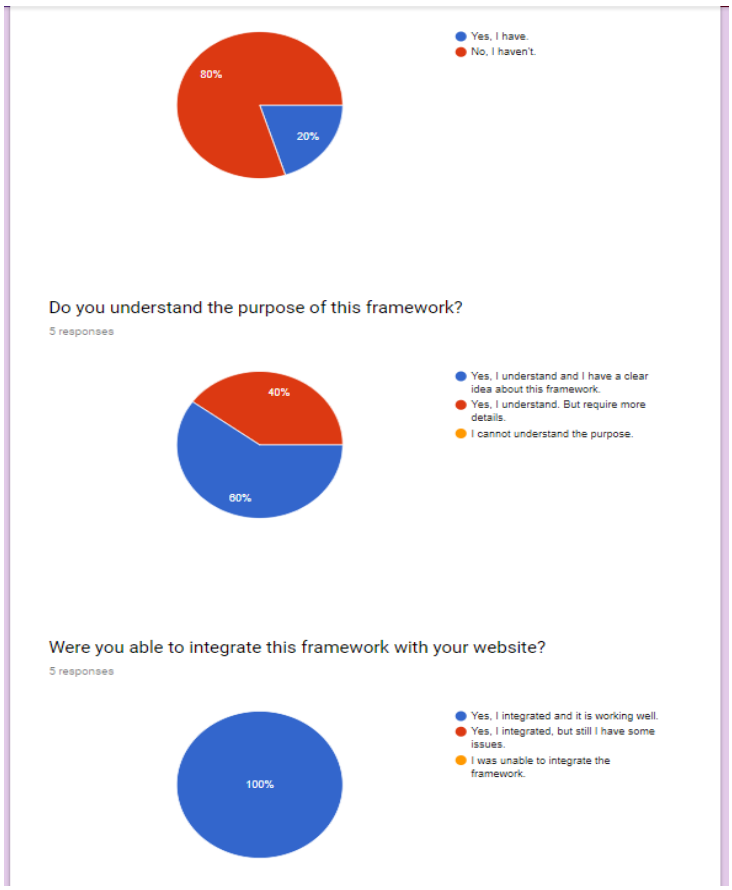


Figure 5.28 – Feedback from the users part 1

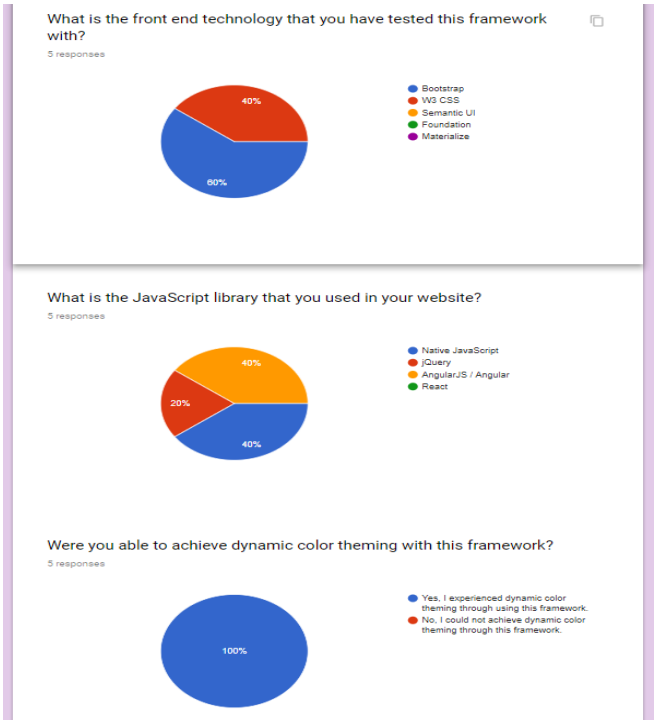


Figure 5.29 – Feedback from the users part 2

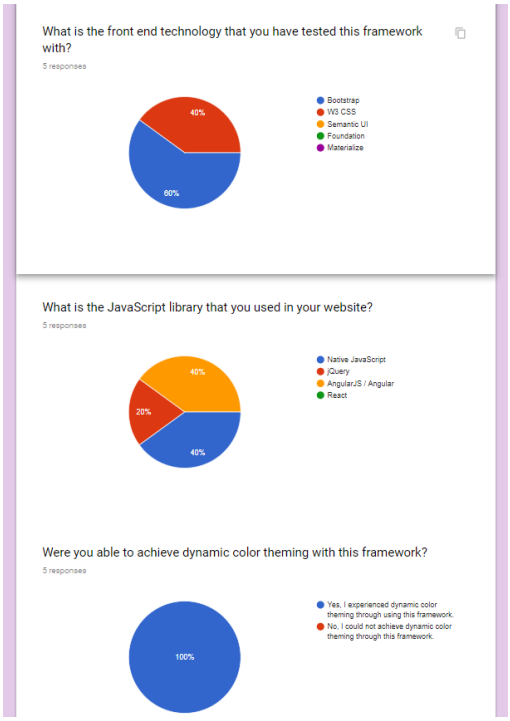


Figure 5.30 – Feedback from the users part 3

