

รายงาน
เรื่อง Hotel Flipper

จัดทำโดย

63010026	นางสาวกฤตศยา	นทีมณฑล
63010120	นายจรัสรวี	ศรีจันทร์สุข
63010159	นางสาวจุไรรัตน์	แดงพวงไพบูลย์
63010295	นางสาวณัฐธนิชา	อาจสุวรรณ
63010339	นางสาวณัฐวดี	ดิณภูมิ
63010354	นายดิษฐพงษ์	จรัสชัยโรจน์
63010357	นายเดชาชาญ	บุญกล้า

เสนอ

รายงานนี้เป็นส่วนหนึ่งของวิชา 01076024 SOFTWARE ARCHITECTURE AND DESIGN

คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 1/2565

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ที่มาของโครงการ

เนื่องจากในปัจจุบันประเทศไทยเป็นประเทศแห่งการท่องเที่ยว จึงมีการก่อตั้งโรงแรมเพื่อรองรับจำนวนนักท่องเที่ยวที่มากขึ้น แต่เพื่อให้มั่นใจว่าโรงแรมนั้นได้มาตรฐานตามกฎหมาย โรงแรมหลายแห่งนั้นจึงต้องมีการบริการต่างๆ ให้ครบครัน และจากที่ทราบหลายๆ โรงแรมไม่มีความสามารถมากพอที่จะจ้างพนักงานประจำ บางโรงแรมมองว่าเพื่อเป็นการช่วยลดค่าใช้จ่ายในระยะยาว การจ้างพนักงานประจำในการทำความสะอาด และบำรุงรักษา ฯลฯ นับว่าเป็นการสิ้นเปลืองโดยใช้เหตุ แต่หากเป็นการจ้างเพียงครั้งคราวจะสามารถช่วยแก้ปัญหาในส่วนนี้ได้ จึงเกิดบริษัทหลายแห่งที่จะช่วยอำนวยความสะดวกให้กับโรงแรมโดยเฉพาะ

ขอบเขตของการศึกษาค้นคว้า

ทางทีมผู้พัฒนา จึงตัดสินใจที่จะพัฒนา Web Application ที่ทำให้ทางโรงแรมสามารถจัดหาพนักงานที่มีความชำนาญในแต่ละด้านมาช่วยในงานต่างๆ เช่น การทำความสะอาดโรงแรมและห้องภายในโรงแรม ฯ ซึ่งเหมาะสำหรับโรงแรมที่ประสบปัญหาในการจ้างพนักงานประจำในการทำความสะอาด บำรุงรักษา และอื่นๆ อีกทั้งยังรับประกันความปลอดภัย ป้องกันความเสียหายและการให้บริการที่มีมาตรฐาน โดยที่ Web Application จะมีระบบการทำงานหลัก ๆ ดังนี้

1. ระบบการให้บริการ

1.1 ราคาของการให้บริการ

- แบ่งตามประเภทของงาน
- แบ่งตามเวลาที่ให้บริการ

1.2 ระยะเวลาในการรับบริการ

1.3 ประเภทของการให้บริการ

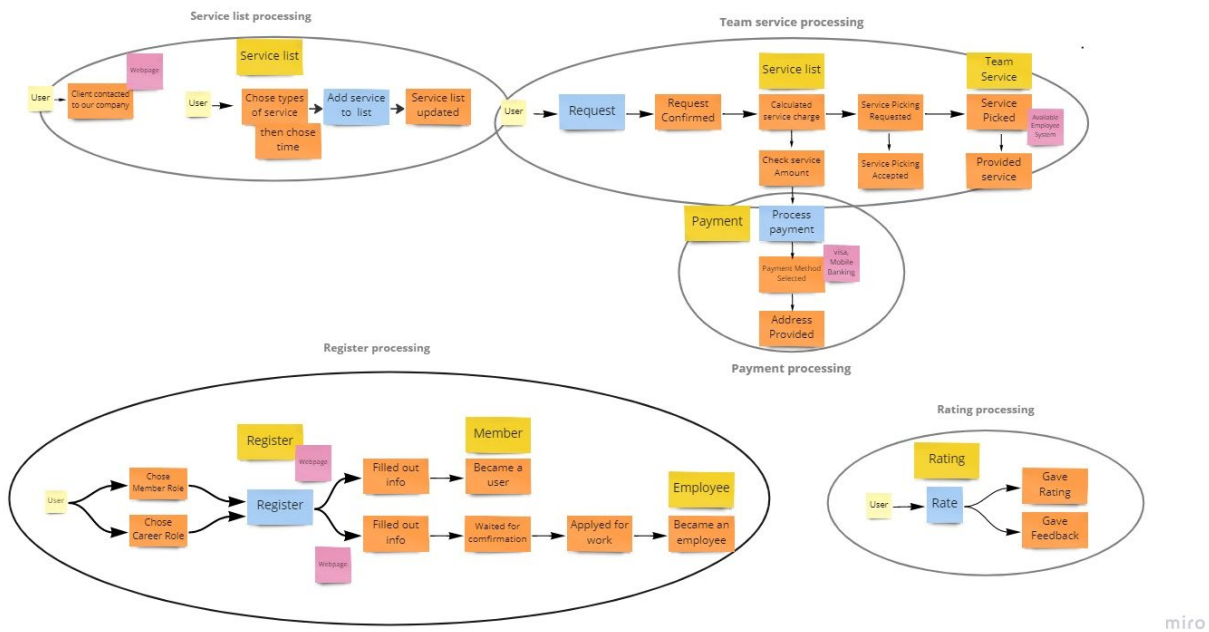
- ทำความสะอาดทั่วไป ได้แก่ ทำความสะอาดห้องพักและทำความสะอาดห้องน้ำ
- บริการทำความสะอาดเครื่องปรับอากาศ
- บริการซักชุดเครื่องนอนและกำจัดไรฝุ่น
- บริการซ่อมบำรุงและอื่น ๆ ได้แก่ ซ่อมเครื่องปรับอากาศ , ติดตั้งและย้ายเครื่องปรับอากาศ , ทำสวน ตกแต่งสวนและจัดสวน , ซ่อมเครื่องใช้ไฟฟ้าต่าง ๆ เช่น เครื่องปรับอากาศ เป็นต้น
- บริการพนักงานรักษาความปลอดภัย

