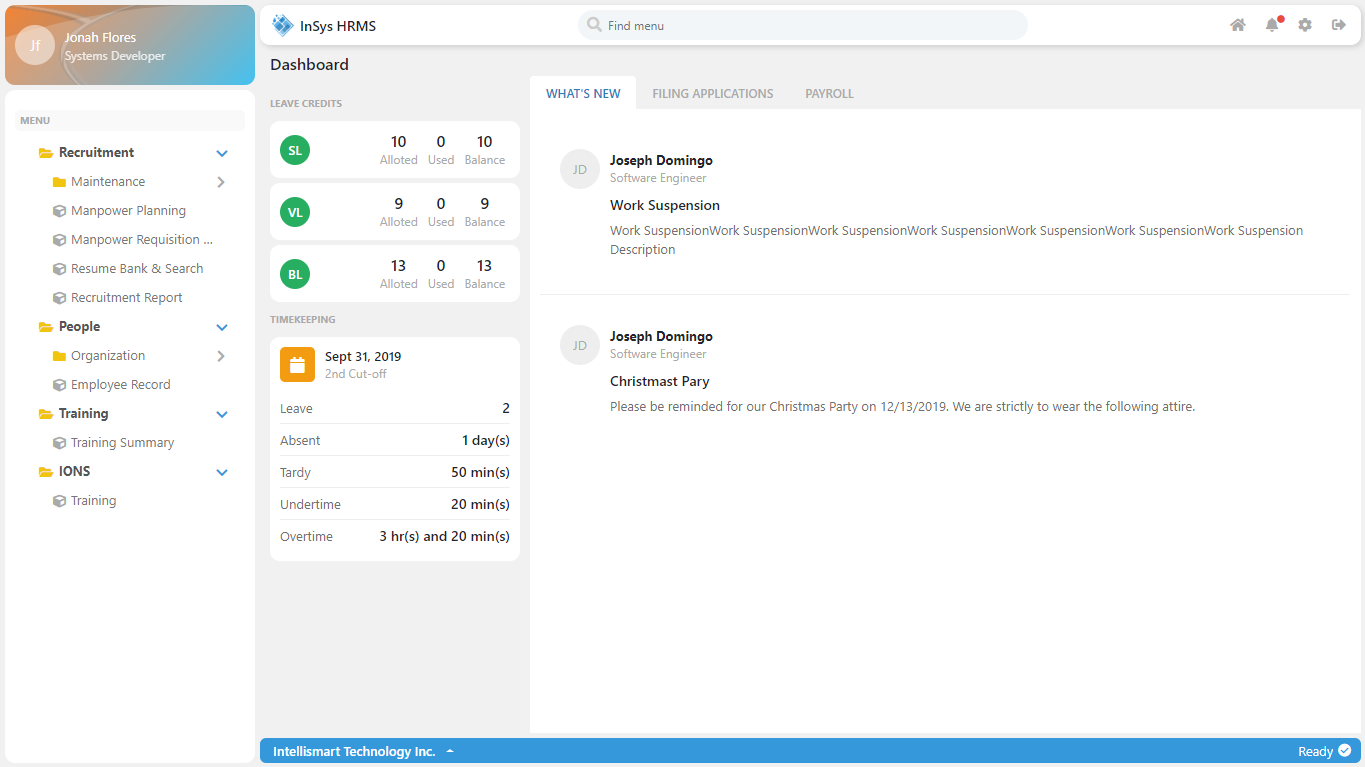
INSYSHRMS STYLE GUIDE

**INTRODUCTION**

InSys HRMS’s design often uses curves and gray or white background colors. White background colors and appropriate use of white spaces (also called negative space) helps users to focus more on the content and minimizes visual distractions. Also, you may take advantage of using colors to emphasize important actions or contents within the page.



*InSys HRMS Landing Page*

**COLORS**

Here are the two important colors in the system. These colors also found in Intellismart logo. Keep in mind that too much colors within the page will distract users and affect their experience.