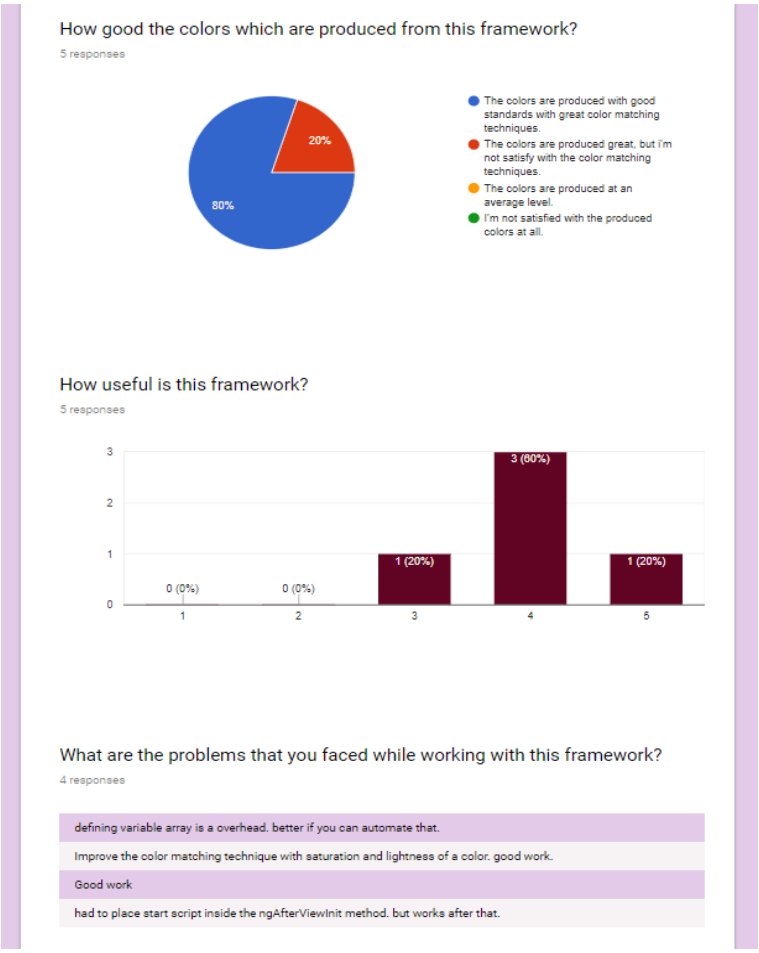


Figure 5.31 – Feedback from the users part 4

5.5 Evaluating the used approach and problems faced

Approach to implement this framework and algorithm would be experimental based. There are few stages of this work as follows.

