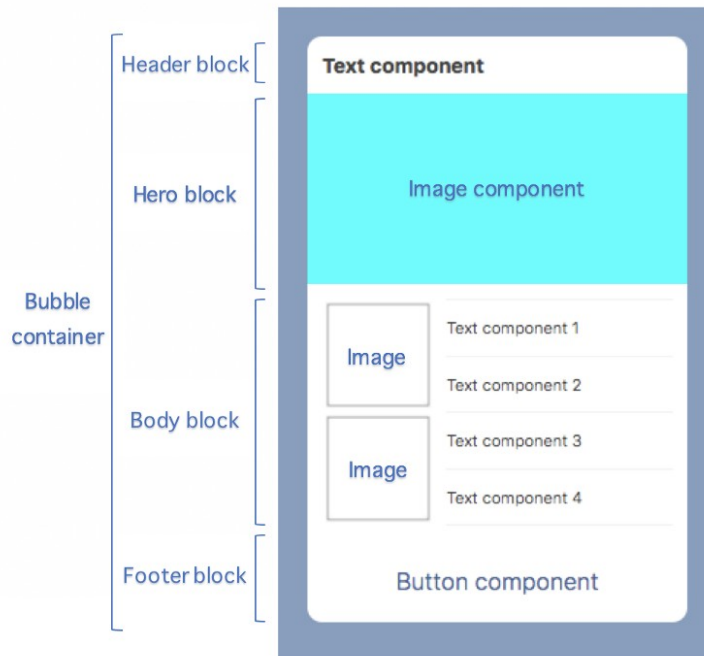


Flex Message

หมายเหตุ ใช้ร่วมกับ LINE Bot

1. องค์ประกอบ



รูปที่ 1.1 แสดงแผนภาพ องค์ประกอบ

1.1 Container

เป็นส่วนนอกสุดของ message มี 2 แบบด้วยกัน คือ

- bubble — เป็นการแสดง message เดียว
- Carousel — เป็นการแสดงหลายๆ message และสามารถเลื่อนได้

1.2 Block

เป็นส่วนข้างใน Container โดยมีองค์ประกอบต่างๆ ดังนี้

- Header — เป็น block ที่แสดงส่วนหัวของ message
- Hero — เป็น block ที่แสดง image หลัก
- Body — เป็น block ที่แสดง ข้อความหลัก
- Footer — เป็น block ที่แสดงส่วนด้านล่างของ message

1.3 Component

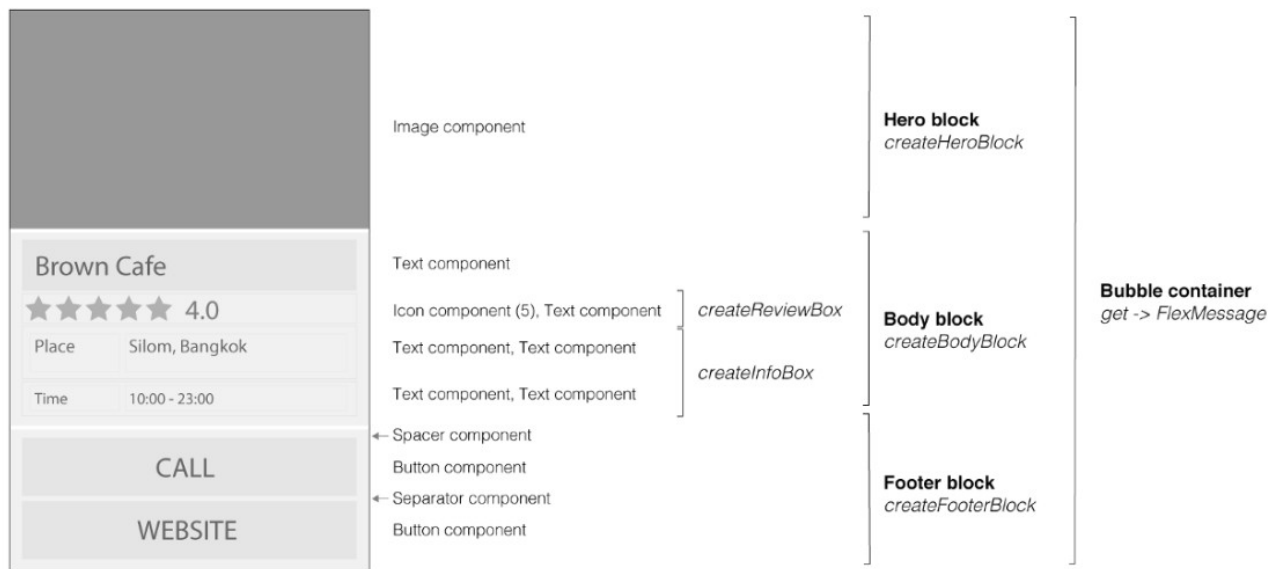
เป็นส่วนข้างในของแต่ละ block มี 8 แบบ คือ

- Button — เป็น component ที่แสดงปุ่มบน message
- Icon — เป็น component ที่แสดงไอคอนบน message
- Image — เป็น component ที่แสดงรูปภาพบน message

- Text — เป็น component ที่แสดงข้อความบน message
- Box — เป็น component ที่ระบุโครงสร้างของ component ลูก
- Filler — เป็น component ที่มองไม่เห็นที่มีความกว้างและความสูงเท่านั้น
- Separator — เป็น component ที่สร้างตัว separator
- Spacer — เป็น component ที่มองไม่เห็นที่สร้างส่วนว่างๆ

