Front-End Development of ITB Parking Information System Prototype using SCSS

Joshua Atmadja / 13514098

Department of Computer Science/Informatics
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jalan Ganeca 10 Bandung 40132, Indonesia

13514098@std.stei.itb.ac.id

***Abstract*— Parking in ITB is one of major problems that most *civitas academica* of ITB have. According to the latest claims, the problem comes from uncontrolled parking lot. For improving the parking issues, an information system is needed to observe and control the lot. The prototype is needed so clients are able to feel the system before it is really launched. Outside from the system, SCSS as the tools to design and style the system is used to make a good interface and comfortable experience. With respect of CSS, SCSS begins the new style of making stylesheet for web pages.**

***Index Terms*—atomic design, CSS, front-end, prototype, SCSS**

I. Introduction

Parking is one of the needs of every owner of any vehicles especially car and motorcycle. In case of ITB, however, parking demand is getting high and not proportional with the parking supply. Every year, especially corresponding new students' enrollment term, the margin of demand is getting higher and scarcely lower. One of the problem is that parking system in ITB is not controlled well. There are still many cases happening such as unauthorized entry, overnight staying, authority misuse, and weak bureaucracy.

In this work, the author and some parties attempt to make such system that helps control the parking in ITB. We choose to make a web application. The author itself specifies to create the system by developing front-end of the web application. The project which the author accomplished is limited only up to user interface and user interaction to the system, i.e. high-fidelity prototype. Further development would be considered on the proceeding agreements between the author and the clients.

To develop the front-end, the author attempts to use SCSS (Sassy CSS) as the stylesheet. SCSS is the next generation of stylesheet which is used in DOMs (document object models) and web pages or HTML (Hypertext Markup Language) pages, and works atop Ruby programming language.

This paper work is organized into five sections. The first section is introduction. The second section will explain some study of literatures about SCSS and other related tools. In the third section, the author describes how the project is developed and how SCSS is used. The fourth describes the results of the project. Finally, in last section, the author states conclusion and suggest future works.

II. Literature Study

A. *SCSS*

Sass (Syntactically Awesome Style Sheets) has two syntaxes. The new main syntax is known as SCSS, a superset of CSS's syntax. This means that every valid CSS stylesheet is valid SCSS as well. SCSS files use the extension `.scss` [1]. However, Sass files use the extension `.sass`.

Initially, Sass was part of another preprocessor called Haml, designed and written by Ruby developers. Because of that, Sass stylesheets were using a Ruby-like syntax with no braces, no semi-colons, and a strict indentation [2].

The difference between Sass and SCSS is mostly the syntax. Sass is known as the indented syntax. Reference [1] states that it is intended for people who prefer conciseness over similarity to CSS. On the other hand, SCSS is a CSS-like syntax, which the author used for developing the project.

SCSS supports styling for DOMs or web pages in the same way as CSS does. It also supports selectors e.g. HTML `tag`, `.class`, and `#id` selectors; attributes e.g. `padding`, `margin`, `background-color`, etc.; size units e.g. `px`, `pt`, `em`, `rem`, etc.; and object's event e.g. `hover`, `active`, `visited`, etc.

There are also few major differences between SCSS and CSS itself. According to reference [3], the advantages of using SCSS over CSS are that we can create variables like color's hex code, nest the selectors easily, create functions, and even do math for calculating sizes such as width, height, margin, and padding.

SCSS files need to be compiled becoming CSS file to be linked to the HTML page. To compile SCSS, we must have Ruby installed to the working environment and SASS Ruby Gem. We may include Javascript library to help compiling linked SCSS files. On further and more advanced development, web developers using Ruby on Rails do not need to compile SCSS files by their own because the rails compiles those automatically.

B. *Bootstrap*

Bootstrap is a free, well-known open-source front-end web framework. It uses HTML, CSS, and JS altogether and is mainly focused for mobile-first design. Bootstrap was known as Twitter Blueprint in mid-2010 as it was originally created by Mark Otto and Jacob Thornton at Twitter [4].