1. Research for a way to define colors relatively.
2. Creating new colors with the approach selected from the color relativity schema.
3. Creating functions and a standard way to apply the colors relatively for any standard HTML page.
4. Apply the work to a real HTML test page.
5. Testing the accuracy of the functions and fine-tune the application for several major color variances.
6. Restructure and build the initial framework and its behavior.
7. Test the framework over standard technologies.
8. Finalize the Framework and testing templates.
9. Creating the user guide and carryout a survey as a questionnaire to get user feedback.
10. Evaluating the user feedback and decide the future work.

5.5.1 Problems and solutions

1. Defining colors relatively was the major problem in the first phase. After initializing a way to define colors with RGB color schema and apply with the test pages, there were more issues caused because of that relativity schema.

2. Initial understanding was colors are made from RGB and it will be possible to define them relatively. But later when changing the colors relative to another color, it was realized that the color variance is very limited and colors that are predicting with the algorithm was not in the visible color range. Solution to that was not direct and I had to further study about the color theory.

3. After learning about the color theory and other aspects of color definitions I decided to use HSL color schema and to find a way to define the colors relatively in a way that is changeable with many color variances.

4. With HSL there is a way for the relativity of colors.

5. Further I wrote a function that could change the colors of a HTML page dynamically and I could cover for a large color variance as possible. Any color variance is now possible with the algorithm.

6. Still there is an issue with lightness in the algorithm. Even though this supports for any color schema reducing the lightness of the color makes no difference for the output.
7. Later I realized this happens due to several reasons in the algorithm. Current approach is focused with the colors and not with the lightness. If we take lightness change to the algorithm, then user expectation with their UI color schema will change. This happens due to several properties in the colors.

8. As a simple explanation colors are made from one or two main elements in HSL schema. Either color will be based on hue and saturation, or it will be made from saturation and lightness. When saturation or lightness is relatively very high then the hue can be ignored since it makes no difference in the final color.

9. Currently I have decided to draw a graph and identify the breakeven points of saturation and lightness. This work is in the progress with testing.

5.6. Data sets for evaluation

Dynamic color theming framework is technically supposed to change any human visible color in the planet. Computer and other standard screens can display 16.8 million difference colors. Here this work will use 16.8 million colors as input and this will be the dataset of the work.

5.7 Experimental research work and benchmarks

Bench mark of this research work will be the breakeven points of the algorithm for saturation and lightness. The level of this is still under experiment and I am in the stage of fine-tuning saturation and lightness for better results in the framework.

Chapter 6

6.1 Conclusion and Future Work

This chapter describes the conclusion of this research. Drawbacks, limitations and future improvements that is identified during the work is mentioned here. It is important that the future works to be detailed in this chapter since it is possible to improve the solution without repeating the same work again.

The purpose of this research was to find a way to change colors of the HTML user interfaces, in a dynamic manner even after creating them. Initial design of a HTML page is created carefully by using the matching colors and other aspects of the HTML web pages. We use this initial color matrix of the page prudently and change them with our framework.

This work achieves the main goal of research and was able to achieve sub goals as well. While this purposed mechanism handles extensible complexity and performing extraordinarily this mechanism has its own limitations too. This chapter is to mention the possible future works, improvements and laminations identifies throughout the work. This chapter will be describing the conclusions, limitations of the framework and the future work that has been identified.

Static design of HTML page limits the design aspects of the website some extent and it may limit the human interaction for specific audiences. This work is to reduce that limitation with respect to color matchings of the web pages. This limitation can be reduced with a dynamic color theming option for the HTML pages. To give a dynamic color theming ability for the HTML page there this work purposes a solution, which is the work carried out as a dynamic color theming framework for CSS and relative color predicting algorithm. This work achieves the state of relative color theming.

As mentioned in above chapters purposed solution successfully achieved the relative color theming state in HTML documents. This was technologically feasible with the introduction of CSS variables from the W3C organization. Today all the standard browsers support for CSS variables and hence this work is production ready for any application with confidence.