2. ระบบ Feedback, Rating

3. ระบบการสมัครสมาชิกและการสมัครเป็นผู้ให้บริการ

4. ระบบชำระเงิน

Bounded Context



รูปที่ 1 Bounded Context

คือขอบเขตของ Domain ที่รวบรวมกติกาต่าง ๆ และข้อมูลที่เกี่ยวข้องเพื่อการทำงานเพื่อการทำงานของ Domain นั้น ๆ โดยมีทั้งหมด 5 Bounded Context ได้แก่

1. Register processing
2. Service list processing
3. Team service processing
4. Payment processing
5. Rating processing

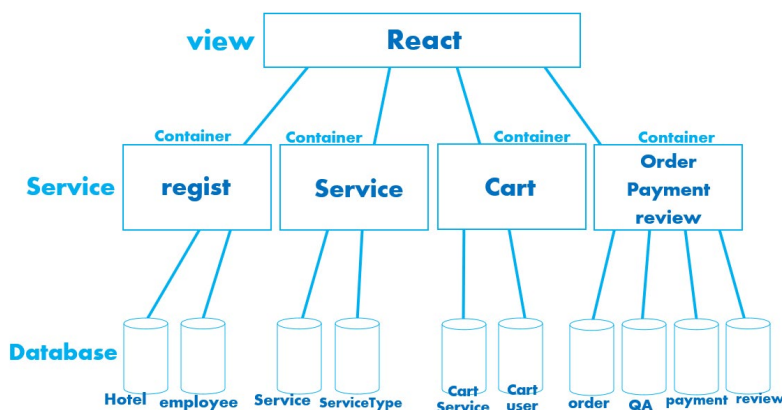
Framework

Back-end เฟรมเวิร์กที่เลือกใช้ได้แก่ Nest.js ซึ่งเป็นเฟรมเวิร์กสำหรับการเขียนโปรแกรมเชิงวัตถุ (OOP) เนื่องจากมีชุดไลบรารีแบบ Dependency Injection (DI) ซึ่งเป็นไลบรารีเฉพาะตัว ทำให้เราไม่ต้องไปหาไลบรารีเพิ่มเติมเองทั้งหมด จึงทำให้เราวางโครงสร้างโค้ดและเขียนโค้ดได้อย่างมีระบบ มีรูปแบบที่ชัดเจน อีกทั้งยังไม่ต้องกังวลเรื่อง Security และ Support

Front-end เฟรมเวิร์กที่เลือกใช้ได้แก่ React.js ซึ่งเป็น JavaScript library ที่ใช้สำหรับสร้าง User Interface ที่มีความซับซ้อนและแบ่งเป็นส่วนเล็ก ๆ ออกจากกันได้ และนำส่วนเล็ก ๆ ไปต่อยอด ใช้ซ้ำและแยกกันทำงานได้อย่างอิสระ

Database เลือกใช้ MongoDB ซึ่งเป็น NoSQL Database สามารถเก็บข้อมูลหลากหลาย และสามารถเข้าถึงตัวข้อมูลได้อย่างรวดเร็วเนื่องจากไม่มี Schema และรองรับ Multiple Cloud Provider

Software Architecture (RESTful microservices)