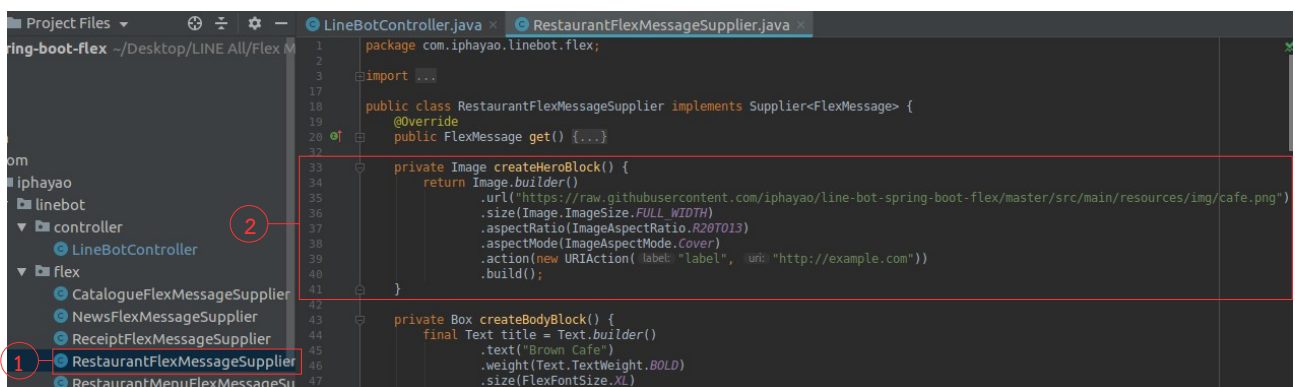
2. Restaurant Flex Message

โหลดโปรแกรม git clone <https://github.com/epsonthenic/Flex-Message.git>



รูปที่ 2.1 แสดงแผนภาพ องค์ประกอบของ Restaurant

2.1 ส่วนของ Hero block



รูปที่ 2.2 แสดงแผนภาพ ส่วนของ Hero block

2.1.1 สร้าง Class RestaurantFlexMessageSupplier implements Supplier<FlexMessage>

2.1.2 กำหนดให้เป็นรูปภาพ และ uri เมื่อกดที่รูป

2.2 ส่วนของ Body block

```
43 private Box createBodyBlock() {  
44     final Text title = Text.builder()  
45         .text("Brown Cafe")  
46         .weight(Text.TextWeight.BOLD)  
47         .size(FlexFontSize.XL)  
48         .build();  
49     final Box review = createReviewBox(); //ดาว  
50     final Box info = createInfoBox(); // Place และ Time  
51  
52     return Box.builder()  
53         .layout(FlexLayout.VERTICAL)  
54         .contents(asList(title, review, info))  
55         .build();  
56 }
```

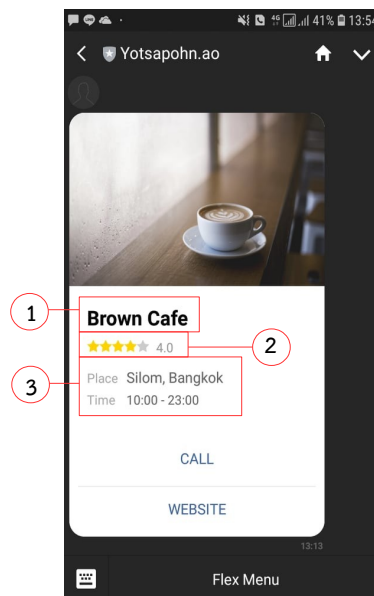
รูปที่ 2.3 แสดงแผนภาพ ส่วนของ Body block

2.2.1 สร้างคำว่า Brown Cafe ตัวหนาขนาด XL เก็บไว้ที่ title

2.2.2 เรียก Methods createReviewBox คือติดดาว เก็บไว้ที่ Object review

2.2.3 เรียก Methods createInfoBox คือ Place และ Time เก็บไว้ที่ Object info

2.2.4 return โดยเอา Object มารวมกัน



รูปที่ 2.4 แสดงแผนภาพ ตัวอย่างที่ออก

2.3 ส่วนของ Footer block

```
125 private Box createFooterBlock() {  
126     final Button callAction = Button.builder()  
127         .style(Button.ButtonStyle.LINK)  
128         .height(ButtonHeight.MEDIUM)  
129         .action(new URAction( label: "CALL", uri: "tel:00000"))  
130         .build();  
131     final Separator separator = Separator.builder().build();  
132     final Button websiteAction = Button.builder()  
133         .style(Button.ButtonStyle.LINK)  
134         .height(ButtonHeight.SMALL)  
135         .action(new URAction( label: "WEBSITE", uri: "https://example.com"))  
136         .build();  
137  
138     return Box.builder()  
139         .layout(FlexLayout.VERTICAL)  
140         .spacing(FlexMarginSize.SM)  
141         .contents(asList( callAction, separator, websiteAction))  
142         .build();  
143 }  
144
```

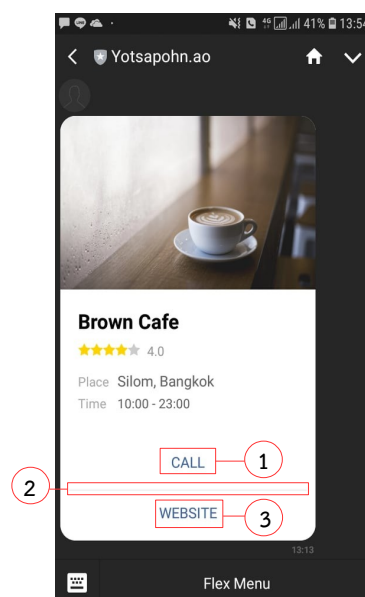
รูปที่ 2.5 แสดงแผนภาพ ส่วนของ Footer block