Bootstrap creates responsive web design. It means that website using Bootstrap will scale its pages according to platforms where it is loaded. It uses twelve grid system and integrated media queries corresponding with device's width such as 360 px for most phones and 1280 px for most desktops.

To use Bootstrap, we simply link Bootstrap's CSS and JS files into the HTML pages. They also simply make one and more pages as usual with HTML and insert classes into respective tags.

C. *Atomic Design*

Atomic design is methodology for creating design systems. According to reference [5], there are five distinct levels in atomic design, that are atoms, molecules, organisms, templates, and pages. Atoms are the basic building blocks of matter. Applied to web interfaces, atoms are our HTML tags, such as a form label, an input, or a button. Molecules are groups of atoms bonded together and are the smallest fundamental units of a compound. For instances, a form label, input, or button are not too useful by themselves, but combine them together as a form and now they can actually do something together. Organisms are groups of molecules joined together to form a relatively complex, distinct section of an interface. Templates consist mostly of groups of organisms stitched together to form pages. Furthermore, pages are specific instances of templates. In analogy of Java programming language, templates are classes and pages are objects.

Atomic design can be implemented in CSS selectors, or we can call atomic CSS. Selectors have only one attribute. Here is the example of atomic CSS.

```
.w200{ width: 200px; }
.h75 { height: 75px; }
.mg10 { margin: 10px; }
.mg20 { margin: 20px; }
.pd8 { padding: 8px; }
```

To make a molecule, we simply add this class in HTML tag. Below is the example of creating molecule with atomic CSS.

```
<span class="w200 h75 mg10 pd8">Your
text goes here</span>
```

III. Methods

A. *Project Development*

This subsection explains about the project and how it is developed. The project which the author accomplished is called ITB Parking Information System. As introduced, the users of this system are gate keepers divided into several zones, security officers, and ITB directorate of facility and infrastructure. The system is built and implemented as a web application using PHP and MySQL for database. It is only developed up to the high-fidelity prototype including interfaces, interactions, and databases.

The development process approach for this project is waterfall model. This model is suitable due to the fact that the prototype does not take much time to develop and to implement. As the author has been only developing the interfaces, security and further integration are set aside. The figure below represents how the project is developed.

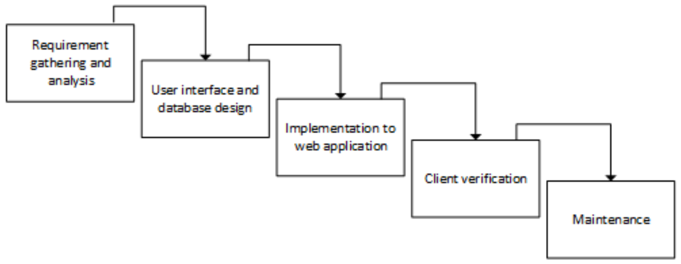

Requirement gathering and analysis
User interface and database design
Implementation to web application
Client verification
Maintenance

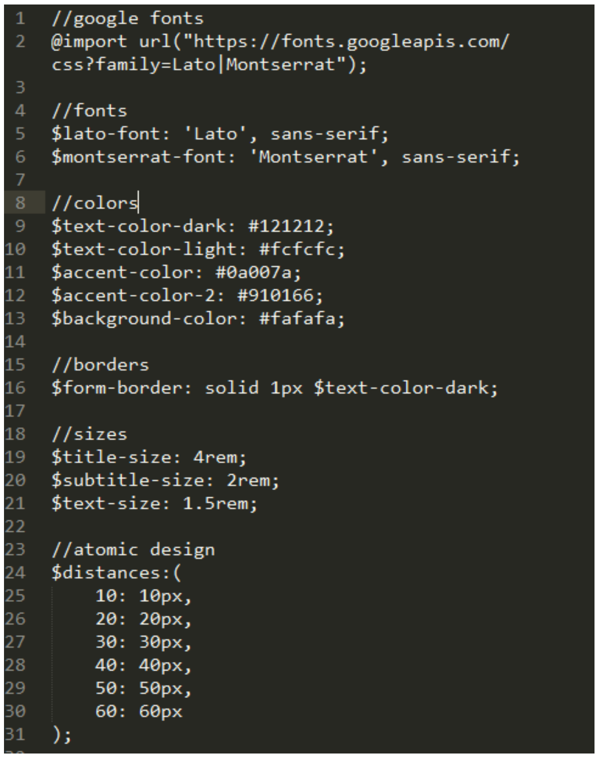
Fig. 1 Waterfall development process for the project