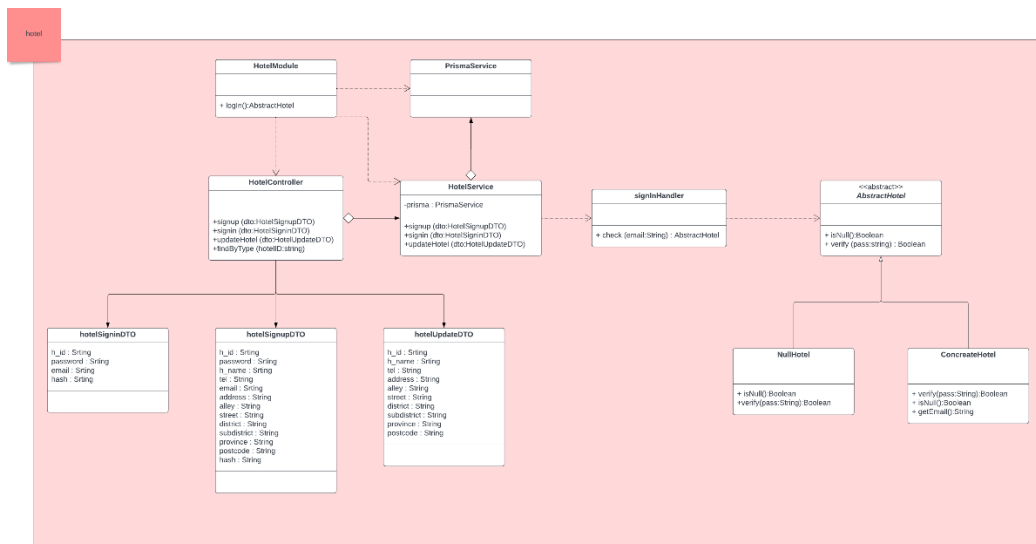


รูปที่ 2 Software Architecture

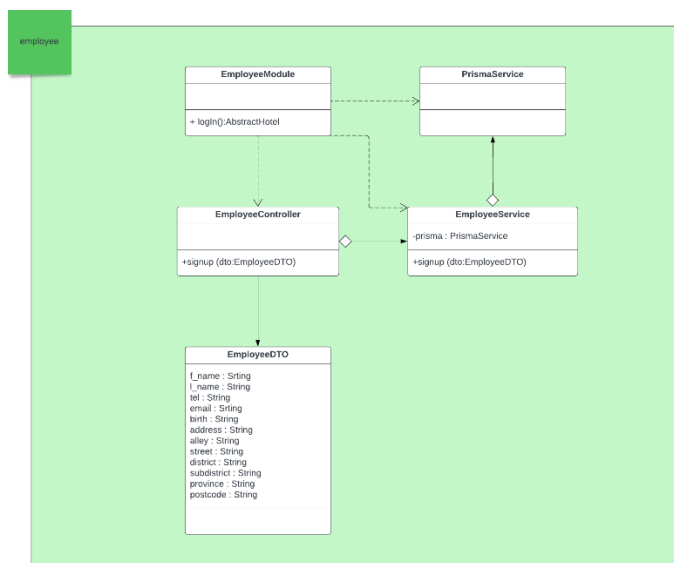
รูปแบบ architecture ที่เลือกใช้คือ Microservice ออกแบบโดยทำการแยก service จากขนาดใหญ่แตกย่อยลงมา ไม่มีการเก็บข้อมูลที่ฐานข้อมูลเดียวกัน เพราะแต่ละ service แยกออกมามีฐานข้อมูลเป็นของตัวเอง ติดต่อกันผ่าน API ซึ่งมีข้อดีคือ service มีขนาดเล็ก ทำให้แก้ไขจุดที่ผิดพลาดได้ง่าย และไม่กระทบกับ service อื่น ๆ

REST API เป็น architecture ที่เลือกใช้เพื่อแลกเปลี่ยนข้อมูลอยู่บนพื้นฐานของ Hypertext Transfer Protocol (HTTP) โดยส่งข้อมูลชนิด JSON ผ่าน URL