2.3.1 Object เมื่อกดปุ่ม CALL จะไปที่การโทร เบอร์ 00000

2.3.2 Object เส้นกั้น

2.3.3 Object เมื่อกดปุ่ม WEBSITE จะไปที่ หน้าเว็บ <https://example.com>

2.3.4 return โดย Object มารวมกัน



รูปที่ 2.6 แสดงแผนภาพ ตัวอย่างที่ออก

2.4 ส่วนของ get

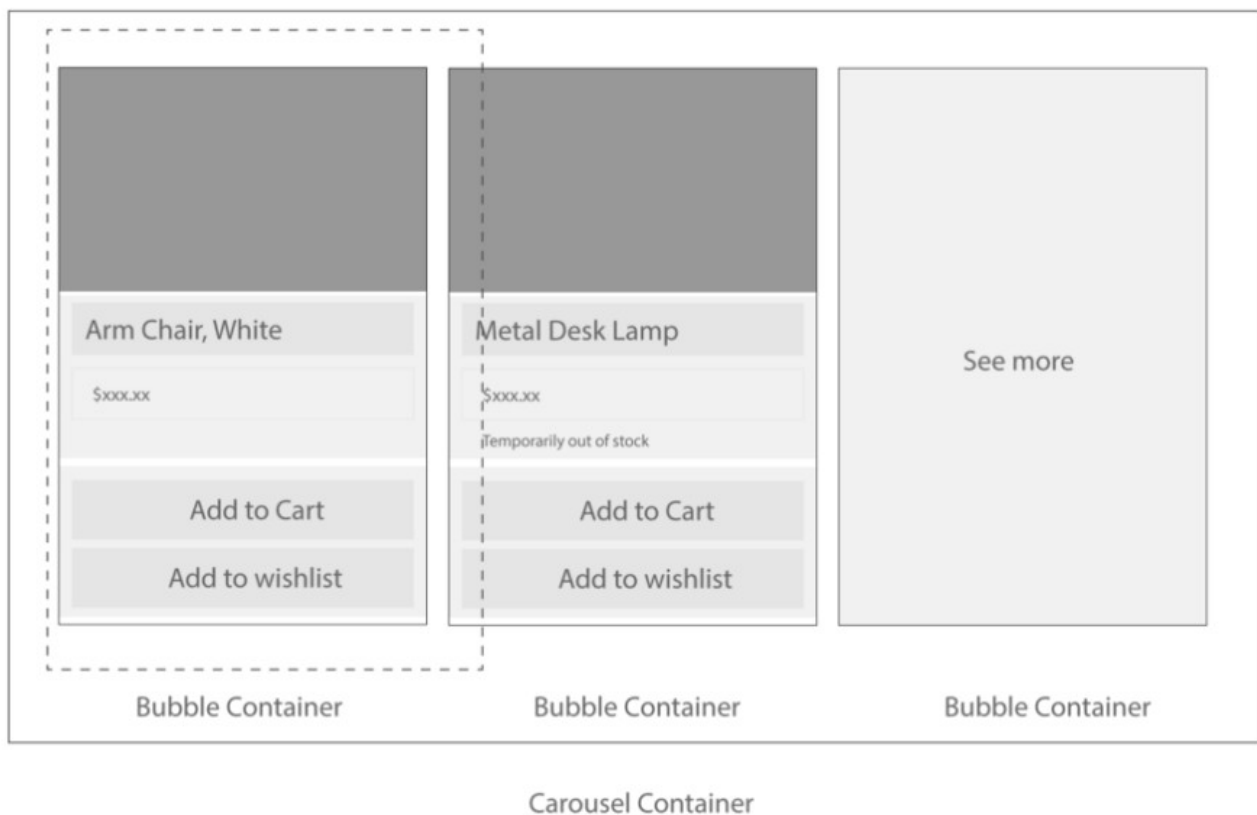
```
18 public class RestaurantFlexMessageSupplier implements Supplier<FlexMessage> {
19     @Override
20     public FlexMessage get() {
21         final Image heroBlock = createHeroBlock();
22         final Box bodyBlock = createBodyBlock();
23         final Box footerBlock = createFooterBlock();
24
25         final Bubble bubbleContainer = Bubble.builder()
26             .hero(heroBlock)
27             .body(bodyBlock)
28             .footer(footerBlock)
29             .build();
30         return new FlexMessage( altText: "Restaurant", bubbleContainer);
31     }
}
```

รูปที่ 2.7 แสดงแผนภาพ ส่วนของ get

2.4.1 นำ Methods มาเก็บไว้ที่ Object จะเป็นค่าที่ return มาจาก Methods นั้น

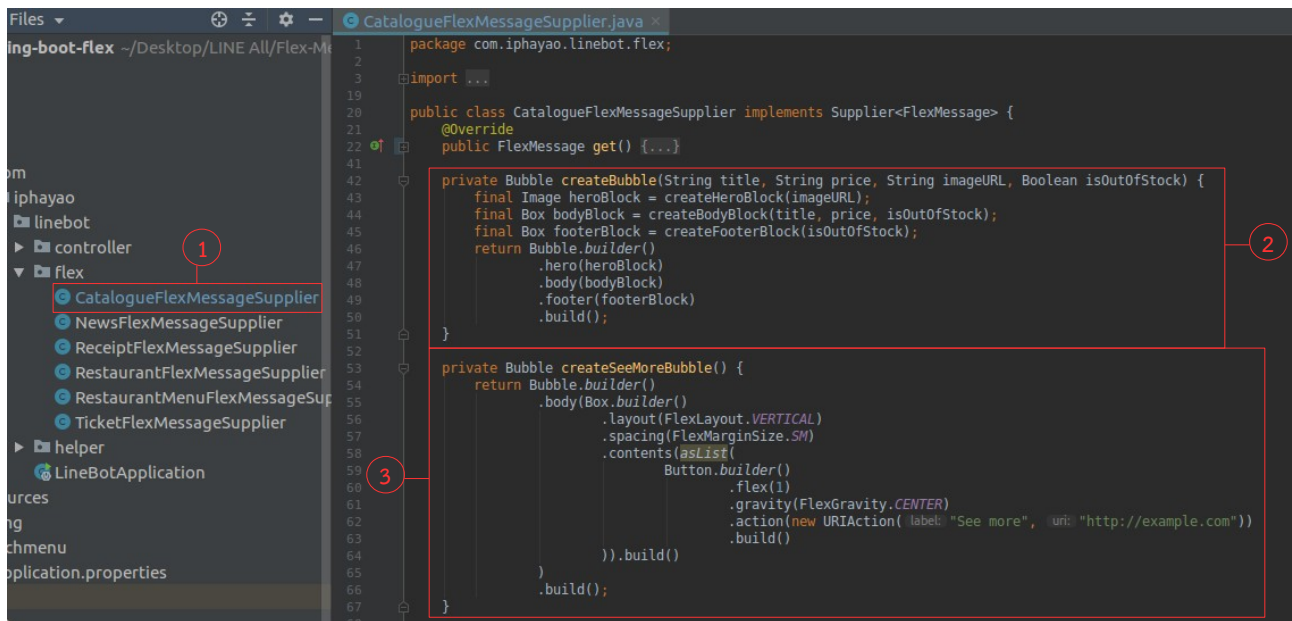
2.4.2 นำ Object ที่ได้เก็บมาประกอบกันให้เป็น hero, body, footer และ ไม่มีheader return Object กลับไปแสดง