The author uses HTML page, CSS for styling, Javascript and its library, jQuery, for client-side programming, Bootstrap as responsive front-end framework, PHP for server-side programming, and MySQL as database server. Furthermore, this project is also accomplished using SCSS, so-called brand-new stylesheets. The project is also accomplished using other supporting tools such as XAMPP Control Panel to build up the local server.

B. *Using SCSS*

The author uses SCSS quite the same way as using CSS. In addition to CSS functionalities such as selecting object in DOM and media query, the author attempts to use SCSS's features e.g. variables, *mixin*, *extend*, and iteration.

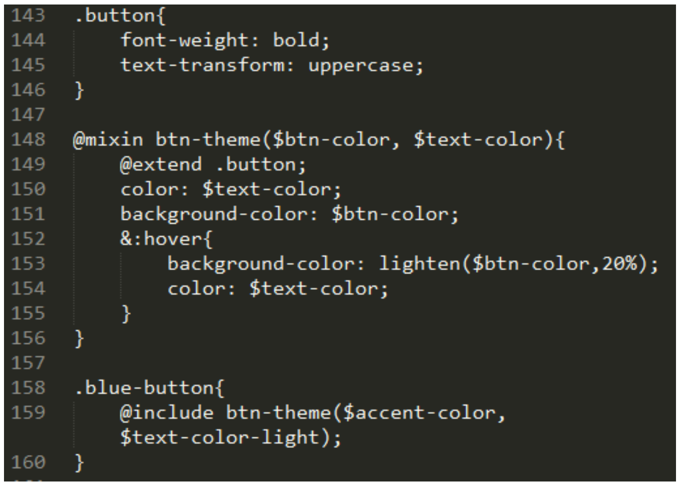
For the sake of variables, the author focuses on the fonts, color hex code, numbers, tuples, and long, irreproducible literal. The figure below illustrates the variables declared and used.



```
//google fonts
@import url("https://fonts.googleapis.com/
css?family=Lato|Montserrat");

//fonts
$lato-font: 'Lato', sans-serif;
$montserrat-font: 'Montserrat', sans-serif;

//colors
$text-color-dark: #121212;
$text-color-light: #fcfcfc;
$accent-color: #0a007a;
$accent-color-2: #910166;
$background-color: #fafafa;

//borders
$form-border: solid 1px $text-color-dark;

//sizes
$title-size: 4rem;
$subtitle-size: 2rem;
$text-size: 1.5rem;

//atomic design
$distances:(
    10: 10px,
    20: 20px,
    30: 30px,
    40: 40px,
    50: 50px,
    60: 60px
);
```

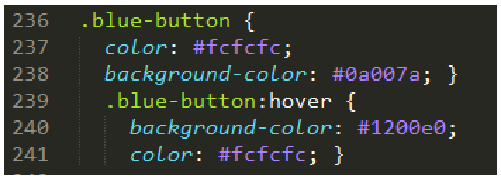
Fig.2 SCSS Variables for fonts, color hex code, long literal, size, and tuple

