<u>กฎข้อที่ 1 : การเริ่มต้นโครงการ:</u>

- ✓ กำหนดให้มีการจัดการประชุมกัน ระหว่างทีมพัฒนา และบุคคลที่เกี่ยวข้องเช่น SA, programmer, tester เพื่อทำความเข้าใจ และกำหนดข้อตกลงก่อนการเริ่มต้นโครงการเป็นต้น
- ✓ กำหนด environment ให้ตรงกันทั้งทีม เช่น กำหนดให้ใช้ PHP version เดียวกัน หรืออาจจะมี environment อื่นๆ ที่เกี่ยวข้อง
- ✓ กำหนดเครื่องมือในการพัฒนา หรือ tool ที่ใช้ในการพัฒนาให้เป็นแบบเดียวกันทั้งทีม เช่น ใช้ NetbeanIDE version 8.1 เป็น Editor ซึ่งแนะนำให้ใช้ เป็น NetbeanIDE version 8.1 เหมือนกันทั้งทีม เป็นต้น
- ✓ หากมีการใช้งาน LAMP stack กำหนดให้ติดตั้ง XAMPP เป็น LAMPP stack server ส่วน version ของ PHP, MySQL, APACHE ให้เป็นไปตามข้อตกลง ซึ่งแนะนำ PHP ตั้งแต่เวอร์ชั่น 5.6 ขึ้นไป MySQL version ตั้งแต่เวอร์ชั่น 5 ขึ้นไป, APACHE ตั้งแต่เวอร์ชั่น 2.4 ขึ้นไป
- ✓ หากเป็นงานเว็บแอปพลิเคชั่นแนะนำให้ใช้ PostgreSQL ตั้งแต่เวอร์ชั่น 9.3 ขึ้นไปเป็นฐานข้อมูล
- ✓ กำหนดให้ใช้ space bar ในการทำ indent ซึ่ง 1 indent = 4 space bar
- ✓ ไม่ให้ใช้ tab ในการ Indent source code เด็ดขาด
- ✓ กำหนดแผนงาน และ organization chart และบทบาทของแต่ละสมาชิกในทีมพัฒนา

<u>กฎข้อที่ 2 : การตั้งชื่อ Local site และ Virtual host:</u>

- ✓ PM หรือ project manager เป็นผู้กำหนดชื่อโครงการ และกำหนด production domain สำหรับแต่ละโครงการ
- ✓ กำหนดชื่อ local site เป็นชื่อเดียวกับชื่อ production domain ของแต่ละโครงการ โดยที่เปลี่ยน นามสกุลหรือประเภทของโดเมน เป็น .local เช่น โครงการมี URL ว่า **https://oacrmd-uat.zicure.com** ให้กำหนดชื่อ Local site ว่า **oacrmd-uat.zicure.local** เป็นต้น
- ✓ ชื่อ folder ที่เก็บ source code ของโครงการนั้นจะต้องชื่อเดียวกับชื่อ domain ซึ่งเป็น .local เช่น **uat.zicure.local** ซึ่งจะอยู่ใน document root อีกที เช่น **D:/xampp/htdocs/ uat.zicure.local** เป็นต้น
- ✓ กำหนดให้พัฒนาและทดสอบใน local site ให้เรียบร้อยก่อนที่นำจะขึ้นไปสู่ uat, production เป็นลำดับ ซึ่งขึ้นอยู่กับผู้บริหารของแต่ละโครงการเป็นผู้กำหนด วัน และเวลาของการ deploy ที่ระบุไว้แล้ว
- ✓ การ deploy สู่ production server นั้น ให้นำ uat ที่ลูกค้าตรวจสอบ และตรวจรับแล้วขึ้นสู่ production เท่านั้น

<u>กฎข้อที่ 3 : การตั้งชื่อฐานข้อมูล (Database):</u>

การตั้งชื่อฐานข้อมูล (database) จะต้องเป็นไปตามข้อกำหนดดังนี้ ซึ่งจะแบ่งแยกประเภทเป็นฐานข้อมูลของโครงการที่มี domain name หรือโครงการที่เป็น web application และโครงการที่ไม่ได้มี domain name หรือไม่ได้เป็น web application โดยมีรายละเอียดเป็นดังนี้