UML

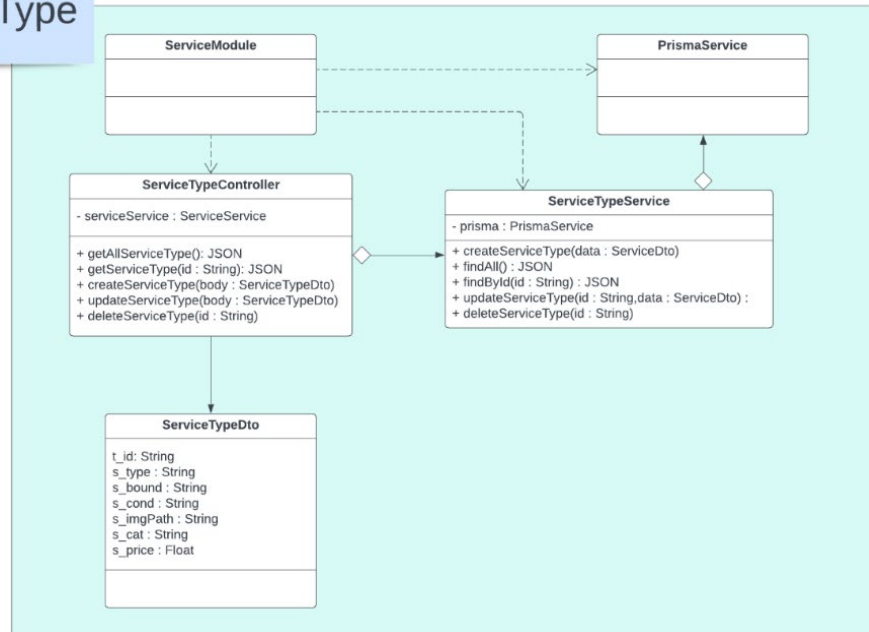


รูปที่ 3 Hotel UML



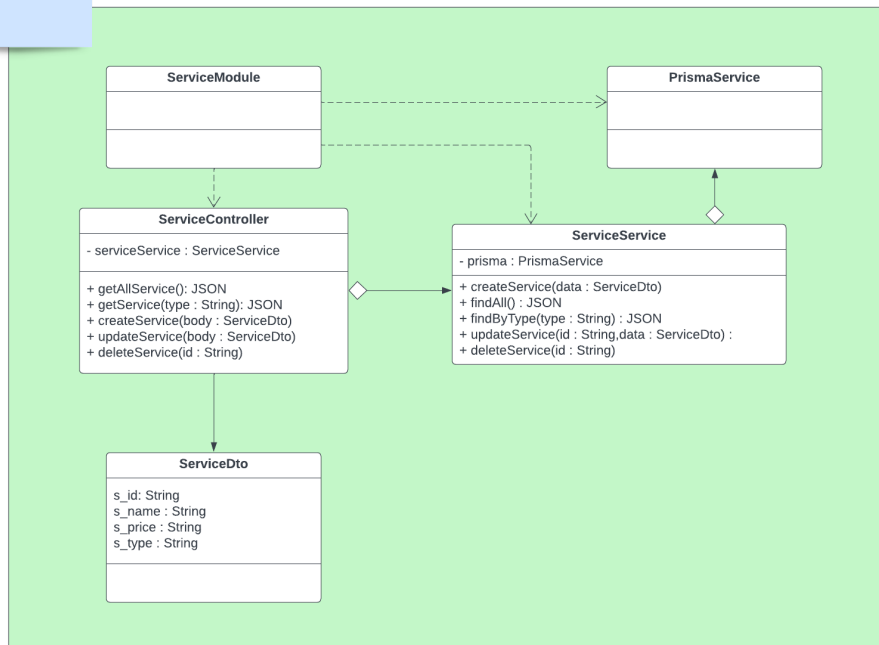
รูปที่ 4 Employee UML

ServiceType

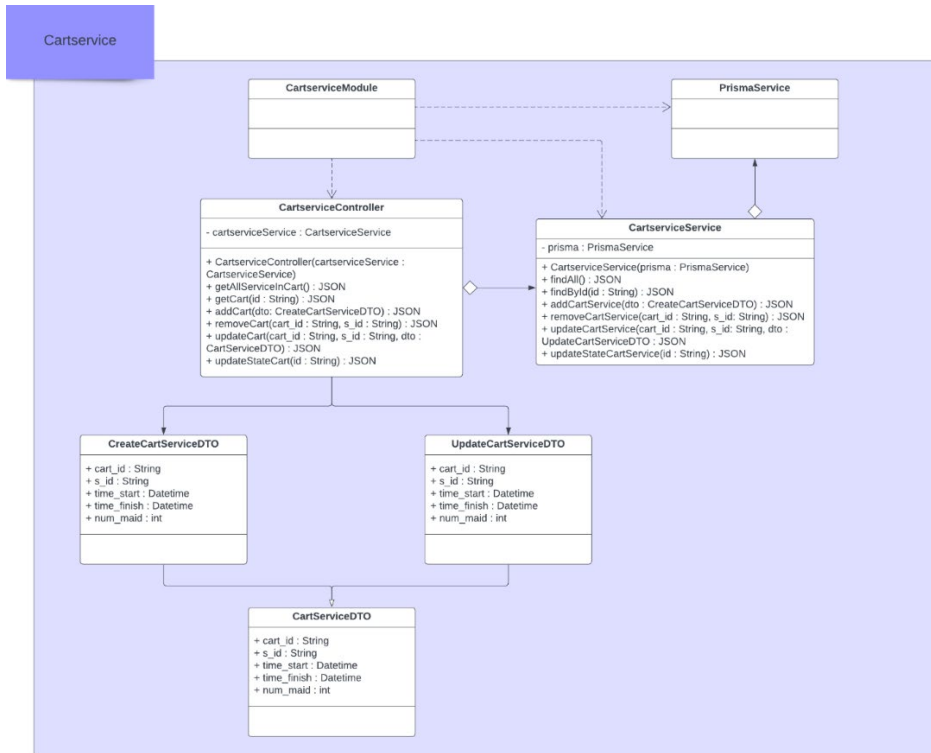


รูปที่ 5 Service Type UML

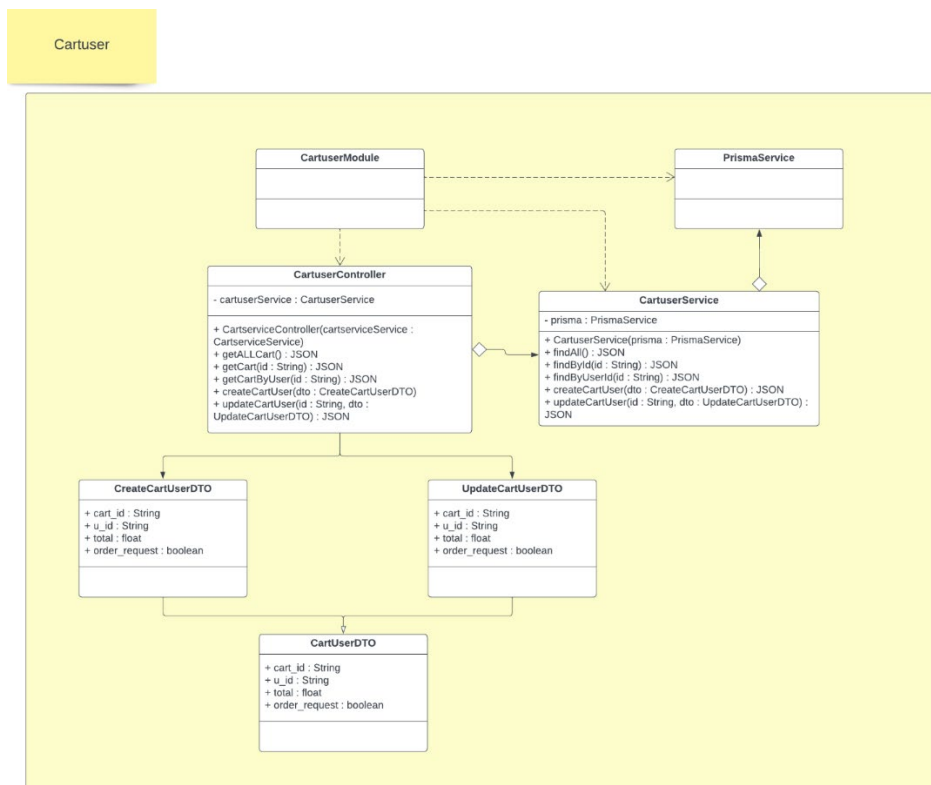
Service



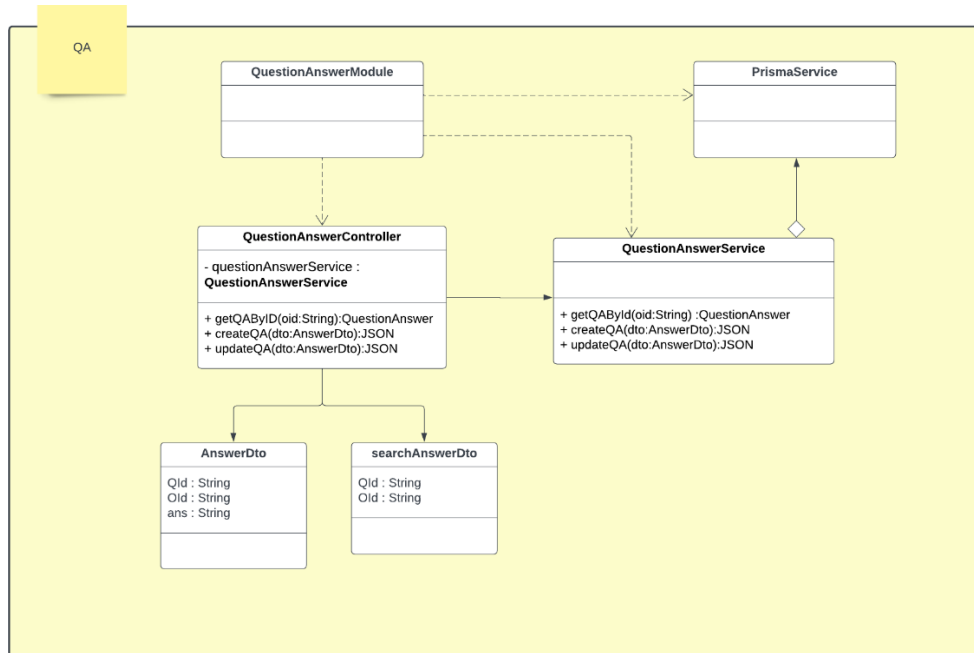
รูปที่ 6 Service UML