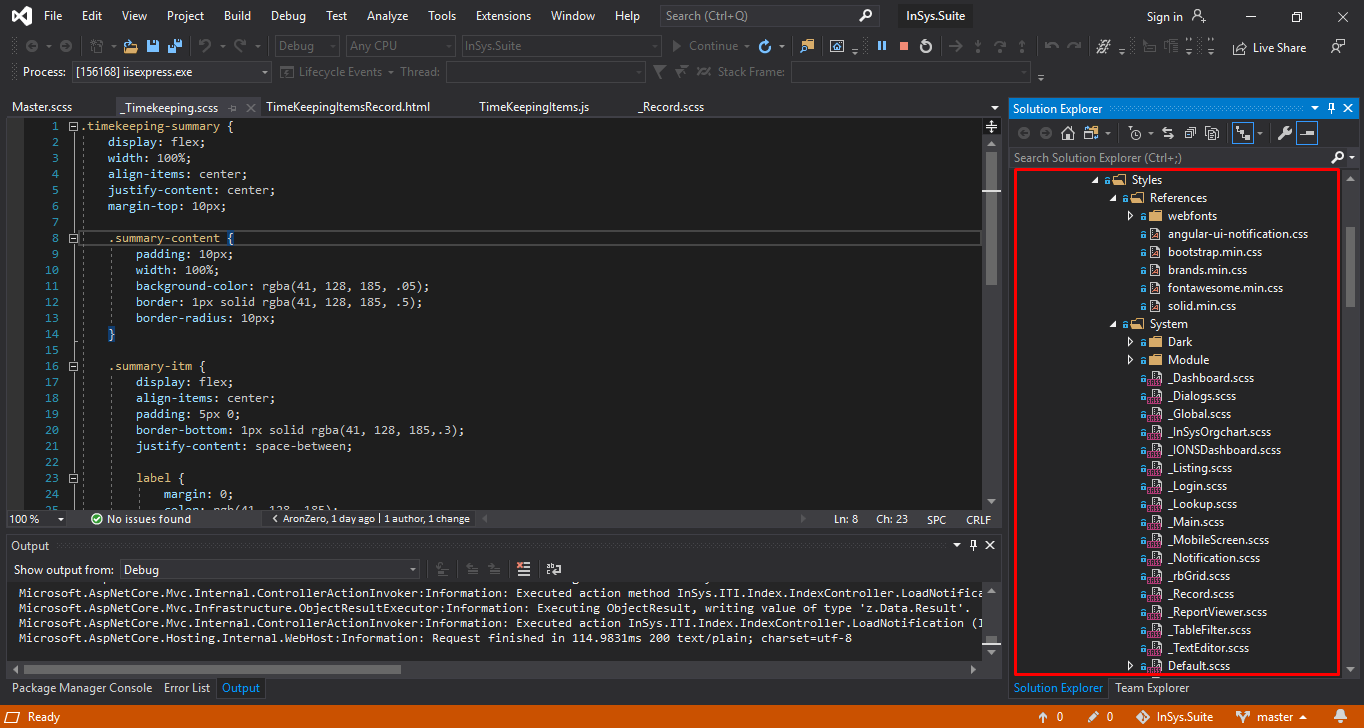
* **Primary Color (#00aeef) –** Dominant color in the system. You may also use this color to indicate action (e.g. buttons, links, etc). Be careful using this much to avoid confusion to users**.**
* **Secondary Color (#f47820) –** Only use this color to emphasize important content or to get the attention of users (any contrasting colors also works).

**TYPOGRAPHY**

**Segoe UI** is used as its font and **Fontawesome 5** for icons. The default size of text is **13px.** Headers, Titles, and Subtitles font sizes varies. The ideal maximum text size is 22px and the smallest would be 12px. Tip: small text (e.g. Subtitle) must use dull colors (such as gray) to distinguish them from the main subject.

