QUY ĐỊNH NGUYÊN TẮC KHI CODE

Những nguyên tắc cần tuân thủ khi code.

FRONT-END

1. Quy tắc comment

1.1 Comment cho Class (Component):

```
/*
[Author] - Tên tác giả
[Desc] - Nội dung của class (component)
*/
```

Ví Dụ

```
/*
[Author] - Đỗ Trường Giang
[Desc] - Đây là Component Header đùng để hiển thị menu chính của app
*/
import React, {Component} from 'react';
class Header extends Component {
  render() {
    return(
      <h1>Giang so handsome</h1>
    );
  }
}

```

1.2 Comment cho Class (Component) khi có BUG:

```
/*
[BUG_01] - Nội dung của bug
  [BUG_02] - Nội dung của bug
*/
    class Header extends Component {
      render() {
        return(
          <h1>Giang so handsome</h1>
        );
```

```
      }
    }
/*
  [/BUG_02]
[/BUG_01]
*/
```

Lưu ý: Khi fix được bug nào thì chỉ cần bỏ tag bug đó

Ví Dụ: Sau khi fix được **BUG_02** nhưng chưa fix được **BUG_01** thì sẽ comment như sau:

```
/*
[BUG_01] - Nội dung của bug
*/
    class Header extends Component {
      render() {
        return(
          <h1>Giang so handsome</h1>
        );
      }
    }
/*
[/BUG_01]
*/
```

1.3 Comment cho Class (Component) khi đã test:

```
/*
[TESTED] Tên người kiểm thử - Ngày kiểm thử - lần thứ bao nhiêu
[Author] - Tên tác giả
[Desc] - Nội dung của class (component)
*/
```

Ví Dụ:

```
/*
[TESTED] Đỗ Trường Giang - 27/02 - 2
[Author] - Đỗ Trường Giang
[Desc] - Đây là Component Header đùng để hiển thị menu chính của app
*/
import React, {Component} from 'react';
class Header extends Component {
  render() {
    return(
      <h1>Giang so handsome</h1>
    );
```

```
  }
}

```

1.4 Comment cho function:

```
/*
[Author] - Tên tác giả
[FunctionName] - Tên function là gì
[Desc] - Nội dung của function là gì
:param1: đối số thứ nhất là gì - kiểu dữ liệu của đối số 1 (nếu có)
:param2: đối số thứ hai là gì - kiểu dữ liệu của đối số 2 (nếu có)
…
:return: function này trả về cái gì
*/

```

Ví Dụ:

```
/*
[Author] - Đỗ Trường Giang
[FunctionName] - onShowList
[Desc] - Hiển thị danh sách các task của user
:param1: tasks - array
:return: danh sách các task của user - jsx object
*/
onShowList  = tasks => {
  let list = tasks.map( task => {
    return <p key={task.id}>{task.content}</p>
  } );
}

```

1.5 Comment khi có người sửa:

```
/*
[MODIFIED]
[Author] - Tên tác giả sửa
[Desc] - Nội dung sửa là gì
*/
```

Lưu ý: comment này phải ở dưới function của người làm ra function đó và phải comment cả function của người làm ra rồi sửa lại ở dưới để tiện cho việc xem lại code cũ

Ví Dụ:

```
/*
[Author] - Đỗ Trường Giang
[FunctionName] - onShowList
[Desc] - Hiển thị danh sách các task của user
:param1: tasks - array
:return: danh sách các task của user - jsx object
*/
/*
  onShowList  = tasks => {
    let list = tasks.map( task => {
      return <p key={task.id}>{task.content}</p>
    } );
  };
*/

/*
[MODIFIED]
[Author] - Lưu Thành Đạt
[Desc] - chuyển task.id thành index
*/
onShowList  = tasks => {
  let list = tasks.map( (task, index) => {
    return <p key={index}>{task.content}</p>
  } );
};

```

1.6 Comment cho function khi có vấn đề về hiệu năng:

```
/*
[metric]
*/
<Function có vấn đề về hiệu năng>
/*
[/metric]
*/

```

Ví Dụ:

```
/*
[metric]
*/
onSort = (tasks) => {
  for( let i = 0; i < tasks.length; i++ ) {
    for( let j = i+1; j < tasks.length - 1; j++ ) {
      <Nội dung xử lý>
    }
  }
};
/*
[/metric]
```

```
*/
```

2. Quy tắc đặt tên cho file và class (component)

Tên của class (component) phải là danh từ và in hoa chữ cái đầu và tên của file phải trùng với tên của class (component)

Ví Dụ:

```
file: Header.js
  import React, {Component} from 'react';
  class Header extends Component {};

  export default Header;
```

3. Quy tắc đặt tên cho function

Tên function phải là động từ viết theo **camel case** và phải có 'on' đằng trước tên function