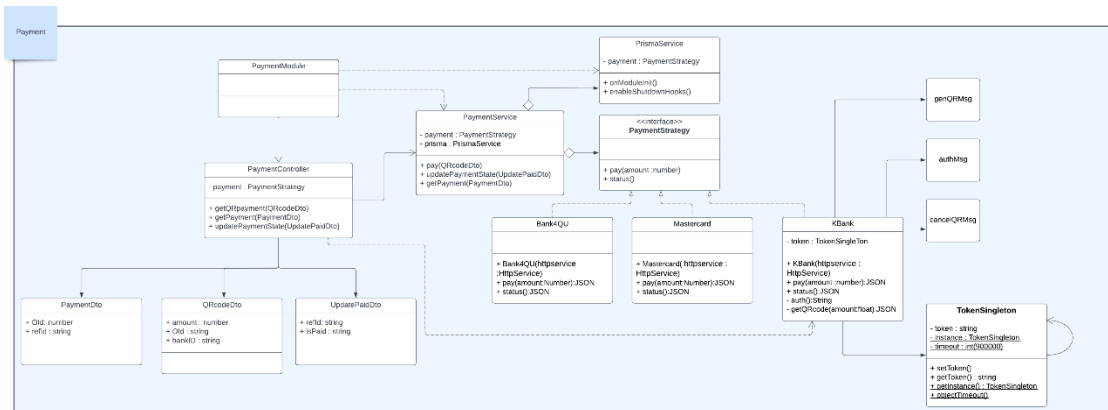
รูปที่ 7 Cart Service UML



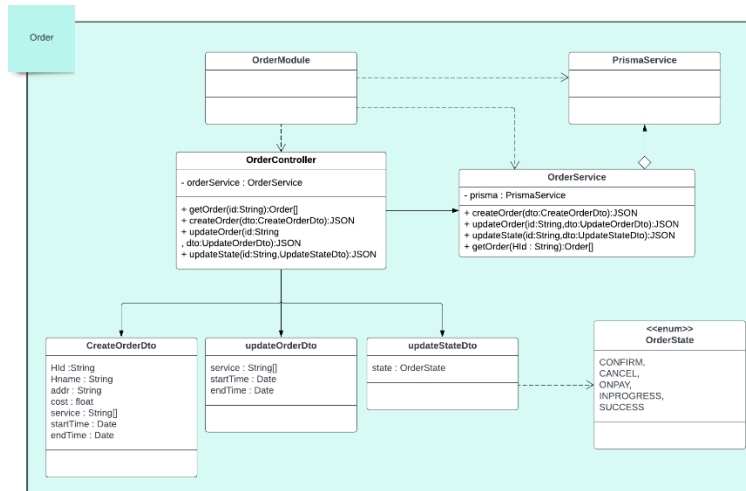
รูปที่ 8 User Cart UML



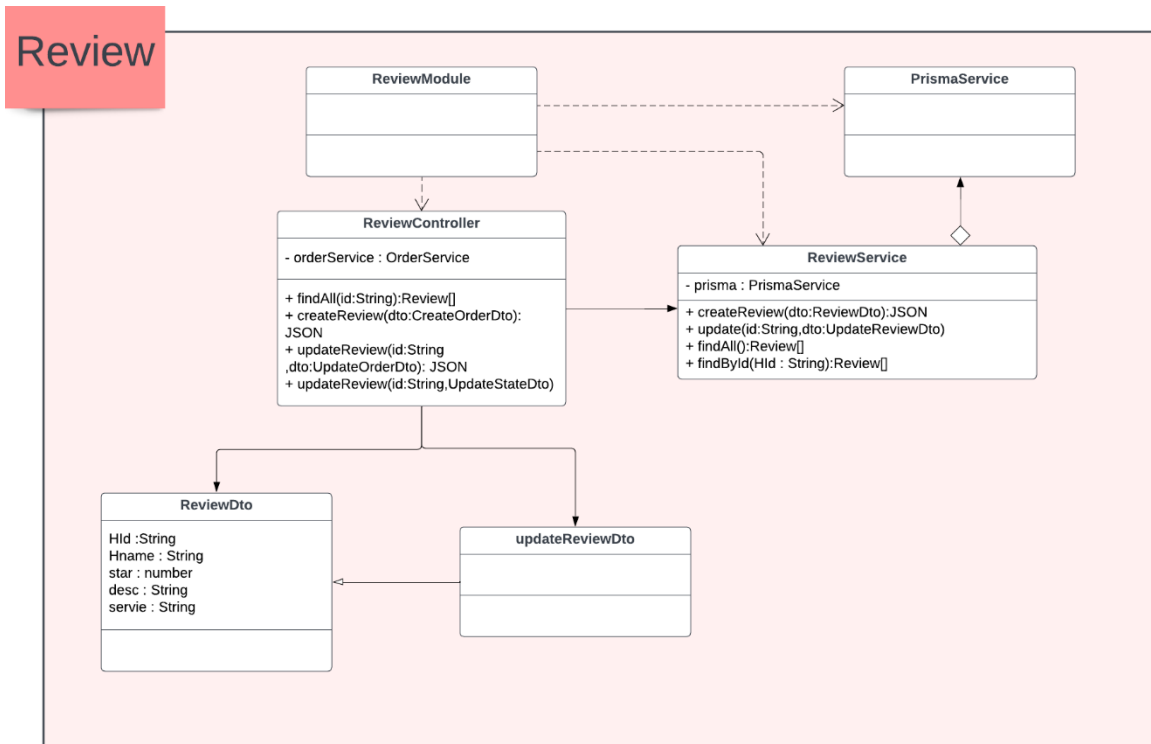
รูปที่ 9 Q&A UML



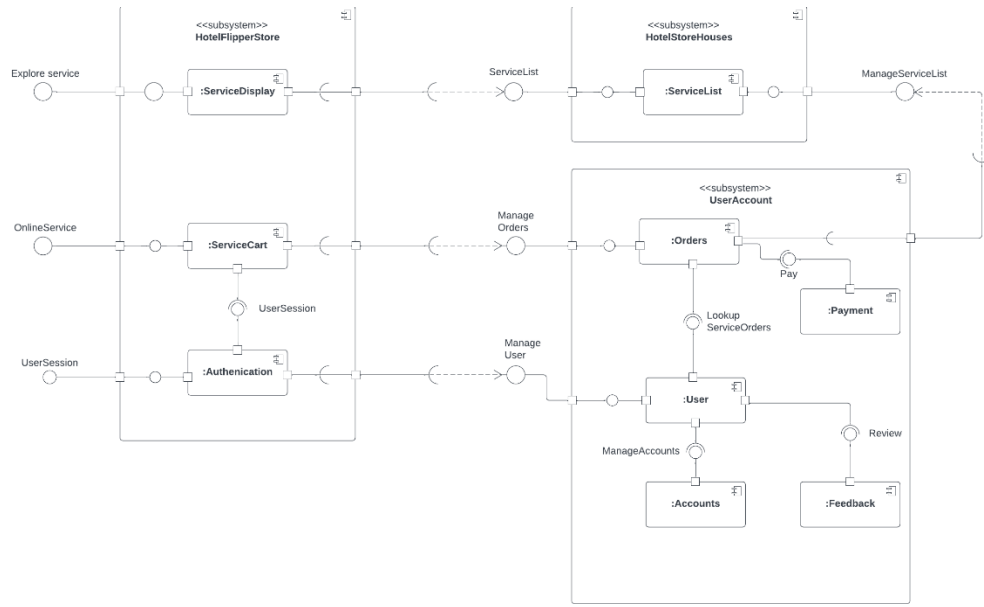
รูปที่ 10 Payment UML



รูปที่ 11 Order UML



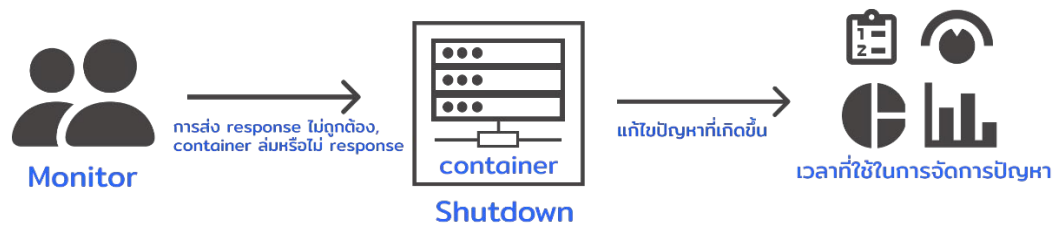