<u>ประเภทเว็บแอพพลิเคชั่น:</u> {domain-name}-web-{environment}

- ✓ **กำหนดเป็นชื่อเดียวกับชื่อ domain name ของโครงการ** โดยที่ไม่ต้องระบุ www และ ประเภทของ domain เช่น ระบบมี domain name เป็น **https://oacrmd-uat.zicure.com** ให้ตั้งชื่อฐานข้อมูลเป็น “**oacrmd-web-uat**” โดย **–uat** เป็นประเภทของการ deploy ระบบ ในที่นี้จะหมายถึง ฐานข้อมูลที่ข้อมูลได้มาจากการทดสอบใช้งานจากลูกค้า และอาจจะมี **–dev** ซึ่งจะหมายถึง ฐานข้อมูลสำหรับที่ใช้ในขั้นตอนพัฒนาโครงการเป็นต้น และ environment ที่เป็น production ไม่ต้องกำหนด **–production** ให้ใช้เพียง oacrmd-web
- ✓ ความยาวรวมกันไม่เกิน 15 ตัวอักษร
- ✓ จะต้องมีคำอธิบายหรือ comment ประกอบถึงจุดประสงค์ของฐานข้อมูลนี้

<u>ประเภทที่ไม่ใช่เว็บแอพพลิเคชั่น:</u> {project_short_name}-{platform}-{environment}

- ✓ **กำหนดเป็นชื่อเดียวกับชื่อของโครงการ** เช่นโครงการชื่อ oacrmd ให้ตั้งชื่อฐานข้อมูลว่า oacrmd-win-uat โดย **–uat** เป็นประเภทของการ deploy ระบบ ซึ่งหมายถึงทดสอบการใช้งานโดยลูกค้า อาจจะมี **–dev** หมายถึงสำหรับพัฒนาโครงการเป็นต้น และ environment ที่เป็น production ไม่ต้องกำหนด **–production** ให้ใช้เพียง oacrmd-win
- ✓ ความยาวรวมกันไม่เกิน 15 ตัวอักษร
- ✓ จะต้องมีคำอธิบายหรือ comment ประกอบถึงจุดประสงค์ของฐานข้อมูลนี้

กฎข้อที่ 4 : การตั้งชื่อ Schema:

- ✓ กำหนดด้วยภาษาอังกฤษเป็นตัวพิมพ์เล็กทั้งหมด และจะต้องสื่อความหมายที่ชัดเจนของกลุ่มงานของแต่ละ table นั้นๆ และอยู่ในรูปแบบ **{snake case}** เช่น schema ที่เป็นตัวแทนกลุ่มของ table ที่จัดการกับ transaction อาจจะตั้งชื่อว่า **trans** หรือ schema ที่เป็นตัวแทนกลุ่มของ table ที่จัดการกับข้อมูลพื้นฐาน อาจจะตั้งชื่อว่า **master** เป็นต้น
- ✓ เป็นเอกพจน์
- ✓ ขนาดความยาวรวมกัน ต้องไม่เกิน 15 ตัวอักษร
- ✓ จะต้องมีคำอธิบายหรือ comment ประกอบถึงจุดประสงค์ของ schema ทุก schema

กฎข้อที่ 5 : การตั้งชื่อตาราง (Table):

การตั้งชื่อตาราง (table) ในฐานข้อมูล จะต้องเป็นไปตามข้อกำหนดตามรายละเอียดดังนี้

- ✓ ตั้งชื่อเป็นภาษาอังกฤษตัวพิมพ์เล็กทั้งหมด และจะต้องสื่อความหมายชัดเจน และเป็น **{snake case}**
- ✓ ต้องเป็นพหูพจน์ ซึ่งพหูพจน์โดยทั่วไปให้เติม s หรือ es ตามหลักไวยกรณ์ ของภาษาอังกฤษ ต่อท้ายชื่อตาราง เช่น ตารางข้อมูลผู้เข้าใช้งาน (user) จะได้เป็น users และ ตารางเก็บคอมเมนต์ผู้ใช้งาน (comment) จะได้เป็น comments เป็นต้น
- ✓ ต้องมีความยาวรวมกันไม่เกิน 15 ตัวอักษร
- ✓ จะต้องมีคำอธิบายหรือ comment ประกอบถึงจุดประสงค์ของตารางข้อมูล หรือ table นั้นๆ ทุก table

