Coding Standards

AiPi

Contents

1. Coding Conventions…… 3
    1.1. Compliance to Conventions…… 3
    1.2. Naming Conventions…… 3
    1.3. Formatting and indentation…… 3
    1.4. Comments…… 4

2. File Structure…… 4
    2.1. Front-end…… 4
    2.2. Back-end…… 4
    2.3. Branching…… 5
    2.4. Commit…… 5

1. Coding Conventions

1.1. Compliance to Conventions

The AiPi team had a meeting earlier in the year where we discussed the necessary steps to take for all code no matter front-end or back-end to be of the same coding conventions before pushing that code onto the GitHub repository. All members agreed to these steps and thus ensured that their own code followed the coding conventions.

1.2. Naming Conventions

- Function Names will not start with a capital letter.
- Function Names will not contain any number or special character, the exception being an underscore.
- Global variables for specific classes will not start with a capital letter.
- Global variables will not contain any number or special character, the exception being an underscore.
- Variable names may contain a number, in the sense that the variables correlate to one another in a chronological order. (e.g. date1 and date2)
- All variable and function names should be meaningful and correlate to their functionality

1.3. Formatting and indentation

The backend language that was used is python, while the front-end language that was used was HTML, JavaScript and CSS. Therefore, there was a huge emphasis on indentation not only from our side as a group but also in order for the code to run correctly. Therefore, our formatting and indentation rules apply to both the front end and back end.

- All loops will have the inner function and code indented with a tab.
- At the end of a function and before the declaration of a new function a line must be left open.
- Within function lines can be left open when parts of the code is not in a following order, this is needed in order to make the code easier to read,(e.g. after declaring variables leave a line before the for loop)
- Indentation and formatting must be applied according to the languages rules as well (e.g., pythons' indentation)

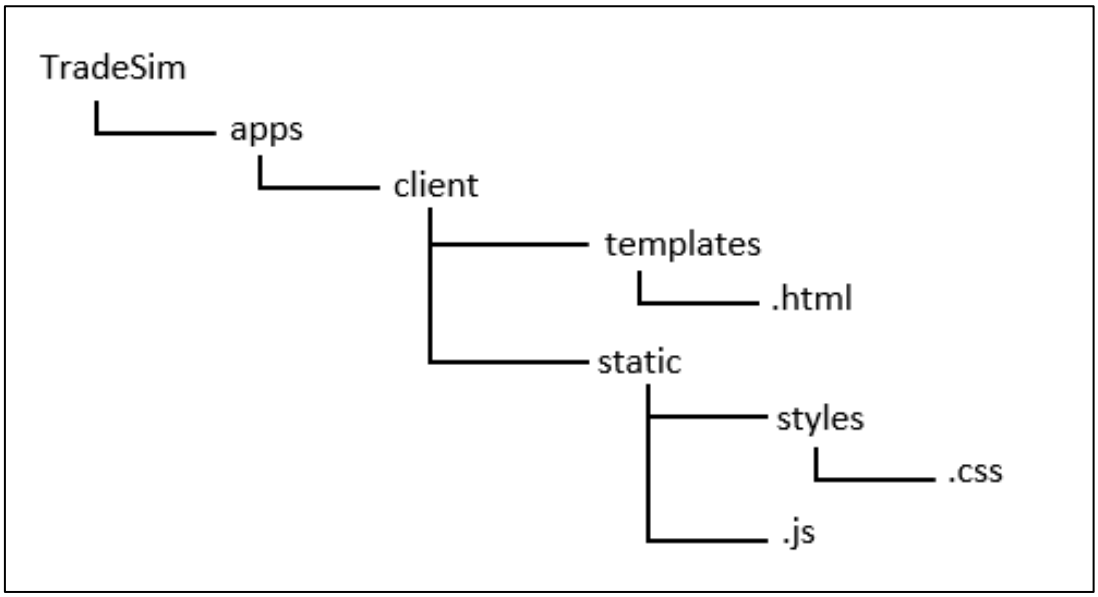
1.4. Comments

- Use of comments is allowed
- Comments can be added when trying to explain what a particular section of code does, such that other developers can read the comment

- Comments may also be used for developers when they are in the testing phase and use comments to keep track of notes. These comments must be removed before the final push to the develop branch,
- Random comments are not allowed and will be deleted

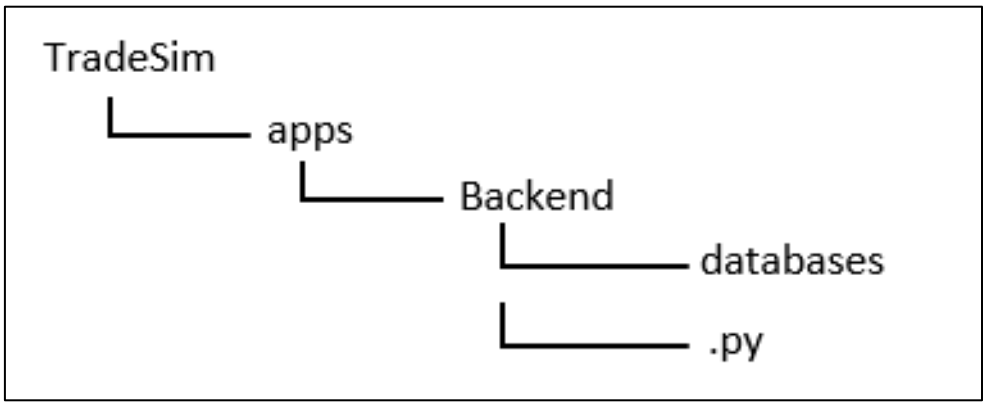
2. File structure

2.1. Front-end



The picture above shows the file structure of the front-end on the repository. All the html files are placed in the templates folder. Within the static folder contains the JavaScript files as while as a folder named styles which contains all the CSS files. Both the templates and static folder are placed within the client folder.

2.2. Back-end



The picture above shows the file structure of the back end on the repository. All the python files are placed within the Backend folder. In the Backend folder is a folder named databases, in these folders is the csv file named industries, this file is used for all industry and sector names with corresponding codes. Finally, the Backend folder is a sub folder of the apps folder.

2.3. Branching

- There is one main Branch
- Each Developer will create their own fork
- Each Developer will work in their own fork and commit to this fork
- When the developer has finished their code, they will create a pull request
- The pull request will merge their code to the main branch

2.4. Commit

- Commits must have a heading; this heading must contain whether the code is for front-end or back-end. By using brackets at the front of the heading e.g. (Backend)
- Each commit must contain a description that contains all the additions or changes to code that was made with this commit.