รูปที่ 12 Feedback UML



รูปที่ 12 Front-End Component Diagram

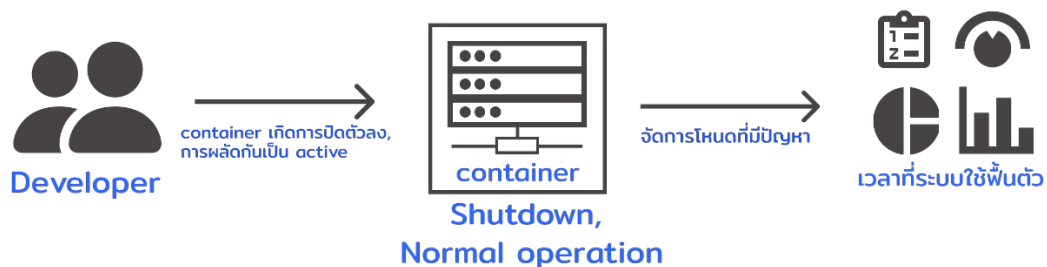
Scenarios

1. Availability



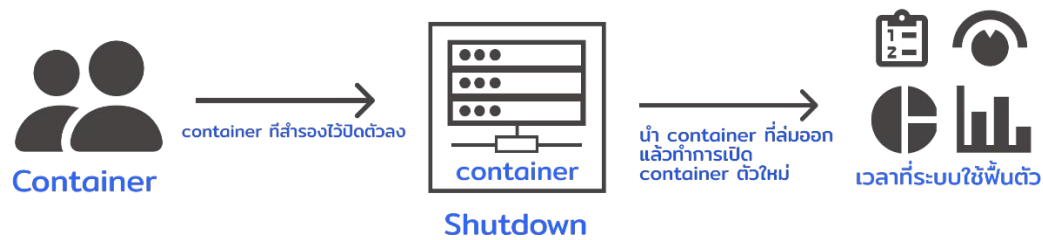
Source of stimulus	Monitor
Stimulus	การส่ง response ไม่ถูกต้อง , container ล่มหรือไม่ response
Artifacts	container
Environment	shutdown
Response	แก้ไขปัญหาที่เกิดขึ้น