กฎข้อที่ 6 : การตั้งชื่อฟิลด์ที่เป็นคีย์หลัก (Primary Key):

- ✓ การตั้งชื่อฟิลด์ที่เป็นคีย์หลัก (PK) ของแต่ละตารางให้ตั้งชื่อฟิลด์นั้นว่า **id** เสมอโดย datatype เป็นอะไรก็ได้ **แต่ต้อง unique ตามหลักการของ database integrity**

กฎข้อที่ 7 : การตั้งชื่อฟิลด์ที่เป็นคีย์รอง (Foreign Key):

- ✓ การตั้งชื่อฟิลด์ที่เป็นคีย์รอง (FK) ของแต่ละตารางให้ตั้งชื่อฟิลด์โดยใช้ชื่อ ตารางที่เป็นคีย์หลัก_id แต่ชื่อตารางที่เป็นคีย์หลักให้เปลี่ยนจากพหูพจน์เป็นเอกพจน์ด้วย เช่น user_id และ comment_id เป็นต้น

กฎข้อที่ 8 : การตั้งชื่อฟิลด์อื่น ๆ:

- ✓ ฟิลด์ที่เป็น list รายการใน select list จะต้องใช้ชื่อว่า name ส่วน data type นั้นจะเป็นอะก็ได้ ขึ้นอยู่กับความจำเป็นของแต่ละโครงการ
- ✓ ฟิลด์ที่เก็บผู้บันทึกข้อมูล หรือเพิ่มข้อมูล (insert) ให้ใช้ชื่อ create_uid ส่วนของ data type ให้เป็น bigint หรือให้สอดคล้องกับตาราง users
- ✓ ฟิลด์ที่เก็บผู้บันทึกข้อมูล หรือเพิ่มข้อมูล (Update) ให้ใช้ชื่อ update_uid ส่วนของ data type ให้เป็น bigint หรือให้สอดคล้องกับตาราง users
- ✓ ฟิลด์ที่เก็บวันที่ของการเพิ่มข้อมูลให้ชื่อว่า created ส่วนของ data type ให้กำหนดเป็น timestamp(6) without time zone NOT NULL
- ✓ ฟิลด์ที่เก็บวันที่ของการเพิ่มข้อมูลให้ชื่อว่า modified ส่วนของ data type ให้กำหนดเป็น timestamp(6) without time zone
- ✓ สื่อความหมายถึงจุดประสงค์ของฟิลด์ชัดเจนและถูกต้อง
- ✓ จะต้องมีคำอธิบายหรือ comment ประกอบถึงจุดประสงค์ของทุกๆ ฟิลด์
- ✓ ความยาวรวมกันต้องไม่เกิน 12 ตัวอักษร

กฎข้อที่ 9 : การตั้งชื่อไฟล์และ Class ของ Controller:

- ✓ การตั้งชื่อไฟล์ของ controller นั้นจะใช้กฎเดียวกันกับการตั้งชื่อตาราง และต้องเป็นไปในรูปแบบ capital camel case และต่อท้ายด้วยคำว่า **Controller.php** เช่น ตาราง users จะได้ ชื่อไฟล์ controller ดังนี้ **UsersController.php**, ตาราง comments จะได้ชื่อของ controller ดังนี้ **CommentsController.php** เป็นต้น
- ✓ ชื่อ controller class ต้องเป็นชื่อเดียวกับชื่อไฟล์และเป็น case sensitive แต่ไม่ต้องมี .php เช่น **class UsersController, class CommentsController** เป็นต้น
- ✓ ต้องกำหนดถึงจุดประสงค์ในการสร้าง controller นี้ก่อนที่จะมีการประกาศ class ด้วย document Comment เช่น

```
/**
 *
 * {Class Description}
 * @author  {writer}
 * @license  PAKGON
 * @since    {YYYY/MM/DD} as date of created of the class
 */
```

- ✓ เปิด php tag ด้วย <?php แต่ไม่ต้องกำหนด tag ปิด (?>) ให้ปิดด้วยการปิด class (}) เช่น

```
<?php
        class UserController extends AppController {
            ..................
        }
```

