|  |
| --- |
| Singapore Polytechnic – Software Application Project |
| ACKTEC Design Studio |
| Source Codes Guide |

|  |
| --- |
| Yu  11-9-2022 |

Table of Contents

[GitHub & Branches 2](#_Toc118907252)

[ReactJS 5](#_Toc118907253)

[Why functional components? 6](#_Toc118907254)

[SASS 7](#_Toc118907255)

[Wth is SASS? 7](#_Toc118907256)

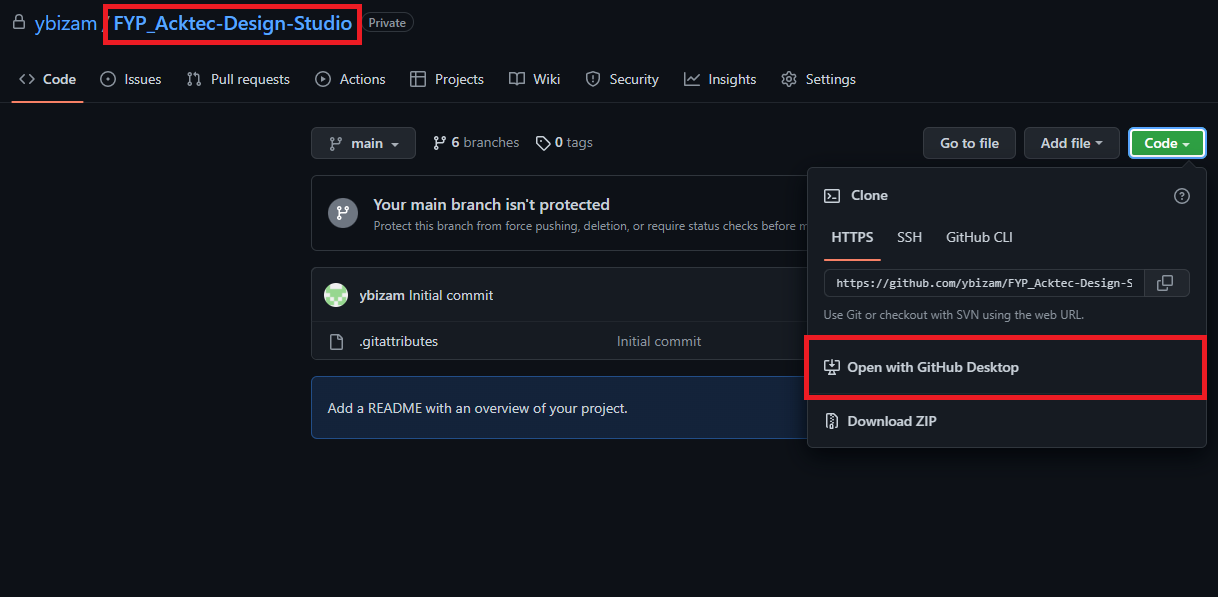
[SASS in Visual Studio Code 9](#_Toc118907257)

# GitHub & Branches

GitHub is being used as our repository to store the source code for the ACKTEC Design Studio application.

You **need** to use GitHub Desktop in order to fully utilize the repository, and connect it to your Visual Studio Code (VSC) for easy editing.

Be sure to clone the repository by visiting the website (<https://github.com/ybizam/FYP_Acktec-Design-Studio>) and open it in your GitHub Desktop like so:



You should clone the repository in GitHub Desktop. You can change the local path if you want to place it somewhere you won’t forget:

Graphical user interface, application

Description automatically generated

You should notice under the “Branches” tab that I’ve already created branches for each member in the group. This will be the place where you should work on your codes. Do **not work on your codes in the main branch**.