Response measure	เวลาที่ใช้ในการจัดการปัญหา

2. Availability (redundant spare(warm spare))



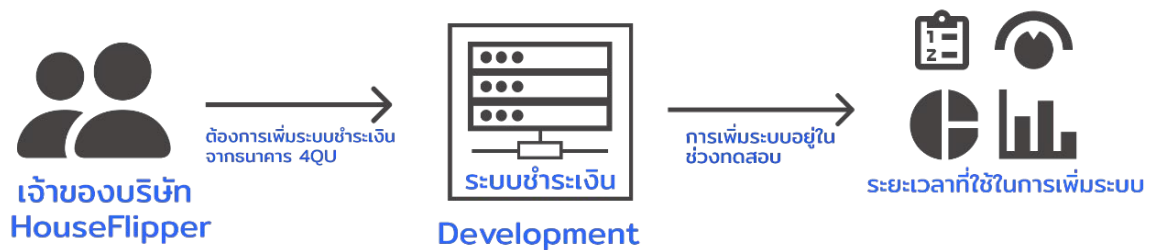
Source of stimulus	developer
Stimulus	container เกิดการปิดตัวลง , การปลัดกันเป็น active
Artifacts	container
Environment	shutdown , Normal operation
Response	จัดการโหนดที่มีปัญหา
Response measure	เวลาที่ระบบใช้ฟื้นตัว

