**CODING STANDARDS**

1. **Thiết kế module:**Code nên chia thành các module riêng biệt, mỗi module thực hiện đúng chức năng của mình. Sẽ không nói rõ chi tiết tại đây, tuy nhiên code cần đơn giản, rõ ràng, tách biệt và liên kết khi cần thiết.
2. **Cách tổ chức source code:**

Các file cùng loại sẽ được tổ chức trong cùng 1 thư mục tương ứng.  
Ví dụ: các file hiển thị giao diện sẽ để trong folder “View”, các file định nghĩa các class tương ứng trong database sẽ để trong folder “Model”,…

Xem chi tiết trong file “Cấu trúc thư mục project”.

# Quy cách code:

## Quy cách đặt tên:

* Đặt tên biến/hàm phải đi liền với ý nghĩa sử dụng.  
  Ví dụ: string username = “John”; user có name là “John”  
  function copyName(string nameInput) – phương thức copy từ 1 name cho trước…
* Tên biến và hàm nên đặt theo camelCase (javascript)  
  Ví dụ: nên đặt là userName chứ không phải là username
* Tên hàm nên là động từ và tên biến là danh từ  
  Ví dụ: var product, function sendMessage(),…
* Các biến hoặc hàm trả về kiểu boolean nên dùng “is”, “can”, “have”,… đứng trước tên biến/hàm  
  Ví dụ: isExist, canApply, havePermission,...
* Các constant sẽ UpperCase toàn bộ, mỗi chữ cách nhau bởi dấu gạch dưới “\_”  
  Ví dụ: const MAX\_LENGTH = 100
* Các biến tạm, hay đặt tên rõ ràng, ko sử dụng temp.
* Các interface thì phải có chữ “I” phía trước
* Tên class đặt theo PascalCase  
  Ví dụ: class OrderDetail,…

## Quy cách viết code:

* Giữa các tên biến, value, property và các kí tự “=”, “:”,.. phải có khoảng trắng  
  Ví dụ: var userName = “John”  
  var user = { name: “John” };
* Luôn nhớ sử dụng dấu chấm phẩy “;”, cả trong trường hợp không bị lỗi nếu không sử dụng dấu chấm phẩy thì cũng nên sử dụng. (trong javascript)
* Ngắt dòng code khi cần thiết, để dễ nhìn hơn.
* Thống nhất 1 cách đóng 1 code block

Ví dụ: trong javascript  
if ( myFunction() ) {  
 // Expressions  
} else if ( ( a &amp;&amp; b ) || c ) {  
 // Expressions  
} else {  
 // Expressions  
}

* Sử dụng tab trong sub code  
  Ví dụ:   
  elements  
   .addClass( 'foo' )  
   .children()  
   .html( 'hello' )  
   .end()  
   .appendTo( 'body' );
* Comment code: comment ở đầu file để đưa ra overview, hoặc ở những đoạn code phức tạp, khó hiểu, không nhất thiết phải ở tất cả các dòng code
  + Comment 1 dòng: sử dụng //
  + Comment nhiều dòng: sử dụng /\* \*/

# Handle Error và Logging:

## Handle Error:

* Lỗi cú pháp hầu hết có thể xảy ra trong giai đoạn phát triển ban đầu và là kết quả của sai cú pháp. Lỗi cú pháp có thể được bắt gặp và sửa chữa một cách dễ dàng khi chương trình được biên dịch để thực thi.
* Lỗi logic là kết quả của việc thực hiện logic không hợp lý. Phát hiện và sửa lỗi dựa trên specification, requirement và bug log từ tester.
* Lỗi runtime là các lỗi khi chúng ta không cân nhắc tất cả các trường hợp xảy ra. Thông thường sẽ sử dụng try catch trong code để handle.

## Logging:

Ghi log là để lưu lại quá trình thực thi của chương trình.

* Nếu ghi log cho lỗi (thường là lỗi runtime) thì khi chương trình có lỗi có thể nhìn được lỗi gì, ở đoạn code nào và dễ dàng trong việc tìm phương án xử lý. Nhiều trường hợp chương trình gặp lỗi trên môi trường production mà không thể tái hiện lại được, khi đó sẽ xem log để tìm nguyên nhân.
* Nếu ghi log cho quá trình sử dụng ứng dụng, sẽ biết user nào vào truy cập vào ứng dụng và thao tác gì, từ đó tìm được thủ phạm của những tác nhân gậy hại đến ứng dụng
* Nếu ghi log ở các đoạn code phức tạp, các tiến trình chạy ngầm, sẽ biết được trạng thái của tiến trình đó đang chạy như thế nào, biết được tốc độ xử lý của các nghiệp vụ phức tạp, có cơ sở để tìm ra các đoạn code gây cao tải cho hệ thống…
* …

Các cách ghi log:

* Print ra màn hình
* Ghi ra file lưu trên ổ cứng
* Insert log vào 1 bảng trong cơ sở dữ liệu...

Tùy vào ngôn ngữ và công nghệ sử dụng mà có các phương pháp ghi log khác nhau. Ví dụ như nodejs sẽ có winston hỗ trợ cho việc ghi log.

# Tool và IDE:

Visual Studio Code, Postman,...