3. Catalogue Flex Message



รูปที่ 3.1 แสดงแผนภาพ องค์ประกอบของ Catalogue

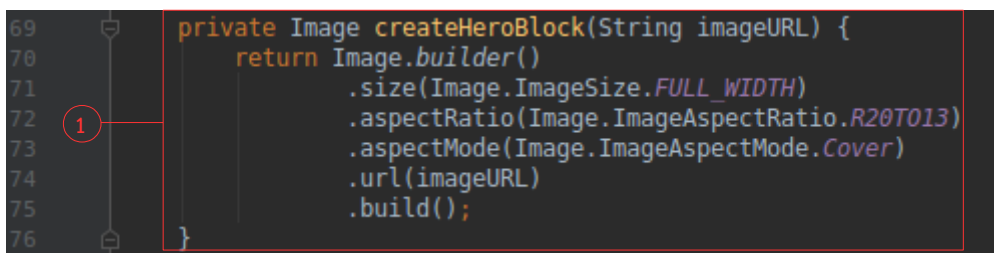
3.1 ส่วนของ Bubble Container



รูปที่ 3.2 แสดงแผนภาพ องค์ประกอบของ Catalogue

- 3.1.1 สร้าง Class CatalogueFlexMessageSupplier implements Supplier<FlexMessage>
- 3.1.2 สร้าง Bubble Container ถ้าใช้ Form เดียวกันสร้างอันเดียว มี hero, body และ footer
- 3.1.3 สร้าง Bubble Container เป็นอีก Form มี body

3.2 ส่วนของ Hero block



รูปที่ 3.3 แสดงแผนภาพ ส่วนของ Hero block

- 3.2.1 รวรับค่าจาก Methods get รับเป็นรูปภาพแล้ว return ค่ากลับ

3.3 ส่วนของ Body block

```
78 @ private Box createBodyBlock(String title, String price, Boolean isOutOfStock) {
79     final Text titleBlock = Text.builder()
80         .text(title)
81         .wrap(true)
82         .weight(Text.TextWeight.BOLD)
83         .size(FlexFontSize.XL).build();
84     final Box priceBlock = Box.builder()
85         .layout(FlexLayout.BASELINE)
86         .contents(asList(
87             Text.builder().text("$" + price)
88                 .wrap(true)
89                 .weight(Text.TextWeight.BOLD)
90                 .size(FlexFontSize.XL)
91                 .flex(0)
92                 .build()
93         ))).build();
94     final Text outOfStock = Text.builder()
95         .text("Temporarily out of stock")
96         .wrap(true)
97         .size(FlexFontSize.XXS)
98         .margin(FlexMarginSize.MD)
99         .color("#FF5551")
100         .build();
101
102     List<FlexComponent> listComponent = new ArrayList<>(Arrays.asList(titleBlock, priceBlock));
103     if(isOutOfStock) {
104         listComponent.add(outOfStock);
105     }
106
107     return Box.builder()
108         .layout(FlexLayout.VERTICAL)
109         .spacing(FlexMarginSize.SM)
110         .contents(listComponent)
111         .build();
112 }
113
```



รูปที่ 3.4 แสดงแผนภาพ ส่วนของ Body block

3.3.1 รอรับค่าจาก Methods get รับเป็นข้อความแล้วเก็บไว้ที่ Object titleBlock

3.3.2 รอรับค่าจาก Methods get รับเป็นข้อความโดยเอา \$ บวกกับข้อความแล้วเก็บไว้ที่ Object titleBlock

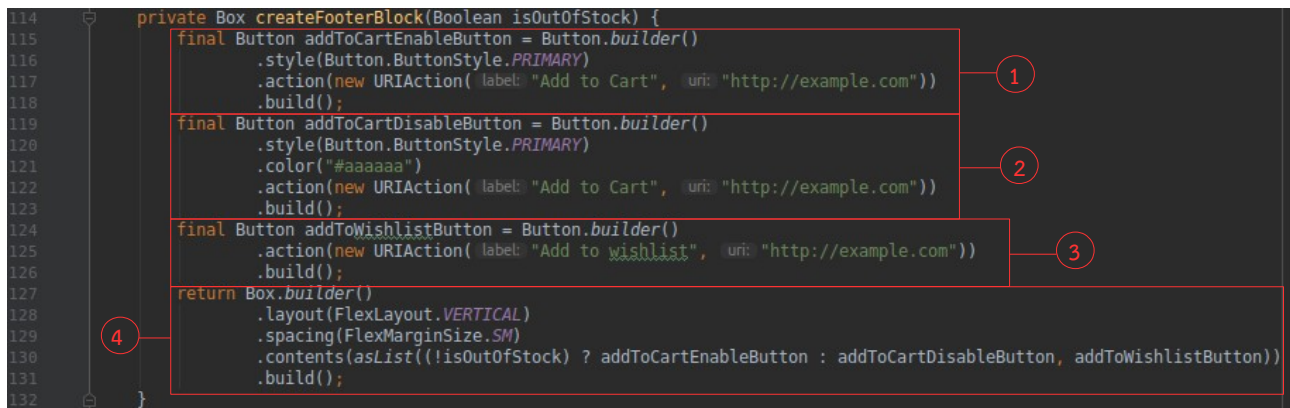
3.3.3 สร้าง Object outOfStock เป็นข้อความ Temporarily out of stock สีแดง

3.3.4 สร้าง ArrayList จากนั้นสร้างเงื่อนไขถ้า isOutOfStock ที่ได้จาก Methods get เท่ากับ true ให้เพิ่ม Object outOfStock ลงไป ArrayList

3.3.5 return โดย Object มารวมกัน

3.4 ส่วนของ Footer block

```
114 private Box createFooterBlock(Boolean isOutOfStock) {
115     final Button addToCartEnableButton = Button.builder()
116         .style(Button.ButtonStyle.PRIMARY)
117         .action(new URIAction( label: "Add to Cart", uri: "http://example.com"))
118         .build();
119     final Button addToCartDisableButton = Button.builder()
120         .style(Button.ButtonStyle.PRIMARY)
121         .color("#aaaaaa")
122         .action(new URIAction( label: "Add to Cart", uri: "http://example.com"))
123         .build();
124     final Button addToWishlistButton = Button.builder()
125         .action(new URIAction( label: "Add to wishlist", uri: "http://example.com"))
126         .build();
127     return Box.builder()
128         .layout(FlexLayout.VERTICAL)
129         .spacing(FlexMarginSize.SM)
130         .contents(asList(!isOutOfStock ? addToCartEnableButton : addToCartDisableButton, addToWishlistButton))
131         .build();
132 }
```



The diagram shows the code structure of the `createFooterBlock` method. Red boxes and numbers highlight specific parts: 1 points to the `addToCartEnableButton` creation, 2 points to the `addToCartDisableButton` creation, 3 points to the `addToWishlistButton` creation, and 4 points to the final `return` statement where the buttons are combined into a `Box`.