**STYLES** **FOLDER**

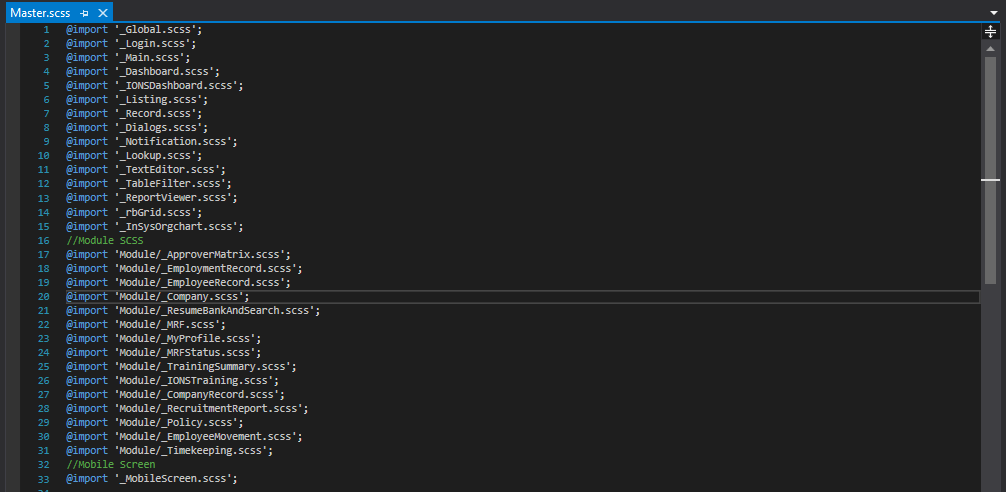
Styles used in the system divides into two main folders:



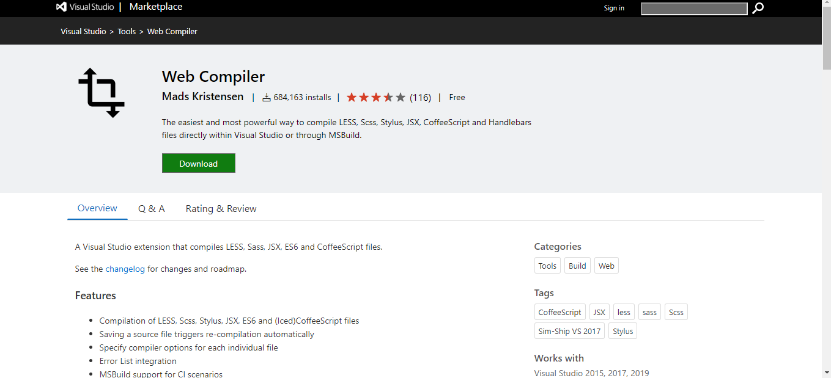
**Reference –** This folder contains only the styles from frameworks such as bootstrap and fontawesome. You may also include your webfont files here.

**System –** This contains all your custom styles. Under this folder, let’s talk about the following:

* Master.scss – This is where your other custom styles should be imported (See image below). As you can see, imported scss starts their name with an underscore (\_) – They are called Partial SCSS. Take note that the compiler compiles only the scss file without an underscore at the start of their name (well, it depends on the compiler you’re using).

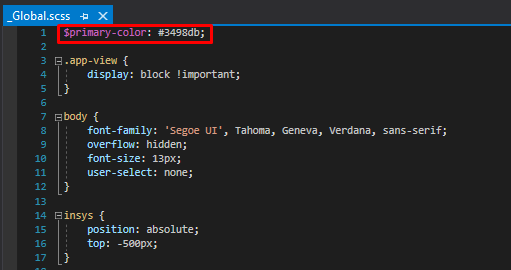


**NOTE**: To compile Master.scss you will need **Web Compiler** extension installed in your Visual Studio (See image below). Right-click on Master.scss > Web Compiler > then click “compile file” or “Re-compile file”. Once compiled, hitting Ctrl + S automatically compiles your scss.



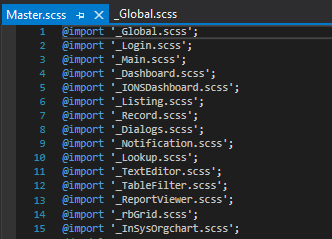
<https://marketplace.visualstudio.com/items?itemName=MadsKristensen.WebCompiler>

* \_Global.scss – Write all your css property here that you want to use globally (e.g. font-family). Ideally, I put down all scss variables such as colors (see image below)



In SCSS, you can declare variables by having “$” at the start of their names. This will help to easily change the values and applies to all css properties that are using those variables.