Ví Dụ:

```
onShowList = () => {};
onAddTask = () => {};
```

4. Quy tắc đặt tên biến

Tên biến không được đặt tùy tiện như là a hay b hay là ironman mà tên biến phải có nghĩa đọc vào có thể hiểu được biến đấy dùng để lưu trữ cái gì

Ví Dụ:

```
let tasks;
```

Lưu ý 01: khi tên biến quá dài thì hãy dùng camel case để thay thế khoảng trắng (dùng bên phía front-end)

```
let numberOfTask;
```

Lưu ý 02: hạn chế sử dụng var để đặt cho biến hãy sử dụng let hoặc const để thay thế

```
var task; // HẠN CHẾ
let task; // NÊN DÙNG

```

Lưu ý 03: khi đặt biến là const thì hãy ghi in hoa toàn bộ tên biến

```
const PI;
```

Lưu ý 04: đối với array thì tên biến phải là danh từ số nhiều và object thì tên biến phải thể hiện được đối tượng

```
let tasks; // array
let task; // object
```

5. Quy tắc đặt tên class và id khi css

Dùng chuẩn BEM để đặt tên cho class và id khi cần css

```
.block {}   /* Block */
.block__element {}  /* Element */
.block--modifier {}  /* Modifier */
```

Ví Dụ: Element

```
<div class="td">
  <div class="td__title">
  </div>
  <div class="td__description">
  </div>
</div>

```

Ví Dụ: modifier

```
.btn--red {
      background: red;
 }
```

BACK-END

1. Quy tắc comment

Tượng tự như Front-End.

2. Quy tắc đặt tên file và class

Tên file và tên class đặt theo chuẩn PascalCase

```
// file TaskController.java

class TaskController{
    // content class
}
```

3. Quy tắc đặt tên function

Tên function/method đặt theo chuẩn camelCase.

```
class TaskController{
    public void addTask(){
        // content method
    }
}
```

4. Quy tắc đặt tên biến

Tên variable/attribute đặt tên chuẩn underscore. (đối với hằng số thì uppercase)

```
int number_of_tasks;
const int MAX_ARRAY;
```

5. Clean Code

- Đặt tên phải có nghĩa.
- tên file/class/biến nên đặt danh từ hoặc cụm danh từ
- tên function/method nên đặt động từ hoặc cụm động từ
- Về spacing

```
// giữa các operator phải có khoảng trắng
a = 1;
b = (a + b) * 2;
for (int i = 1; i <= n; i++)
```

- Về các cấu trúc if, for, while, ...

```
// spacing rõ ràng và {} đúng nguyên tắc

for (int i = 0; i <= 10; i++){
    // content
}

while (n != 0){
    // content
}

if (a == 1){
    // content
}
else if (b == 1){
    // content
}
else{
    // content
}

// content dù có 1 lệnh cũng phải có {}

// Khi for một đối tượng dạng array
// Nếu không cần quan tâm đến index thì dùng iterator hết

for (int e : arr_e){
    // content
}
```

Nguyên Tắc đặt tên commit message

```
// Cú Pháp
[<Tên task id>]-[<Hành động>] : <thông tin>

+ Tên task id nằm trong jira (nếu chưa có task đó thì nói leader tạo task trên jira)
+ Hành động:
    + Add = Thêm vào mã nguồn.
    Ví dụ: chức năng, test, thư viện

    + Drop = Xóa khỏi mã nguồn.
    Ví dụ: chức năng, test, thư viện

    + Fix = Sửa trong mã nguồn.
    Ví dụ: lỗi, typo

    + Bump = Thay đổi version.
    Ví dụ: nâng phiên bản một thư viện đang sử dụng

    + Make = Thay đổi công cụ hoặc quy trình build liên quan hạ tầng

    + Refactor = Sửa đổi nhằm mục đích tái cấu trúc mã nguồn cũ.
    Ví dụ: Tách logic xử lý trong controller layer về business layer

    + Optimize = Sửa đổi nhằm mục đích tối ưu hiệu năng cho mã nguồn cũ.
    Ví dụ: Tối ưu hiệu năng chức năng tìm kiếm đơn hàng bằng cách
    psử dụng thêm caching layer để giảm thời gian truy vấn từ cơ sở dữ liệu.

    + Reformat = Sửa đổi nhằm mục đích định dạng lại code cũ.
    Ví dụ: xóa khoảng trắng, dòng trắng sai coding convention

    + Rephrase = Sửa đổi liên quan tài liệu trong source code.
    Ví dụ comment trong source code (TODO / FIXME / …)

    + Document = Sửa đổi liên quan đến tài liệu bên ngoài source code.
    Ví dụ thêm mô tả vào file README.md

+ Thông tin: đối tượng của hành động

Ví dụ:
[TODO_05]-[Add] : file TaskController.java

** Lưu ý: Commit từng file
```