- ✓ ไฟล์ต้องใช้ encoding UTF-8 without BOM เท่านั้น
- ✓ กำหนดนามสกุลของไฟล์เป็น .php เช่น UsersController.php
- ✓ กำหนดให้จัดรูปแบบ source code ให้สวยงาม มีการกำหนดย่อหน้า และช่องไฟที่สวยงามดูเป็นระเบียบเรียบร้อย

<u>กฎข้อที่ 10 : การตั้งชื่อฟังก์ชั่นของ Controller:</u>

- ✓ สำหรับ function ที่มีหน้า view หรือส่วนแสดงผลให้ขึ้นต้นด้วย

```
public function functionName {
    ..........
}
```

- ✓ สำหรับ function ที่ไม่มีหน้า View หรือเป็นฟังก์ชั่นที่เรียกใช้กันใน controller เดียวกันให้ขึ้นต้นด้วย

```
private function functionName {
    ..........
}
```

- ✓ ให้ตั้งชื่อฟังชั่นก์เป็น **{camel case} สำหรับฟังก์ชั่นที่มี accessor เป็น public**
- ✓ ให้ตั้งชื่อฟังก์ชั่นนำหน้าด้วยขีดล่าง **_{camel case} สำหรับฟังก์ชั่นที่มี accessor เป็น private**
- ✓ สื่อความหมาย และจุดประสงค์ชัดเจน
- ✓ ความยาวรวมกันต้องไม่เกิน 30 ตัวอักษร
- ✓ ต้องกำหนดถึงจุดประสงค์ในการสร้าง function นี้ก่อนที่จะมีการประกาศ function ด้วย document comment เช่น

```
/**
 *
 * {Function Description}
 * @author  {writer}
 * @param      {type of parameter $xxxParam as a description}
 * @param      {type of parameter $yyyParam as a description}
 * @license    PAKGON
 * @since      {YYYY/MM/DD} as date of created of the class
 * @return     {return type}
 */
```

- ✓ กำหนดให้จัดรูปแบบ source code ให้สวยงาม มีการกำหนดย่อหน้า และช่องไฟที่สวยงามดูเป็นระเบียบเรียบร้อย
- ✓ หากมีการ set ค่าเพื่อส่งไปใช้งานที่ส่วนแสดงผล กำหนดให้ชื่อของตัวแปลที่จะกำหนดให้ส่วนแสดงผลใช้งาน จะต้องเป็นชื่อเดียวกับที่ใช้ใน controller เท่านั้น เช่น **$this->set('params', $params); หากมีการส่งข้อมูลจากหลายตัวแปล ให้กำหนดผ่าน $this->set(compact('params', 'users', 'comments')); เท่านั้น**

<u>กฎข้อที่ 11 : การประกาศตัวแปรและใช้งานตัวแปล:</u>

- ✓ การประกาศ property เพื่อใช้งานใน class ให้ใช้ accessor เป็น private และชื่ออยู่ในรูปแบบ **_{camel case}** ให้ใช้ accessor ที่เป็น public ได้ตามความเหมาะสมเท่านั้น
- ✓ ให้ประกาศตัวแปรเพื่อใช้งานภายในฟังก์ชั่นโดย ให้ใช้การตั้งชื่ออยู่ในรูปแบบ **{camel case}**
- ✓ ชื่อตัวแปลเป็นภาษาอังกฤษ และสื่อความหมายชัดเจน
- ✓ ความยาวรวมกันไม่เกิน 16 ตัวอักษร
- ✓ การใช้งาน **if, if else, if else if, for, while, do while, foreach** ภายใน Class, Controller, Model ให้ใช้งานแบบปรกติคือ เปิด และ ปิด ด้วย { .... }
- ✓ กำหนดให้เครื่องหมาย **"{"** อยู่บรรทัดเดียวกับการประกาศ **class, function, if, else, for, foreach, while, do, switch** และต้องมีการเว้นระยะ 1 space
- ✓ กำหนดให้เครื่องหมาย **"}" เมื่อจบ block การทำงานของแต่ละ block ให้เว้นบรรทัด 1 บรรทัด แล้วค่อยเติม "}" และจะต้องอยู่แนวเดียวกับตัวแรกของคำสั่ง**
- ✓ กำหนดให้ expression **ใน if, if else if, for, foreach, while จะต้องไม่มีช่องว่างระหว่าง เครื่องหมาย "(" และ ")"** หาก มีหลาย expression ให้เว้นว่าง 1 space และ ตามด้วย logical operand แล้วเว้นว่าง 1 space ตามด้วย expression ชุดที่ 2 เช่น

```
class UsersController extende AppController {
        ..............
        public function index() {
                ..............
                If(expression && expression) {
                        ..............
                }
                ..............
        }
}
```