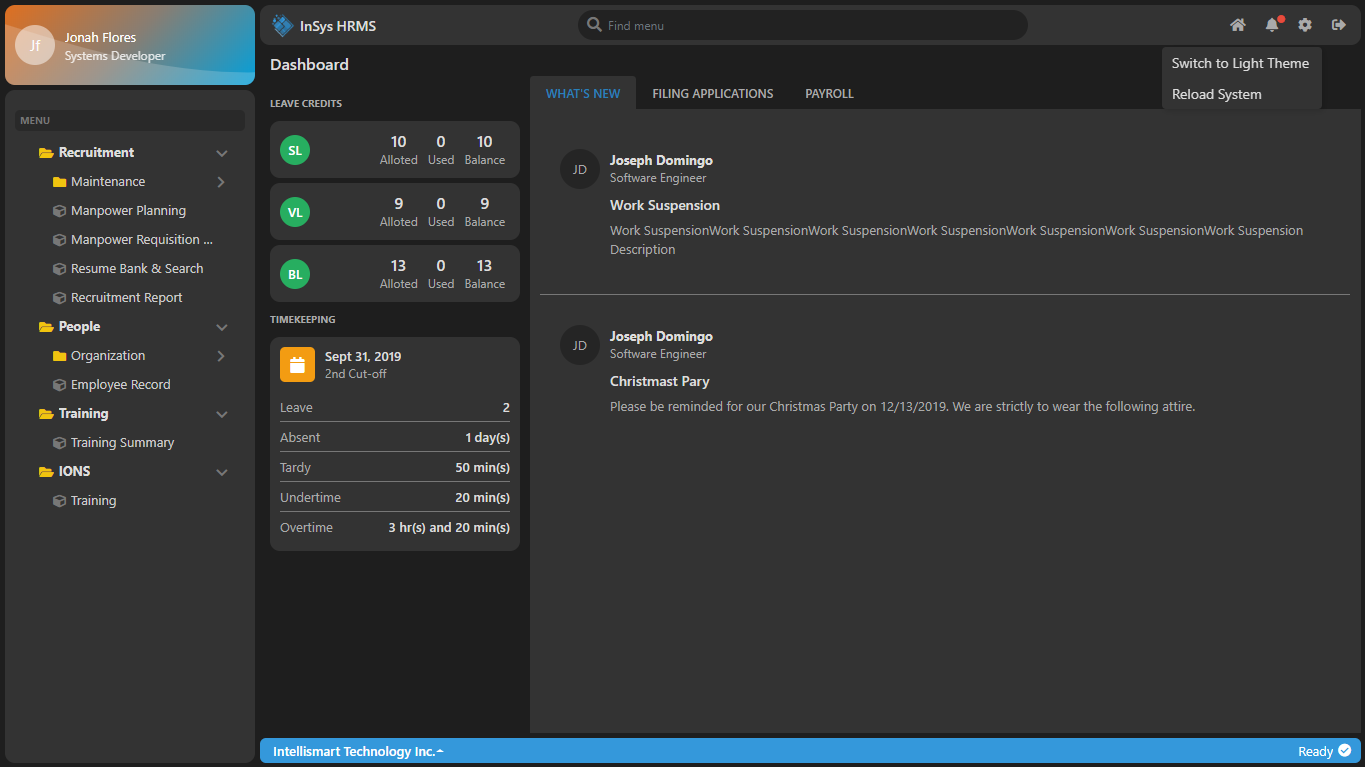
**NOTE**: Always import \_Global.scss at the top line of your code (see image below) specially if you declare scss variable inside it since css code is also interpreted sequentially.



* Other partial scss – If you decide to create your own style, make sure to name it relevant to what you are styling with. Create folders if necessary and be organized as possible.

**DARK** **MODE**

Well, this is optional but whenever you want to activate this feature, make sure that any changes made on the default style must reflect on the dark style.



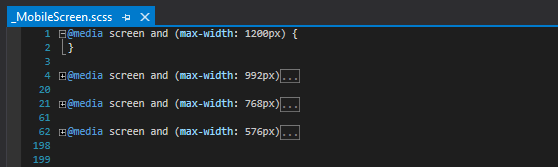
*Landing Page in Dark Mode*

If you check on Styles > System > Dark folder, all scss file are the counterpart of the default scss files except the partial scss files starts their name on “\_Dark\_” and their css properties differs only in color to compliment in the dark theme.