รูปที่ 3.5 แสดงแผนภาพ ส่วนของ Footer block

3.4.1 สร้าง Object `outOfStock` เป็นปุ่ม Add to Cart เมื่อกดลิงก์ไปที่ `http://example.com`

3.4.2 สร้าง Object `outOfStock` เป็นปุ่ม Add to Cart สีเทา เมื่อกดลิงก์ไปที่

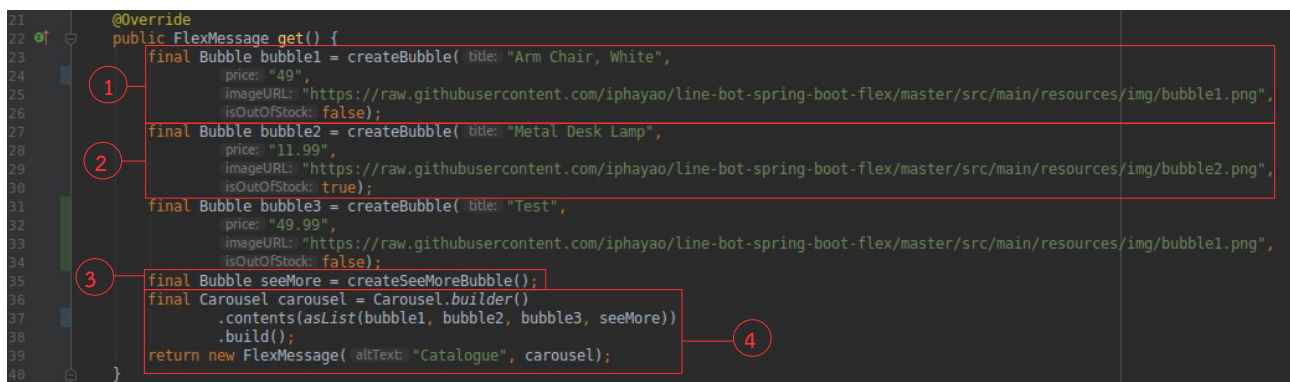
`http://example.com`

3.4.3 สร้าง Object `outOfStock` เป็นปุ่ม Add to wishlist เมื่อกดลิงก์ไปที่ `http://example.com`

3.4.4 return โดย Object มารวมกัน ถ้า `isOutOfStock` ถ้าเป็น false ให้ใช้ Object `addToCartEnableButton`

3.5 ส่วนของ get

```
11 @Override
12 of public FlexMessage get() {
13     final Bubble bubble1 = createBubble( title: "Arm Chair, White",
14         price: "49",
15         imageUrl: "https://raw.githubusercontent.com/iphayao/line-bot-spring-boot-flex/master/src/main/resources/img/bubble1.png",
16         isOutOfStock: false);
17     final Bubble bubble2 = createBubble( title: "Metal Desk Lamp",
18         price: "11.99",
19         imageUrl: "https://raw.githubusercontent.com/iphayao/line-bot-spring-boot-flex/master/src/main/resources/img/bubble2.png",
20         isOutOfStock: true);
21     final Bubble bubble3 = createBubble( title: "Test",
22         price: "49.99",
23         imageUrl: "https://raw.githubusercontent.com/iphayao/line-bot-spring-boot-flex/master/src/main/resources/img/bubble1.png",
24         isOutOfStock: false);
25     final Bubble seeMore = createSeeMoreBubble();
26     final Carousel carousel = Carousel.builder()
27         .contents(asList(bubble1, bubble2, bubble3, seeMore))
28         .build();
29     return new FlexMessage( altText: "Catalogue", carousel);
30 }
```



The diagram shows the code structure of the `get` method. Red boxes and numbers highlight specific parts: 1 points to the creation of `bubble1`, 2 points to the creation of `bubble2`, 3 points to the creation of `bubble3` and `seeMore`, and 4 points to the final `return` statement where the carousel is built and returned.

รูปที่ 3.6 แสดงแผนภาพ ส่วนของ get

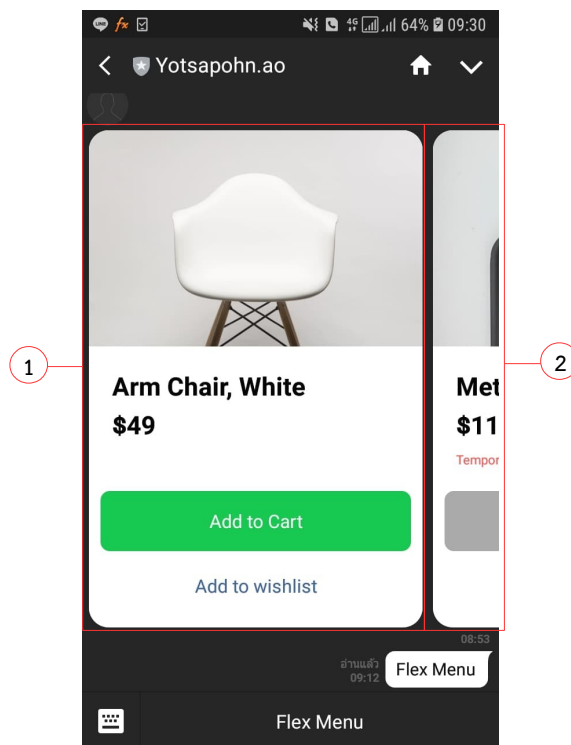
3.5.1 นำ Methods ที่กำหนดค่ามาเก็บไว้ที่ Object จะเป็นค่าที่ return มาจาก Methods นั้น

3.5.2 นำ Methods ที่กำหนดค่ามาเก็บไว้ที่ Object จะเป็นค่าที่ return มาจาก Methods นั้น