Before mentioning the technical aspects of the solution here, it is important to mention about the relative color predicting algorithm which is used underneath of this framework. This relative color predicting algorithm is written by taking the very basic aspects of the natural color and the way we see it in the computer screens. RGB and HSL which are in-detail explained in the introduction and background chapters, we use HSL color scheme to define a color relative to another color. We have used this mechanism to by doing many tastings but the conclusion is that it is very easy to deduct colors into many parts with HSL color schema than the RGB color scheme. Basically, in relative color predicting algorithm color is deducted into hue, saturation and lightness. Color predicting algorithm clearly deducts the color into few parts and compare them to guess or calculate the new colors. This calculation is described in the system design chapter.

The relationship of initial colors is taken into consideration when calculating the new color. This method is very efficient in calculation since it has only few mathematical additions and subtractions but give the good outputs to the HTML pages. This method has its own limitations but enough for our framework to guess the colors for the HTML pages. Limitations will be discussed immediately in the next section of this document.

By using this algorithm, a framework was built using native CSS and JavaScript. This framework is named as “Chameleon-JS” since it logically issues an ability for a HTML page to show color changing effects as a chameleon in the nature. One should use this framework in the development time to integrate this into their HTML pages.

This framework is written with very small and simple CSS code lines and any person with small CSS knowledge even can change the control panel look and feel as they need in their respective applications.

Persistency of the outputs and states should be handled by the developer but the framework itself supports for outputting the JSON format text file as a download, and users can restore the color settings by uploading the JSON formatted file at any moment.

Initial understanding was to develop this framework with a rule set to eliminate or prevent possible color conflicts which may occur with this framework. But later with experiments of the work, it is realized that possible color conflictions cannot be prevented by implementing a rule set. But it was feasible with layers approach of the HTML page. The approach to implement this was to make the framework applicable to the HTML pages with few stages called layers. Developers should do this layering in the time of applying this framework to their respective web applications.

Colors of the layers can be controlled independently and users can see the colors right away in the HTML page and identify the conflictions easily. They can use the best suitable colors as they wish for these layers but with care for the possible color conflictions.

Framework is written in native CSS and JavaScript as mentioned above and it is possible for any web application to use this as any standard web browser supports for standard JavaScript and CSS. No any other external library or framework is used as a support for this framework. Hence, we can introduce “Chameleon JS” as a browser independent cross platform framework where they can implement without the fear of compatibility issues. Furthermore, this framework is tested with many standard browsers for the compatibility.

A survey is being conducted to check the easiness of integrating this framework. Few developers are asked to use this framework and their ideas and feedback was taken by using this survey. Many ideas that they gave are taken into consideration and most of the changes are applied lately into the framework.

This framework should be distributed to the users who need it. And the mechanism was to create a web site called “chameleonjs.xyz” users can visit there get more intimation and improve their design with Chameleon JS. Usage of the framework and other relevant information is there in the website available for users.

Research objectives were successfully achieved with the limitation for some objectives. Below is a small description to recall can reconsider the objectives.

Objective	Status	Limitations and Satisfaction	Achieved or Not
Research for a methodology to define one color relative to another color.	Done with HSL color scheme.	Lightness definition was not straight forward but handled that using a control panel given for the users to change lightness manually.	Achieved.
Find a suitable and effective method to change the color properties of the HTML documents on the fly.	Done with CSS variables.	Only Standard browsers support for the CSS variables. Usage of browser version older then 2 years may not be possible.	Achieved
Research for a methodology to predict new colors relative to another color.	Done with HSL color scheme	With a good satisfactory level.	Achieved
Design a framework for users to use this work in their HTML pages.	Framework called Chameleon JS was created.	With a good satisfactory level.	Achieved
Possible color conflicts that can occur with the framework should be minimized or prevented.	Handled the color conflictions with layers approach	A control panel is given for the users and they can manually handle such situations if occur.	Achieved
Design the framework independent of other technologies, so any standard website can use this solution.	Done by only using native JavaScript and CSS	With a good satisfactory level.	Achieved

Table 6.1 Objectives of the Research and Conclusions

6.2 Limitations of the framework

Framework will be quite useful for developers who need the dynamic color changing ability in their HTML documents. The framework has its own limitations. But the design of the framework was trying to minimize the limitations as much as possible. It is important to mention the limitations

6.2.1 Color Conflict Handling Limitation

One limitation is when the users are working with lightness and saturation color mismatches or conflictions may occur. This will lead to destroy the initial color combinations and law readability of the HTML document. There are actions taken to prevent this but it is important to consider how they can occur.