**NOTE**: The counterpart of Master.scss is Dark\_Master.scss. Dark\_Master.scss must be compiled like Master.scss. There are instances that when you hit “Ctrl + S” (when you are developing), changes are not applied even when you refresh the web page. Just re-compile both Dark\_Master.scss and Master.scss using Web Compiler then refresh the page and you’re good to go.

**MOBILE** **RESPONSIVENESS**

In our system, **\_MobileScreen.scss** (In dark theme, \_Dark\_MobileScreen.scss) handles custom behavior of our element whenever we reach certain screen width – also what we called **CSS** **Breakpoints** or **Media Query Breakpoints**. The example below contains breakpoints based on the version of our bootstrap. The version we are currently using is Bootstrap 4.



It’s up to you if you want to add more breakpoints, the example above is aligned to bootstrap to easily detect the changes/behavior since other elements are solely depend on bootstrap css classes to make them responsive. Note that, everything you write down inside these breakpoints are custom and may overwrite some default behavior.

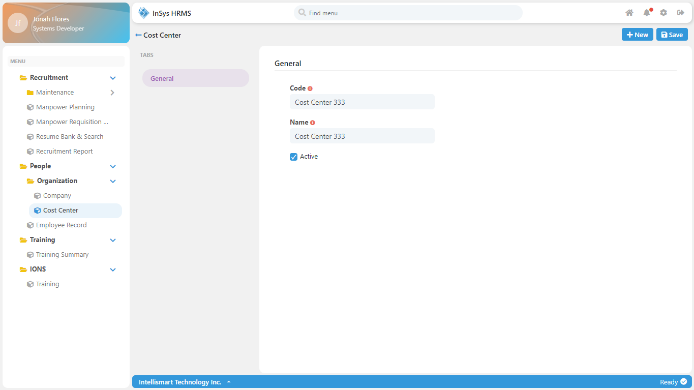
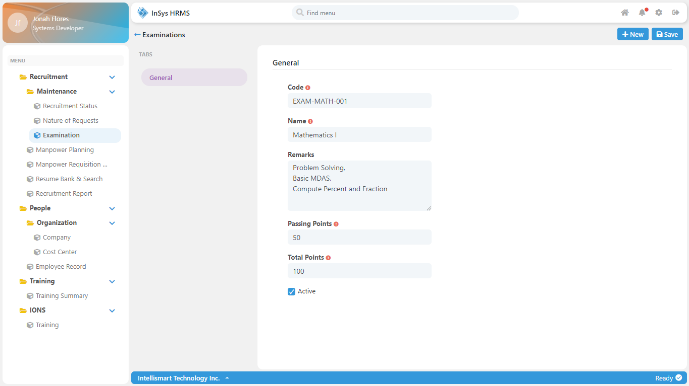
**TIP**: It’s better if you have basic understanding of Bootstrap Grid System since our system mostly depend on these to make itself responsive. Just use custom breakpoints for complex styling.

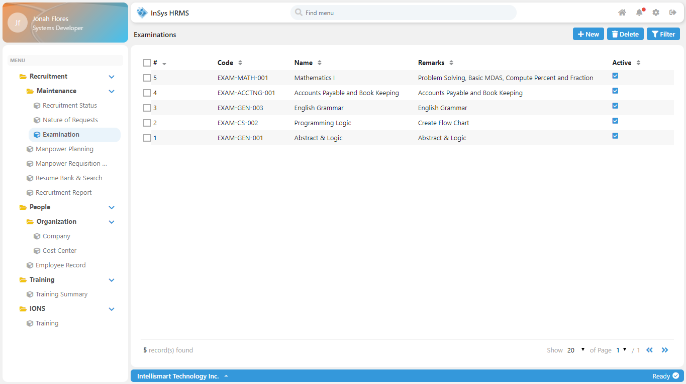
**GENERIC** **TEMPLATES**

Consistency is very important. This will help users to be familiar in the system in a short period of time. Most of our modules uses the same templates and css properties. Listing and Record Page are the templates that have resemblance in most modules (See examples below).