Mixin and *extend* are quite programmatical. Defining and calling *mixin* are similar like defining and calling function and procedure in procedural programming. There are function's or procedure's name, parameters, and return value if any. However, *mixin* is typically a function which always returns one or more values, that are style attributes. On the other hand, *extend* feature is used if all attributes of a selector are likely to be used in other selectors. The concept is similar like inheritance in object-oriented programming. What being inherited are all of the attributes of the extended selector. These figures below show a SCSS code example using *extend* and *mixin*, and how it is compiled into CSS.



```scss
.button{
    font-weight: bold;
    text-transform: uppercase;
}

@mixin btn-theme($btn-color, $text-color){
    @extend .button;
    color: $text-color;
    background-color: $btn-color;
    &:hover{
        background-color: lighten($btn-color,20%);
        color: $text-color;
    }
}

.blue-button{
    @include btn-theme($accent-color,
    $text-color-light);
}
```

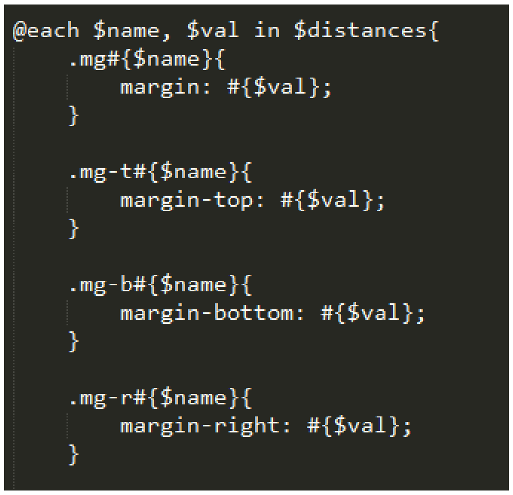
Fig.3 SCSS code example using *mixin* and *extend*



```css
.blue-button {
  color: #fcfcfc;
  background-color: #0a007a; }
  .blue-button:hover {
    background-color: #1200e0;
    color: #fcfcfc; }
```

Fig.4 CSS code after compiled

Last but not least, SCSS supports iteration. The author uses iteration to make atomic design with respect of class selector. We can create atomic CSS using iteration feature using *each*. This figure below illustrates how we use *each*.



```scss
@each $name, $val in $distances{
    .mg#{$name}{
        margin: #{$val};
    }

    .mg-t#{$name}{
        margin-top: #{$val};
    }

    .mg-b#{$name}{
        margin-bottom: #{$val};
    }

    .mg-r#{$name}{
        margin-right: #{$val};
    }
```

Fig.5 SCSS code example using *each*

IV. Results and Discussion

The prototype of this system is perfectly styled with SCSS. In this case, the author manually compiles SCSS file becoming CSS file using command prompt. Since the application also uses PHP, it needs `localhost` to run. Here are few snapshots of the prototype.

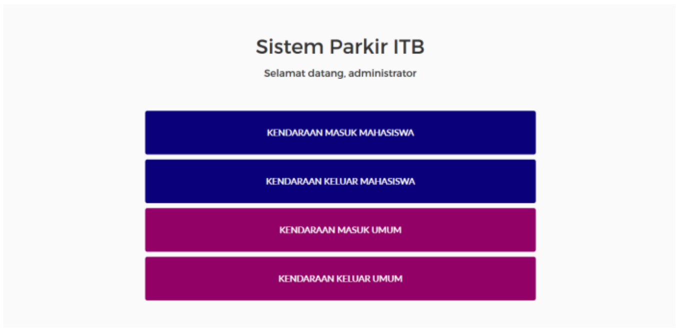
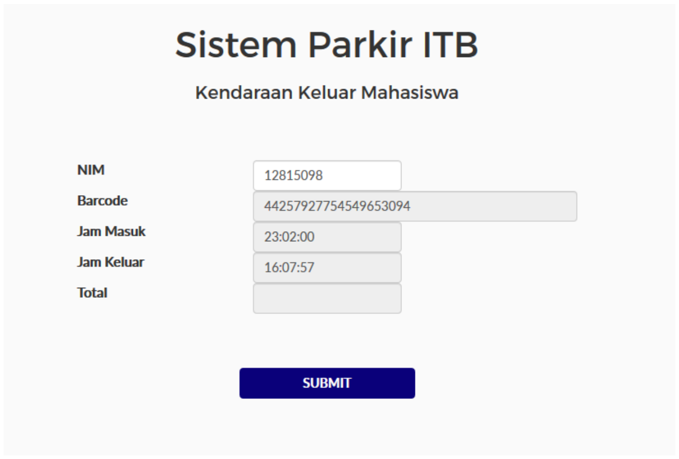