When changing the colors initially we do not let the users to change the lightness of the color. We only consider hue and saturation. At this point color conflicts will not be a problem. But this limits the ability for users to change the colors they need irrespective of initial color lightness. Hence there is an option given to the users in the control panel as lightness and users can turn it on. Once they activate this mode, users are able to change the lightness of the color if that particular layer. Here when users are doing it color conflictions may occur. It is highly advices for yours to take a manual look into the document after a successful color change. If conflictions are there, then users can change the colors immediately.

However, the limitation is that the framework is not directly able to handle all the possible color conflictions automatically. But the mechanism forces the users to prevent such conflictions manually by giving them the control.

6.2.2. Color Definition Limitation

HTML documents can use a property called “rgba(x,x,x,y)” this property is able to give a opacity to the layer and define the color at the same time. With the technical limitations and

by taking the scope of this research into consideration this framework does not support to use this special element with color changes. Users can use this functionality and framework will not hold users using it but rather framework does not support for the dynamic color changes. However, this is a limitation in the framework.

6.2.3 Necessity to use CSS Variables

This framework is highly dependent on the CSS variables. The predicted colors are applied directly to the HTML elements by using CSS variables. Today all the standard browsers support CSS variables. However, there are old browsers, non-standard browsers which does not support CSS variables. If the browser is not supporting CSS variables, then any of the color will not be visible in the specific browser. This is a limitation for this framework.

It was initially understood this much of a tradeoff to achieve the highly valuable dynamic color changing ability for the HTML page. And as of today, for the standard applications this limitation does not effect.

6.2.4 Development time application

The framework is not able to use just by adding the CSS and JavaScript files into the HTML document. The developer has a sufficient amount of work to reassign the colors of the page into variables and declare them in the framework. Steps are clearly given to the users and they can follow it easily but still developers have to compromise their time and important material in the real development environments. This is a limitation of this framework since it does not offer 100% automated service.

6.3 Future Work.

- Reading the HTML DOM and changing the colors of the HTML elements by using this framework.

This framework currently works with defining the colors of HTML elements manually. But it is being realized that with some other mechanism it may be possible to read the HTML document's DOM element directly and get the CSS color definitions. These color definitions can be grouped in several sections and they can be used as the input for this framework.

This will help the framework to work more efficiently and framework will be more automated than a manual process. This will increase the developer satisfaction and accuracy of the work as well.

Since all the HTML definitions anyway go to the HTML DOM element it will not be a problem from cross technology applications as well. But currently this is hard to achieve to a certain level with the current technological background. However, this will not be infeasible if we can implement a mechanism to read DOM and subscribe for colors of the elements and use them with the framework.

- Improving the color predicting algorithm for saturation and lightness as well.

Current color predicting algorithm is able to do required work but can be improved by using artificial intelligence program. There are many aspects to be solved before starting this as a research but many more impressive results will be possible with a much more advance color predicting algorithm.

When the algorithm becomes more powerful there well be many aspects that can be covered by this concept of dynamic color changing and relative color predicting.

- Creating a tool to convert CSS or HTML pages in to CSS variables and usable with this framework for user friendliness.

This framework uses a manual method which is handled by the developers to change the color names and order them into an array. Instructions are given to the developers but there is a trade off with the time they have to spend for this. In future if we can

create a converter to read the HTML pages. Convert the relevant colors into variables and output the finalized HTML document and the JavaScript file the work will be automated.

This will be grate for developers since their development time can be reduced to a considerable level. However, there may be limitations with this approach as well due to many technologies uses many syntaxes in the HTML pages. There will be validation limitations in this mechanism.