**Examinations (Record Page)**

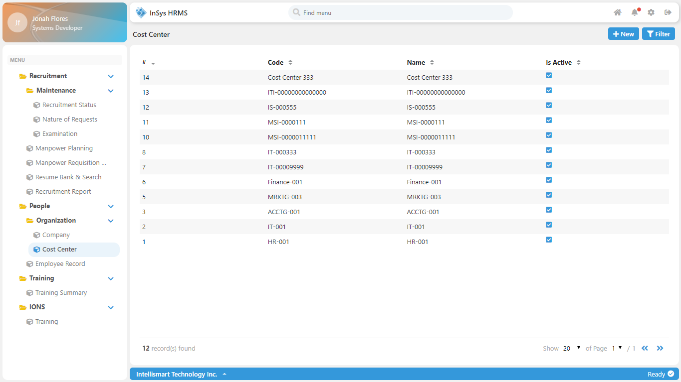
**Cost Center (Record Page)**

****



**Cost Center (Listing Page)**

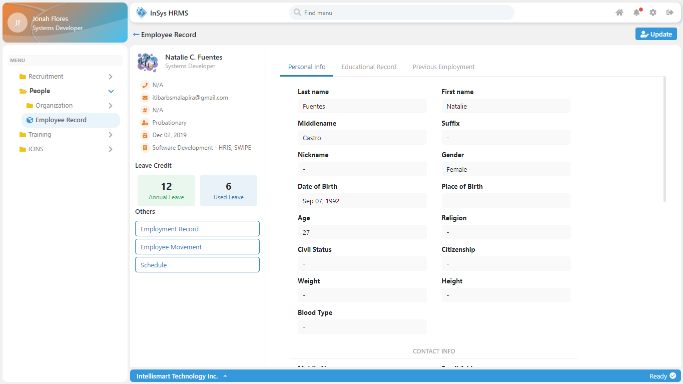
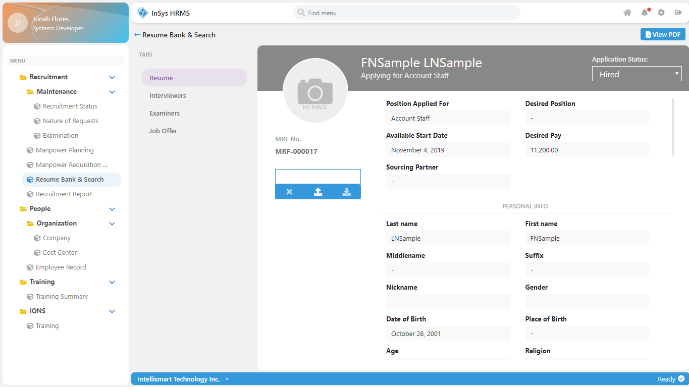
**Examinations (Listing Page)**



**NOTE:** In some cases, complex modules need to have custom template. In this phase, you need to understand the process and design a template that is easy for the users to use. Make it intuitive and prioritize user experience. Here are some of the modules with custom templates:

**Resume Bank & Search (Record Page)**

**Employee Record (Record Page)**



**TIP**: Add separate partial scss file for module with custom template. You most likely need to be organized since they have unique css properties. You may add your module-specific scss file to **Styles > System > Module** folder (In dark theme: Styles > Dark > Module).

You’re on your own creating custom templates. Fields and other components are already available in the system so you’ll only need to pay attention on the layout and contents to display. You may also check on other modules and look for existing components and layout for you to re-use.

**OTHER** **THINGS** **TO** **CONSIDER**

Here are some of the tips that may help you in your development.

* Create mock-up using prototyping tool or just a sketch. It helps you to visualize and make changes quick and easy before you start coding.
* Don’t be afraid to use third-party frameworks. It will make your development much faster. There are frameworks that are already beautiful, optimized and compatible to all browsers.
* Store color in SCSS variables. With this, you can easily change the entire system’s look and feel.
* Search for design inspirations on the internet. Be updated on web design trends.
* Prioritize User Experience. Design is not just the appearance, it’s also the experience.
* Practice having the maximum of 3 nested CSS selector when coding in SCSS to avoid clutter and confusion