Sistem Parkir ITB
Selamat datang, administrator
KENDARAAN MASUK MAHASISWA
KENDARAAN KELUAR MAHASISWA
KENDARAAN MASUK UMUM
KENDARAAN KELUAR UMUM

Fig.6 Snapshot of administrator dashboard



Sistem Parkir ITB

Kendaraan Keluar Mahasiswa

NIM	12815098
Barcode	44257927754549653094
Jam Masuk	23:02:00
Jam Keluar	16:07:57
Total	

Fig.7 Snapshot of example form

Somehow, the output of learning which the author has got in the project development especially in SCSS learning is quite challenging. As SCSS is used in most dynamic (especially with Ruby on Rails) pages as brand-new stylesheet, implementing styles is not really hard if we have been used to CSS. The figure below illustrates the learning curve of SCSS with respect of effort used and time elapsed according to the author.

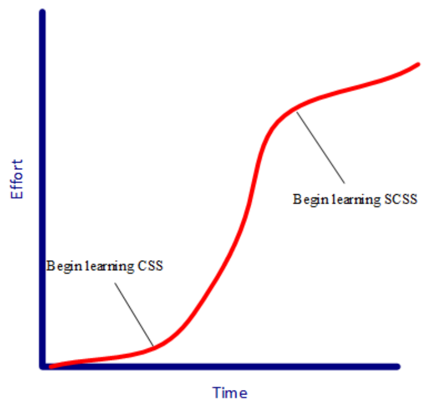


Fig. 8 Learning curve of SCSS

V. Conclusion

To sum up, in this paper, the author has presented how SCSS is used to style web pages in a specific way that is developing front-end of ITB Parking Information System. This system is expected to be used someday when infrastructure and the system itself have been ready to deploy.

The future development may be proposed to enhance this prototype. The author suggests to improve security, manage users, improve user experiences, and handle network issues.

Acknowledgment

First of all, the author would like to thank the Almighty God. Without His blessings, this work would never be accomplished. The author also would like to thank Mr. Rinaldi Munir, Mrs. Ayu Purwarianti, and Mrs. Dessi Puji Lestari for assigning this work. Also, the author would like to thank the clients of the project: Selestina Jessica, Adinda Olivia, and Ananda Dwi (Industrial Engineering ’14) for supporting and cooperating within the project.

References

[1] "Sass," [Online]. Available: https://sass-lang.com. [Accessed May 2017].

[2] H. Giraudel, "What's the Difference between Sass and SCSS?," SitePoint, 28 April 2014. [Online]. Available: https://www.sitepoint.com/whats-difference-sass-scss/. [Accessed May 2017].

[3] Ashley, "Goodbye CSS, Hello SCSS," Nose Graze, 30 October 2014. [Online]. Available: https://www.nosegraze.com/goodbye-css-hello-scss/. [Accessed May 2017].

[4] "Bootstrap," Bootstrap, [Online]. Available: https://v4-alpha.getbootstrap.com. [Accessed May 2017].

[5] B. Frost, "Atomic Design," 6 October 2013. [Online]. Available: http://bradfrost.com/blog/post/atomic-web-design/. [Accessed May 2017].

Statement

I hereby state that the paper I have been writing is original, not an adaptation, nor a translation of one’s paper, nor a plagiarism.

Bandung, May 5 2017

[illegible]

Joshua Atmadja / 13514098