- Testing this with large websites and suggest the persistency mechanism by the framework.

  This framework is tested with many technologies and browsers. This was tested with many websites too. But still it would be good to test this framework with large website which can handle more than 20000 requests per second then we can get an idea about the performance of this framework this can be done in the future in a real web application whenever the chance is there to test.

- Using an artificial intelligence tool to identify color conflictions and improve accuracy of the framework.
  To handle the color conflictions, the framework uses a manual procedure. But if we can train neural network for this suppose and use it with this application then it would be able to prevent the color conflictions in more accurate way.

  It will not be easy for this framework to work with a neural network or any other artificial intelligent application but in the future if we can implement a way to connect that then it can produce more effective color conflicting mechanism for this framework.

References.

[1]. W3C HTML 4.0 specification (W3C Recommendation) by Dave Raggett, Arnaud Le Hors, Ian Jacobs. Page number 19. -at : http://www.w3.org/TR/1998/REC-html40-19980424

[2]. Document Object Model (DOM) Level 2 HTML Specification Version 1.0 W3C Recommendation 09 January 2003; Page 1. at : https://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/DOM2-HTML.pdf

[3] W3C, CSS, at : http://www.w3.org/Style/CSS/

[4] Accessibility features of CSS, at : https://www.w3.org/TR/CSS-access

[5] Mozilla Developer Network. Web developer survey research. Technical report, Mozilla at : https://developer.mozilla.org/en-US/docs/Mozilla/Mozilla_Web_Developer_Community

[6] Web Technology Survey. Use of CSS for websites. at :https://w3techs.com/technologies/details/ce-css/all/all

[7] CSS Custom Properties for Cascading Variables Module Level 1 at : https://www.w3.org/TR/css-variables-1/

[8] CSS variables and the specification at : https://www.w3.org/TR/css-variables/

[9] CSS variable specification at : https://www.w3.org/TR/2015/CR-css-variables-1-20151203/

[10] On the Analysis of Cascading Style Sheets, Pierre Genevès CNRS pierre.geneves@inria.fr, Nabil Layaïda INRIA nabil.layaida@inria.fr, Vincent Quint INRIA vincent.quint@inria.fr

[11] P. M. Marden and E. V. Munson. Today's style sheet standards: the great vision blinded. Computer, 32(11):123–125, nov 1999.

[12] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (Page 35), at : https://www.w3.org/TR/2011/REC-CSS2-20110607/

[13] Constraint Cascading Style Sheets for the Web , by Greg J. Badros, Alan Borning, Kim Marriott, Peter Stuckey at : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.3055&rep=rep1&type=pdf

[14] Automated Analysis of CSS Rules to Support Style Maintenance by Ali Mesbah, University of British Columbia Canada. And Shabnam Mirshokraie, University of British Columbia Canada. at : http://www.ece.ubc.ca/~shabnamm/doc/css-icse12.pdf

[15] W3C, "CSS," at : http://www.w3.org/Style/CSS/ as at 25th March 2018

[16] Adobe Color wheel at : https://color.adobe.com/create/color-wheel/ as at 25th March 2018

[17] psychology of color at : http://www.moosepeterson.com/techtips/color.html as at 25th March 2018

[18] visible spectrum at : https://en.wikipedia.org/wiki/Visible_spectrum as at 25th March 2018

[19] Colors in the world : http://www.rit-mcsl.org/fairchild/WhyIsColor/files/ExamplePage.pdf as at 25th March 2018

[20] JavaScript Standard at : http://www.ecma-international.org/ecma-262/7.0/index.html as at 25th March 2018

[21] CSS specification of colors at : https://www.w3.org/TR/css-color-3/ as at 25th March 2018

[22] HSL color scheme at :https://nixsensor.com/what-is-hsl-color/ as at 25th March 2018

Appendix A – Integration Manual Documentation.

Integration manual will give the relevant information for the framework to be integrated with any HTML document.