3.5.3 นำ Methods มาเก็บไว้ที่ Object จะเป็นค่าที่ return มาจาก Methods นั้น

3.5.4 นำ Object ที่ได้เก็บมาประกอบกัน return Object กลับไปแสดง

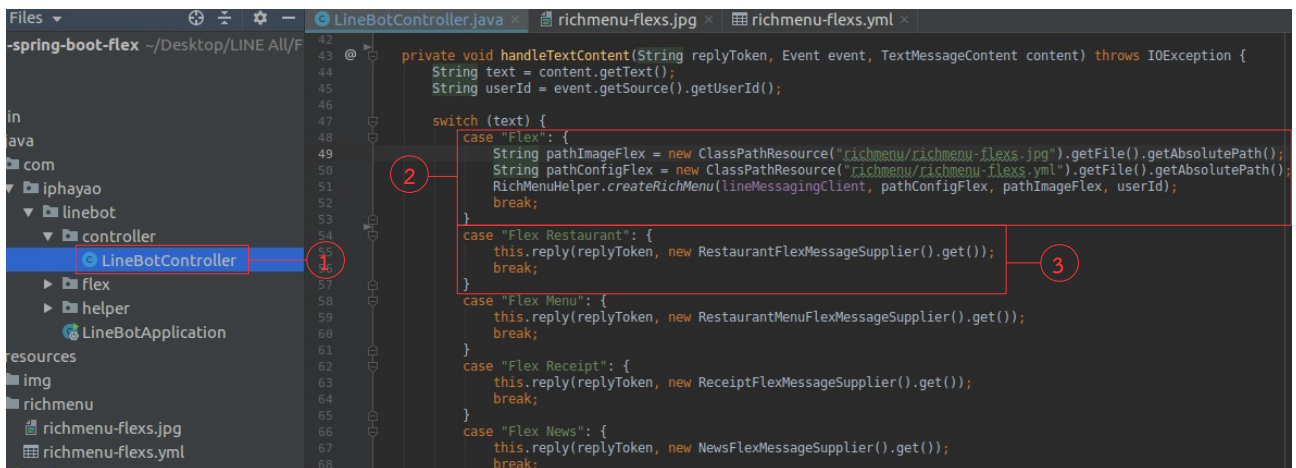
3.6 ตัวอย่างที่ออก



รูปที่ 3.6 แสดงแผนภาพ แสดงแผนภาพ ตัวอย่างที่ออก

4. เรียก Menu

4.1 เหตุการณ์ข้อความ LINE Bot



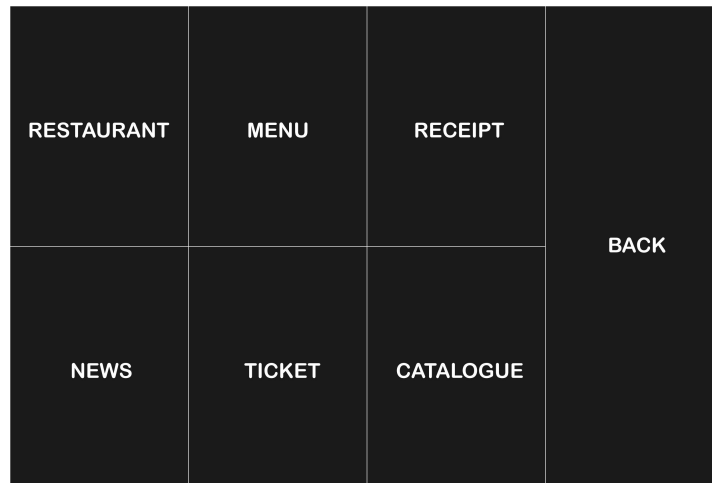
รูปที่ 4.1 แสดงแผนภาพ Controller เหตุการณ์ข้อความ LINE Bot

4.1.1 สร้าง Class LineBotController แล้วทำให้เป็น LINE Bot เหตุการณ์รับข้อความ

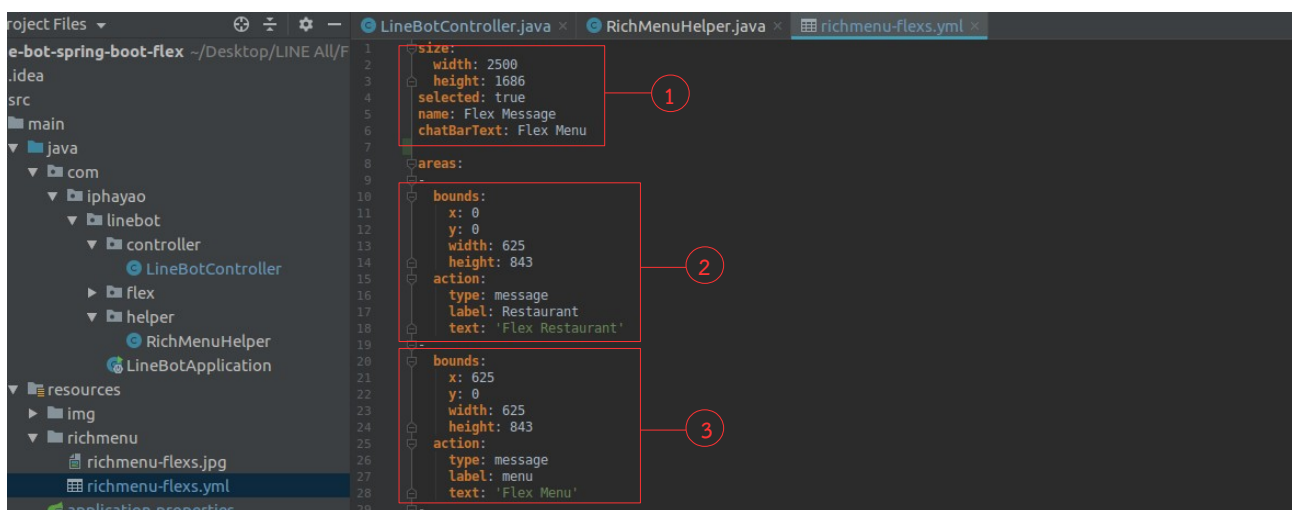
4.1.2 เมื่อผู้ใช้พิมพ์ Flex LINE Bot จะทำการดึงรูปเก็บไว้ที่ pathImageFlex และ ไฟล์ Menu เก็บไว้ที่ pathConfigFlex จากนั้นส่งค่าไปที่ Class RichMenuHelper Method createRichMenu โดยมี Menu, รูปภาพ, และ id LINE

4.1.3 เมื่อกดที่ Menu RESTAURANT จะแสดงหน้า RestaurantFlexMessageSupplier ออกทาง LINE