- ✓ การใช้งาน if , if else แม้ภายใต้ if มีเพียงการทำงานเดียวก็ยังต้องใส่ {} เพื่อครอบคำสั่งภายใต้ if หรือ else ไว้
- ✓ ถ้ามีการประกาศตัวแปร Constant หรือค่าคงที่ให้ประการด้วยตัวอักษรพิมพ์ใหญ่หากมีหลายคำให้เชื่อมกันด้วย _ เท่านั้น เช่น VAT, VAT_RATE เป็นต้น
- ✓ หากเป็นตัวแปรประเภท array จะต้องระบุให้อยู่ในรูปแบบของพหูพจน์เท่านั้น คือเติม **s** และ **es** ตามหลักไวยากรณ์ ภาษาอังกฤษ

- ✓ กำหนดให้ใช้ space bar ในการทำ indent ซึ่ง 1 indent = 4 space bar
- ✓ ไม่ให้ใช้ tab ในการ Indent source code เด็ดขาด
- ✓ แต่ละ block การทำงาน ควรมีการทำ indent 1 indent
- ✓ code แต่ละบรรทัด ไม่ควรยาวเกิน 250 ตัวอักษรซึ่งนับรวม space bar และ indent ด้วย

กฎข้อที่ 12 : การตั้งชื่อไฟล์และ Class ของ model:

- ✓ การตั้งชื่อไฟล์ของ model นั้นจะใช้ชื่อเดียวกับชื่อของตาราง ที่ model ที่สร้างขึ้นเพื่อเป็นตัวแทนของตารางข้อมูลนั้น แต่จะอยู่ในรูปแบบที่เป็นเอกพจน์ และอยู่ในรูปแบบ **{capital camel case}** ในการตั้งชื่อไฟล์ เช่น **User.php, Comment.php** เป็นต้น
- ✓ ไฟล์ต้องใช้ encoding UTF-8 without BOM เท่านั้น
- ✓ ชื่อ Model Class ต้องเป็นชื่อเดียวกับชื่อไฟล์และจะต้องเป็น case sensitive แต่ไม่ต้องมี .php เช่น **class User**, **class Comment** เป็นต้น
- ✓ เปิด php tag ด้วย **<?php** แต่ไม่ต้องกำหนด tag ปิด (**?>**) ให้ปิดด้วยการปิด class (**}**)
- ✓ กำหนดนามสกุลของไฟล์เป็น .php เช่น User.php
- ✓ ต้องมีความยาวรวมกันไม่เกิน 15 ตัวอักษร

กฎข้อที่ 13 : การตั้งชื่อโฟลเดอร์ของ view:

- ✓ การตั้งชื่อโฟลเดอร์ของ view หรือส่วนแสดงผล ในการตั้งชื่อโฟลเดอร์ที่เก็บส่วนแสดงผลของแต่ละ controller ซึ่งการตั้งชื่อจะคือชื่อเดียวกับชื่อ class controller แต่ตัดส่วน suffix controller ออก เช่น users เป็นต้น
- ✓ ไฟล์ต้องใช้ encoding UTF-8 without BOM เท่านั้น
- ✓ การตั้งชื่อไฟล์ส่วนแสดงผลหรือ view ของต่ละ controller ให้ใช้ชื่อเดียวกับชื่อ function ที่เป็นเจ้าของ แต่ให้ตั้งชื่อไฟล์เป็นแบบ **{snake case}**
- ✓ กำหนดนามสกุลของไฟล์เป็น .ctp
- ✓ กำหนดให้ใช้งาน แท็กเปิดแบบเต็มเท่านั้น เช่น **<?php ไม่ให้ใช้แท็กแบบสั้น หรือ <? หรือ <?=**
- ✓ การใช้งาน if, if else, if else if, for, while, foreach ให้ใช้งานโดยการเปิดและปิดด้วยแท็ก
- ✓ การใช้งาน if , if else แม้ภายใต้ if มีเพียงการทำงานเดียวก็ยังต้องใส่ {} เพื่อครอบคำสั่งภายใต้ if หรือ else ไว้ เช่น

```
If(expression) :
        .....
endif;


for($i = 0; $i < 10; $i++):
        .....
endfor;
```