Chameleon JS – Integration Manual

Welcome to Chameleon JS. If you need to have dynamic color theming for your web page you are here in the right place. First get an understanding on how this amazing feature works and learn how easy it Is to integrate it to your own website.

Chameleon JS is designed to give your website the dynamic color changing ability. You can change the colors of your website on the fly, any time you want. There will be a control panel which the Chameleon give you to change the colors of your website. You can choose any color you need from 16.8 million different colors or you can mention RGB values or HSL values to input your favorite color to the framework. At any time, you can save your current color combinations of the page or initial color pattern of the page and reload it any time you need. For the technical details and other questions please see the FAQ section.

Framework uses a relative color predictive algorithm underneath to predict the most suitable colors for your website. Framework saves the initial color combination of your web page and uses that matrix to predict new colors. Hence the initial color matches and the details of the colors will not be vanished even after a color change. The Framework is written in native CSS and JavaScript and there are no external frameworks underneath. Hence it is guaranteed to work in with any application but CSS version should support the CSS variables.

Prerequisites for usage.

1. Interest to have dynamic color changing feature on your web site.
   If you have a requirement to do dynamic color theming in your website or may be to integrate the dynamic color theming and give it as a feature to your users it will be a grate hit since yours can choose their own interesting colors.
2. Web site should be running only on browsers which supports CSS variables.

Chameleon JS uses CSS variables to implement its dynamic color theming behavior. Hence your application will only be viewable with the browsers which supports CSS variables. CSS variables was there in the technological world for some time and hence every standard browser today supports CSS variables.

3. Website should have the support for CSS and JavaScript
   Every standard browser supports CSS and JavaScript. Hence this will be the least of your problems if you are doing a web application. Because you must be having an environment where it supports these technologies.

Usage of the Chameleon JS

1. Follow the steps to integrate the Chameleon JS into your favorite website.

2. Download Chameleon JS distributable bundle and extract the zip file. And include Chameleon.CSS file in your HTML document. 3 JavaScript files should be included at the end of the HTML document. Notice adding them in the end of file is necessary.

3. You can download the Chameleon JS with the link below or just visit to http://chameleonjs.xyz

4. In your download there is a sample HTML file which is working properly and you may see the CSS and JS folders which the actual framework is.
5. Direct download link: https://goo.gl/RmK7pN

```
<link href="CSS/chameleon.css" rel="stylesheet">

<script src="JS/chameleon.js"></script>
<script src="JS/chameleon-algorithm.js"></script>
<script src="JS/chameleon-popup.js"></script>
```

6. Replace all colors of your document with CSS variables and define them in the root element of CSS file. You can complete this by following below steps.

i. Replace the color of CSS style with a variable which follows the Chameleon naming convention. As an example, you will be replacing the value of white on background property with a CSS variable as below. Note that your color can be on a external CSS sheet internal definition or even inline definition.

```
background: var(--L3-white);
```

Then go to the Chameleon.CSS which is given on download bundle and open the file. You can define your variable here with your favorite color.

```
:root {
    --L3-white : #FFFFFF;
}
```

There are many ways to define color with this Chameleon JS as used in standard CSS. As an example, you can mention white color as any of the following.

- #FFFFFF
- #FFF
- White

(Chameleon naming convention is explained after this introduction.)

ii. Open the chameleon.js file and find the definition of “variablesArr” which is defined on the top of the file. Add your previously defined variable here as an array element. As an example, following array holds two CSS variables created as of in the 1st step given above. Notice you only add the variable name here.

```
var variablesArr = ["—L3-white", "--btn-hover-color"];
```

iii. Chameleon JS control panel is auto created on your HTML document. To give the space for this control panel you need to add following div element to your page. You may add any CSS definition to this div element to match with your website. Remember to place this div in a place where your users can identify it, so it will be easy for them to find the control panel.

```
<div id="chameleon-js-main"></div>
```

Chameleon JS uses a native popup mechanism which is supported by HTML5. Control panel will be visible there.