4.2 ออกแบบ Menu Code ข้อดี กำหนดได้ตามใจชอบ, ข้อเสีย ต้องมีการเรียกใช้ Menu



รูปที่ 4.2 แสดงแผนภาพ Menu มี 7 Menu ขนาด กว้าง:2500 สูง:1686

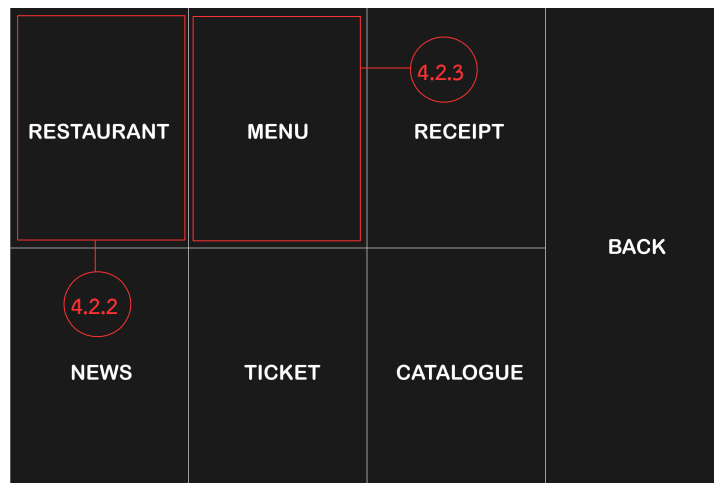


รูปที่ 4.3 แสดงแผนภาพ ออกแบบ Menu

4.2.1 ขนาดของรูปภาพ กว้าง:2500 ยาว:1686

4.2.2 X ตำแหน่งที่ 0 Y ตำแหน่งที่ 0 กว้าง:625 สูง:843 จากนั้นจะทำการส่งคำว่า Flex Restaurant ให้กับ LINE Bot

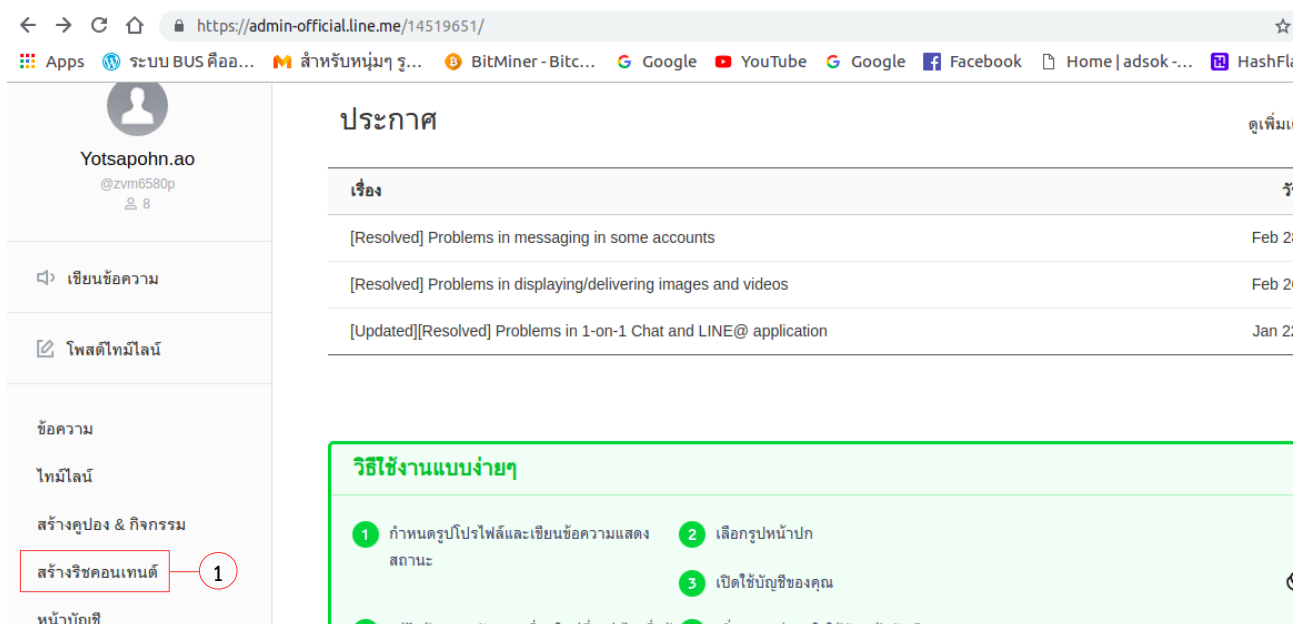
4.2.3 X ตำแหน่งที่ 625 Y ตำแหน่งที่ 0 กว้าง:625 สูง:843 จากนั้นจะทำการส่งคำว่า Flex Menu ให้กับ LINE Bot



รูปที่ 4.4 แสดงแผนภาพ ออกแบบ Menu

4.3 ออกแบบ Menu ข้อดี เรียกใช้ Menu ตั้งแต่เริ่ม ข้อเสีย ต้องใช้แบบฟอร์มของทาง LINE

4.3.1 <https://admin-official.line.me>



รูปที่ 4.5 แสดงแผนภาพ ออกแบบ Menu

4.3.2 กดไปที่ สร้างรื้ชคอนแทนด์ และ สร้างใหม่

ริชเมนู

คุณสามารถแสดงริชเมนูที่ห้องแชท แล้วส่งข้อความตอบกลับแบบคีย์เวิร์ดหรือแสดงข้อมูลสำคัญต่างๆ

การแสดงผลข้อมูล

ทั้งหมด

ระยะเวลา

YYYY-MM-DD

00

~

YYYY-MM-DD

00

ค้นหา

ลบ

1

สร้างใหม่

ชื่อ	เทมเพลต	ลิงก์	ระยะเวลาแสดง	การแสดงผลข้อมูล
------	---------	-------	--------------	-----------------

รูปที่ 4.5 แสดงแผนภาพ ออกแบบ Menu

4.3.2 กดไปที่ สร้างรื้ชคอนแทนด์ และ สร้างใหม่

การแสดงผลข้อมูล	<input type="radio"/> ไม่แสดง <input checked="" type="radio"/> แสดง 1
ระยะเวลาแสดง	2 2019-03-08 00 : 00 ~ 2023-04-29 00 : 00 3
ชื่อ	3 Test 4/30
เมนูในห้องแชท	<input checked="" type="radio"/> ดูรายละเอียด <input type="radio"/> <div>กำหนดชื่อเมนูเพื่อแสดงในห้องแชท</div>
การแสดงผลเมนูแบบเริ่มต้น	<input checked="" type="radio"/> แสดง <input type="radio"/> ไม่แสดง
เลือกเทมเพลต	<input checked="" type="radio"/> ใช้รูป <input type="radio"/> ใช้ข้อความ + ไอคอน