A screenshot of a computer

Description automatically generated with medium confidence

Do **not merge any of the branches** unless we’re all in a call together to settle the merging and conflicting codes between different branches. This is to prevent any missing codes or work that you or any other member in the group have done. **If the merge is important**, **just message the group chat about it and we can have a short meeting about it**. I don’t want any progress to be lost (or any heartbreaks either lol)

To make sure that your written codes are still available, I recommend **writing codes in a backup folder** before putting it in the GitHub repository branch. This is what I did, just in case: Graphical user interface, text

Description automatically generated

Moreover, you **cannot** have the *node\_modules* folder in **any** of the branches. Please make sure it isn’t there when trying to push your requests. It takes like 70 eons just to move it around.

Once you’ve gotten the repository and moved the files inside to a different folder, please use the *npm i* command to download all the current node modules in the folder itself, in case one of us adds or removes any modules in the project itself. You can also let people know by telling the group chat.

# ReactJS

All source codes are written under the *src* folder. You can see I already sectioned it off to backend and frontend, as well as the different folders for the different JavaScript files for the different pages in the application.

All static images (like the ACKTEC logo) will be in the *public* folder.

**DO NOT** modify the *index.css*, *manifest.json* or *package.json* file. This may affect the application greatly and can cause crashes, making it unusable.

*App.js* will only be used to create navigation links, importing the pages and store any global variables needed for most pages. Don’t know what I mean? Here’s an example:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

It shouldn’t really contain anything else other than functions to check if the user is logged in or not.

All JS files **need to be a functional component**. If you’re wondering what’s the difference, this is a functional component:

A screenshot of a computer

Description automatically generated with medium confidence

The entire code is encased in an export function written like:

|  |
| --- |
| export const pageName = () => {  const [variable, setVariable] = useState(“I am a variable!”);  return(  <div>  Your HTML code is here. To use a variable, we write it like {variable}.  </div>  )  } |

With the exception of importing any components you need of course.

A class component will be written like:

|  |
| --- |
| export default class App extends component (  constructor() {  this.state = {  variable: “I am a variable!”  }  }  render() {  return <div>  Your HTML code is here. To use a variable, we write it like {this.state.variable}.  </div>  }  ) |

## Why functional components?

Honestly, it’s just much easier to read by others if you look at the above two examples. Carrying over variable data is also much more easier using functional components, and it reduces the number of lines of code we need to write. It also separates the functions you need to write for the front-end and the html code to display it to the users very well.

Literally, it’s recommended by the ReactJS creators as well. Look at this (<https://reactjs.org/docs/hooks-intro.html>):

A screenshot of a computer

Description automatically generated with medium confidence

(also TIL reactjs is owned by Meta lmao)

TLDR; use Functional Components (aka React Hooks) when writing the pages.

If you need more guidance on how to use ReactJS and functional hooks, you can take a look at this video. It’s fairly long, but follow through it and you’ll be able to understand ReactJS with ease:

Graphical user interface, application

Description automatically generated

<https://www.youtube.com/watch?v=w7ejDZ8SWv8>

# SASS

## Wth is SASS?

It says it’s an extension language of CSS that interprets/compiles back into CSS, but it’s a boring description of it. It generally helps to make things look neater whilst writing the source codes without damaging the styling itself by converting it to CSS as VSC is watching the SCSS file. It just basically helps us as coders to understand the StyleSheet better.

Here’s a good example of how SASS is used:



It helps to nest different parts of the styling into one main component as opposed to multiple. It’s fairly easy to read due to the nesting.

Text

Description automatically generated

Gives variables to allow easy edits for the styling, like the color of certain components in the application.