7. Open your HTML document and enjoy!

Chameleon Naming Convention

Variables defined in this framework follows a naming convention. Users of the framework should follow this naming convention in order to define the layer of the variable which they are referring to. Naming convention should follow the following rules in order to achieve the color relativity in HTL documents.

I. All the names should be declared as CSS variables and assigned to a color.

II. Names must be unique, but one unique variable can be used in many places in the HTML page, where users need to have the same color.

III. Name should start with "--" due to the standards in CSS variables.

IV. Names which needs to go to particular layer should mention the layer name starting with "L" followed by layer number. Ex: "--L1-my-background-color"

V. Variables which does not need to go to a particular layer can be declared without "L" but these variables will fall into the category of default layer which is the "0" layer.

Visit Chameleonjs.xyz for more details. Or email to herathnadeeshan@gmail.com for any assistance. Please leave a comment of your experience with Chameleon.JS

Appendix B – Description of functions in the framework.

Chameleon JS uses many functions written I JavaScript language. Some of the important functions will be shown here and described.

Color Relativity Generation.

This function takes three parameters as the “baseColor” which is the basic color defined on the original HTML page, “newMainColor” which is the newly selected main color that user is trying to change to and the “actualColor” which is the actually defined color of the current HTML document.

Function calculates few parameters which is really simple in JavaScript language, it is important to notice that inputs are running through two more functions called “hexToDecimal” which will convert the hexadecimal code to RGB numbers and “calculateHSL” function which will convert the RGB numbers to a HSL number array. This function returns a string which contains a HSL color definition ready to use for the HTML page.

```
function generateRelativeColorHSL(baseColor, newMainColor, actualColor){

    var base = calculateHSL(hexToDecimalArray(baseColor));
    var newMain = calculateHSL(hexToDecimalArray(newMainColor));
    var actual = calculateHSL(hexToDecimalArray(actualColor));

    var h = base[0]-actual[0];
    h = h + newMain[0];
    if(h <0 ){h = h+360};
    h = h%360;

    var s = (base[1]-actual[1]);
    s = s + newMain[1];
    if(s>100){
        s = 100;
    }

    if(s <0 ){
        s = 0;
    }

    var l = (base[2]-actual[2]);
    l = l + newMain[2];
    if(l>100){
        l=100;
    }

    if(l<0){
        l=0;
    }

    var color = 'hsl('+h+','+ s +'%,' + actual[2] +'%)';
    return color;
}
```

Appendix B, Figure 1 – Generate relative color HSL function. (color relativity algorithm)

Calculate HSL function

This function takes the RGB value as an array and converts the RGB color to the respective HSL colors after doing few validations to ensure the color is in the visible range. Finally it returns the HSL color as an array.

```
function calculateHSL(color)
{
    r = color[0];
    g = color[1];
    b = color[2];
    if( r=="" ) r=0;
    if( g=="" ) g=0;
    if( b=="" ) b=0;
    r = parseFloat(r);
    g = parseFloat(g);
    b = parseFloat(b);
    if( r<0 ) r=0;
    if( g<0 ) g=0;
    if( b<0 ) b=0;
    if( r>255 ) r=255;
    if( g>255 ) g=255;
    if( b>255 ) b=255;
    hex = r*65536+g*256+b;
    hex = hex.toString(16,6);
    len = hex.length;
    if( len<6 )
        for(i=0; i<6-len; i++)
            hex = '0'+hex;
    r/=255;
    g/=255;
    b/=255;
    M = Math.max(r,g,b);
    m = Math.min(r,g,b);
    d = M-m;
    if( d==0 ) h=0;
    else if( M==r ) h=((g-b)/d)%6;
    else if( M==g ) h=(b-r)/d+2;
    else h=(r-g)/d+4;
    h*=60;
    if( h<0 ) h+=360;
    l = (M+m)/2;
    if( d==0 )
        s = 0;
    else
        s = d/(1-Math.abs(2*l-1));
    s*=100;
    l*=100;
    return [h,s,l];
}
```

Appendix B, Figure 2 – Calculate HSL function