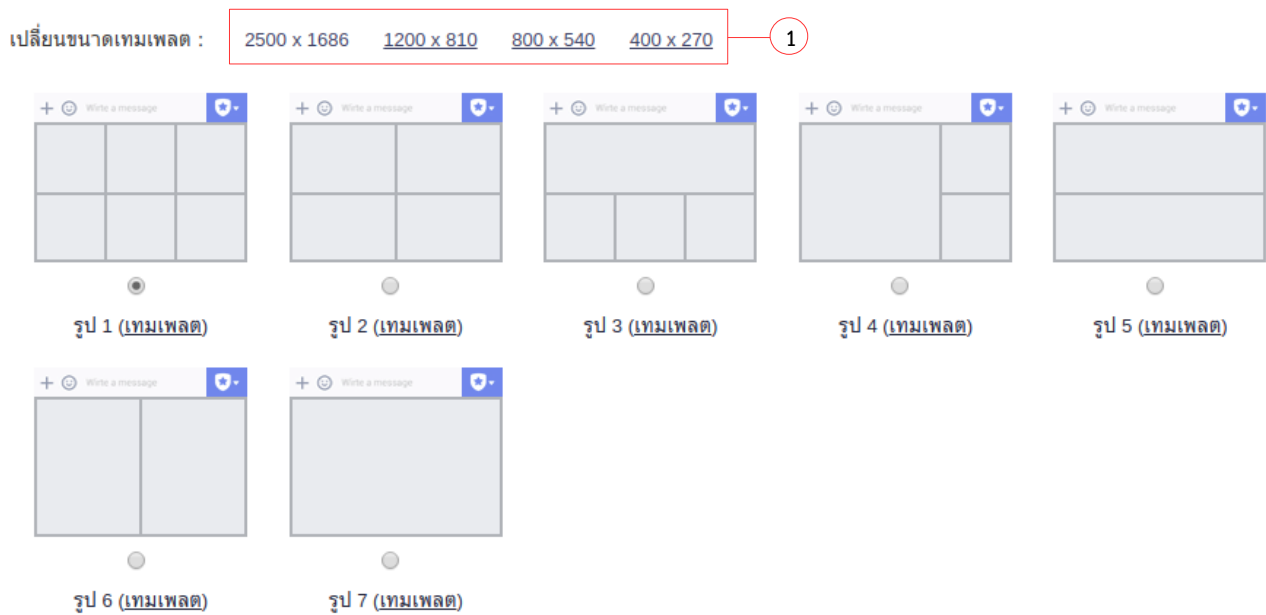
รูปที่ 4.6 แสดงแผนภาพ ออกแบบ Menu

4.3.2.1 เปิดการทำงาน

4.3.2.2 กำหนดระยะเวลาการใช้งาน

4.3.2.3 ตั้งชื่อ Menu

4.3.3 เลือกขนาดของ Menu



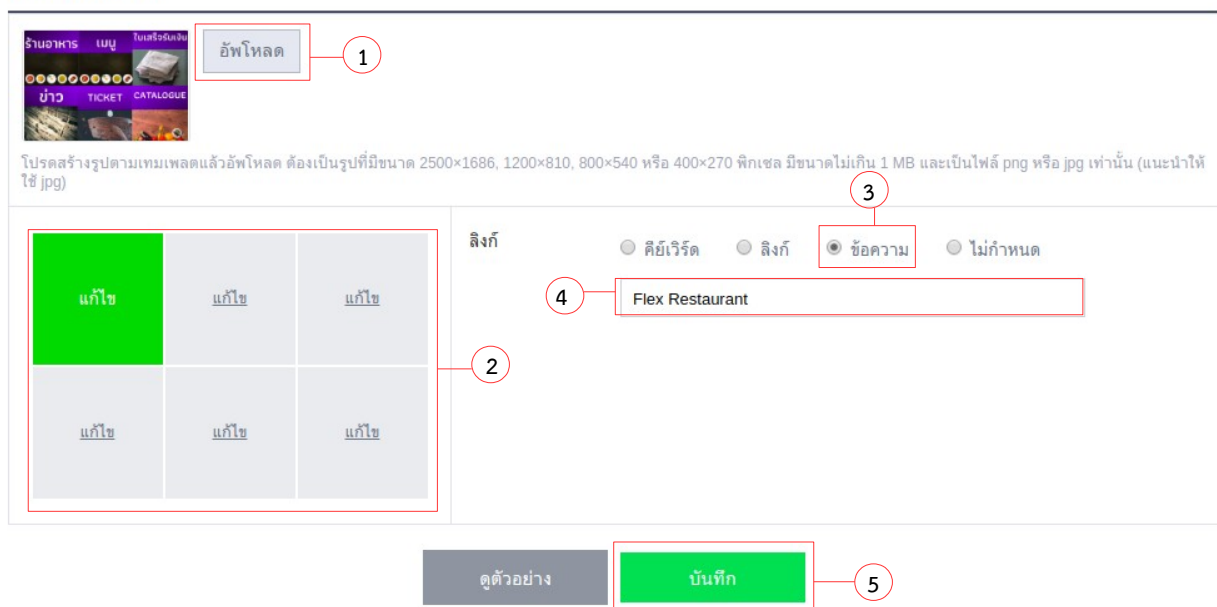
รูปที่ 4.7 แสดงแผนภาพ ออกแบบ Menu

4.3.3.1 เลือกขนาด และ รูปแบบ

4.3.4 เลือกขนาดของ Menu

เว็บปีบอัดรูป <https://www.iloveimg.com/th/compress-image/compress-jpg>

ตั้งค่าคอนเทนต์



รูปที่ 4.8 แสดงแผนภาพ ออกแบบ Menu

4.3.4.1 อัปโหลดรูปภาพ ขนาด 2500 x 1686 ขนาดไม่เกิน 1M

4.3.4.2 เลือกช่องที่จะกด โดยแบ่งช่องละ 833.34 x 843

4.3.4.3 เลือกที่ข้อความ

4.3.4.4 ข้อความที่ตรงกับโปรแกรม

4.3.4.5 บันทึก