กฎข้อที่ 14 : การ Backup file และ folder:

- ✓ กำหนดชื่อไฟล์ที่ backup เป็นชื่อเดิมเพียงต่อท้ายด้วย **{_YYYYMMDD_HHII}** เช่น UsesController.php เมื่อ backup จะได้ชื่อเป็น UsersController_20171001_1345.php
- ✓ กำหนดที่อยู่ของไฟล์ที่ backup เป็น path เดิมกับไฟล์ต้นฉบับ

กฎข้อที่ 15 : การ Backup project:

- ✓ กำหนดชื่อไฟล์ backup เป็นชื่อเดียวกับชื่อ root project ของ โครงการ และต่อท้ายด้วย **{_YYYYMMDD_HHII}.source.zip หรือ {_YYYYMMDD_HHII}.source.tar.gz เท่านั้น**

กฎข้อที่ 16 : การ Backup database:

- ✓ กำหนดชื่อไฟล์ backup เป็นชื่อเดียวกับชื่อ database ของ โครงการ และต่อท้ายด้วย **{_YYYYMMDD_HHII}.database.zip หรือ {_YYYYMMDD_HHII}.database.backup หรือ {_YYYYMMDD_HHII}.database.tar.gz เท่านั้น**

กฎข้อที่ 17 : การใช้งาน GIT Version control:

- ✓ กำหนดให้ใช้งาน GIT เป็น version control เท่านั้น
- ✓ กำหนดให้แตก branch ต่างๆ เพื่อใช้งานในจุดประสงต่างๆ ดังนี้
  - **master:** จะเป็น branch ที่ตรงกับ production
  - **integration:** จะเป็น branch ของการ integration ของทุก feature branch
  - **uat:** จะเป็น branch สำหรับการทำ uat test จากทางฝั่งลูกค้า
  - **feature:** จะเป็น branch ของการพัฒนาซึ่งจะเป็น branch ที่แยกออกไปพัฒนา feature แต่ละ feature ของนักพัฒนาแต่ละคน ซึ่ง

กฎข้อที่ 18 : การใช้แตก Feature branch:

- ✓ นักพัฒนาแต่ละคนจะต้องทำงานที่ได้รับมอบหมายใน branch ของตัวเองเท่านั้นและมีกฎเกณฑ์ในการแตก branch โดยใช้ชื่อ branch ที่แตกออกไปเป็นชื่อเดียวกับชื่อเมล์ของตัวเองที่บริษัทตั้งให้ แต่ไม่ต้องมี @pakgon.com เช่น sarawutt.b@pakgon.com จะใช้ชื่อ branch ว่า sarawutt.b เป็นต้น
- ✓ เมื่อพัฒนา feature ของตนเองแล้วเสร็จ จะต้องทดสอบว่า feature ที่พัฒนาทำงานถูกต้องครบถ้วน ตรงตาม requirement และจะต้องไม่มี bug หรือ error หรือ notice เกิดขึ้น แล้วต้อง merge กับ branch integration

กฎข้อที่ 19 : การ Commit:

- ✓ กำหนดให้ เมื่อเลิกงานก่อนกลับบ้าน ควรมีการ commit code และ push code ขึ้น branch ของตัวเองทุกครั้งเพื่อกันเหตุการณ์ไม่พึงประสงค์
- ✓ การ commit ทุกครั้ง ให้มีการใส่หมายเหตุ เพื่อบ่งบอกถึงงานที่ได้ทำไปแล้ว ทุกครั้ง
- ✓ ไม่ให้มีการ commit โดยไม่มีการใส่คำอธิบายเป็นอันขาด