Text

Description automatically generated

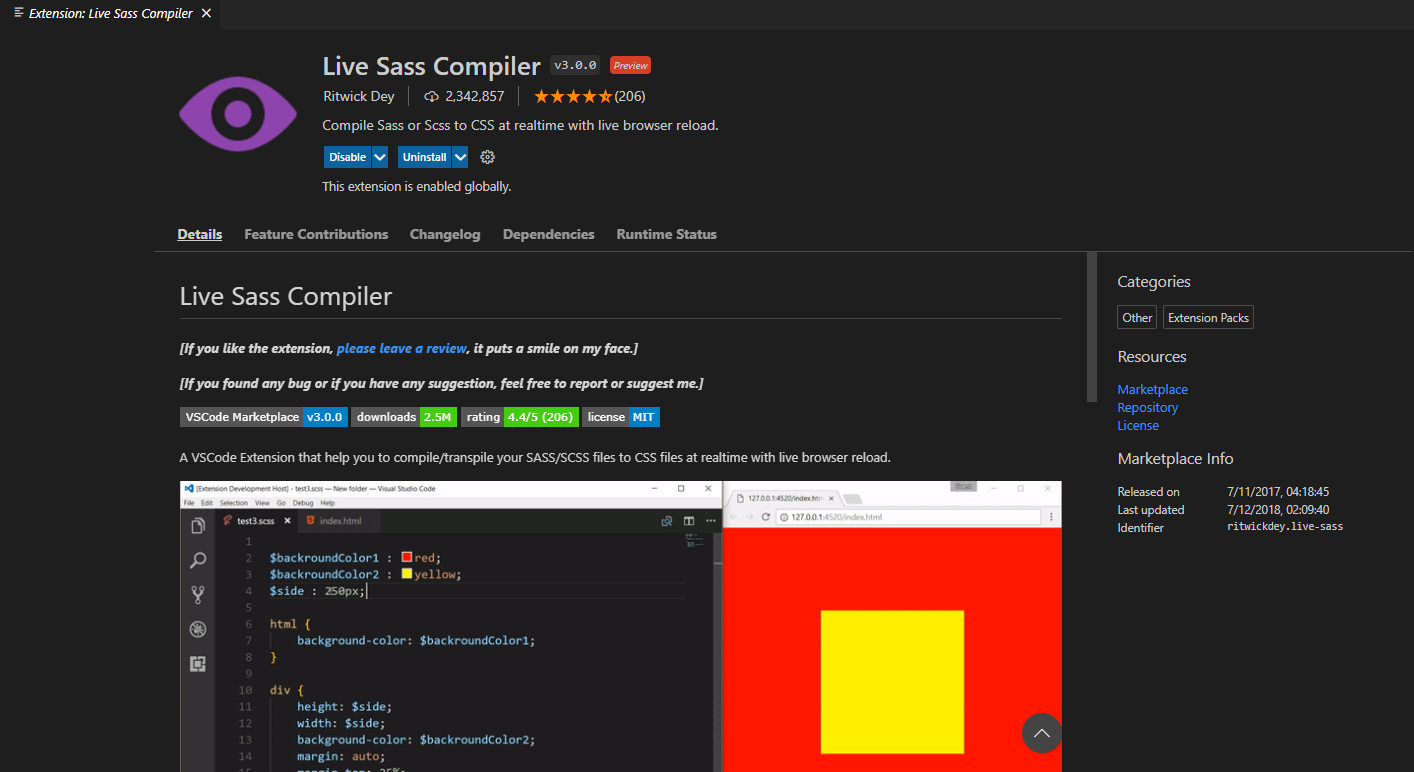
It also allows to use pre-written declarations that can be reused for any element using the *@mixin* command!

There’s more to SASS that you can do. Check it out in this link:

<https://dev.to/annequinkenstein/sass-2dph>

## SASS in Visual Studio Code

Install this extension in Visual Studio Code called the “Live Sass Compiler”:



This will help automatically update the css file whilst you’re writing the styles in the SCSS (SASS) file.

Once it’s downloaded, go to the *App.scss* file I’ve already created in the source code itself:

Graphical user interface, text, application

Description automatically generated

On the bottom right, you should see a button that says “Watch Sass”, with the eye icon beside it:



Once clicked, you should see this pop-up, indicating that it is watching the SASS file to automatically compile the code as a CSS file:

Text

Description automatically generated

Now, you only have to write your styling in the SCSS file. Do **not** write/manipulate anything in the CSS file. The extension will help to automatically convert it.

If you wanna know more about using SASS, you can also take a look at this video here:

A picture containing text, screenshot, businesscard, clipart

Description automatically generated

<https://www.youtube.com/watch?v=nu5mdN2JIwM>