<u>กฎข้อที่ 20 : การ Merge:</u>

- ✓ เมื่อพัฒนา feature ของตนเองแล้วเสร็จ จะต้องทดสอบว่า feature ที่พัฒนาทำงานถูกต้อง ครบถ้วน ตรงตาม requirement และจะต้องไม่มี bug หรือ error หรือ notice เกิดขึ้น แล้วต้อง merge กับ branch integration
- ✓ ให้มีการ pull code จาก branch integration ลงมาที่ branch integration ก่อนทุกครั้ง ก่อนที่จะทำการ merge กับ code ของตัวเอง
- ✓ ทุกคนมีหน้าที่ต้อง merge code ใน feature branch ของตัวเองเข้ากับ integration

<u>กฎข้อที่ 21 : การ Version Reversion Tracking:</u>

- ✓ กำหนดให้ lead ของแต่ละโครงการเป็นคน จัดการ source code ที่ branch master, uat
  - การกำหนด tag ให้เป็นไปตามรูปแบบนี้
  - กำหนดให้ใช้งานเลขเวอร์ชั่น 3 หลัก โดยแต่ละหลักแทนความหมายดังนี้
    - **<u>major.minor.patch</u>**
      - major คือ phase ของโครงการที่พัฒนาซึ่งกำหนดตามสัญญาที่ทำกับลูกค้า และให้ PM ของโครงการเป็นคนกำหนด
      - minor คือ major update ซึ่งจะต้องอิงตาม major ตัวหน้า
      - patch คือ major หรือ minor patch และ bug fix ซึ่งจะต้องอิงตาม minor ตัวหน้า

  - **uat: UAT-v{major}.{minor}.{patch}** เช่น
    - UAT-v1.0 สำหรับ UAT version 1.0
    - UAT-v1.1 สำหรับ update ครั้งที่ 1
    - UAT-v1.1.1 สำหรับ patch ครั้งที่ 1 ของอัพเดทครั้งที่ 1
    - UAT-v1.12 สำหรับ อัพเดทครั้งที่ 12

  - **master: R-v{major}.{minor}.{patch}** เช่น
    - R-v1.0 สำหรับ production version 1.0
    - R-v1.1 สำหรับ update ครั้งที่ 1
    - R-v1.1.20 สำหรับ update ครั้งที่ 1 และ patch ครั้งที่ 20

References:

Camel case: คือลักษณะ การเขียนเพื่อแบ่งคำสองคำ โดยเขียนในรูปแบบตัวหนังสือภาษาอังกฤษตัวพิมพ์ใหญ่ และพิมพ์เล็กผสมกัน โดยที่ตัวหนังสือตัวแรกของของคำแรกจะเป็นตัวหนังสือภาษาอังกฤษพิมพ์เล็กหมด ส่วนคำที่เหลือ ให้ตัวหนังสือตัวแรกของแต่ละคำเป็นตัวหนังสือภาษาอังกฤษพิมพ์ใหญ่ส่วนตัวอื่นของคำเป็นตัวหนังสือภาษาอังกฤษตัวพิมพ์เล็กหมด เช่น userComment, userRole เป็นต้น

Capital Camel case: คือลักษณะ การเขียนเพื่อแบ่งคำสองคำ โดยเขียนในรูปแบบตัวหนังสือภาษาอังกฤษตัวพิมพ์ใหญ่ และพิมพ์เล็กผสมกัน โดยที่ตัวหนังสือตัวแรกของแต่ละคำเป็นตัวหนังสือภาษาอังกฤษพิมพ์ใหญ่ส่วนตัวอื่นของคำเป็นตัวหนังสือภาษาอังกฤษตัวพิมพ์เล็กหมด เช่น UserComment, UserRole เป็นต้น

Snake case: คือลักษณะ การเขียนเพื่อแบ่งคำสองคำโดยให้เขียนคำทั้งสองคำ โดยเขียนในรูปแบบตัวหนังสือภาษาอังกฤษตัวพิมพ์เล็กทั้งหมด และคั่นระหว่างแต่ละคำด้วยเครื่องหมายขีดล่าง (_) เช่น user_comment, user_role เป็นต้น