3. Availability (Removal from service)



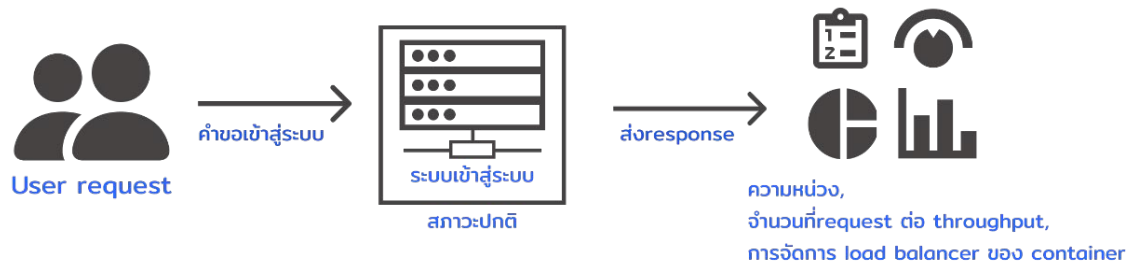
Source of stimulus	container
Stimulus	container ที่สำรองไว้ปิดตัวลง
Artifacts	container
Environment	shutdown
Response	นำ container ที่ล่ม ออก แล้วทำการเปิด container ตัวใหม่
Response measure	เวลาที่ระบบใช้ฟื้นตัว

4. Integrability



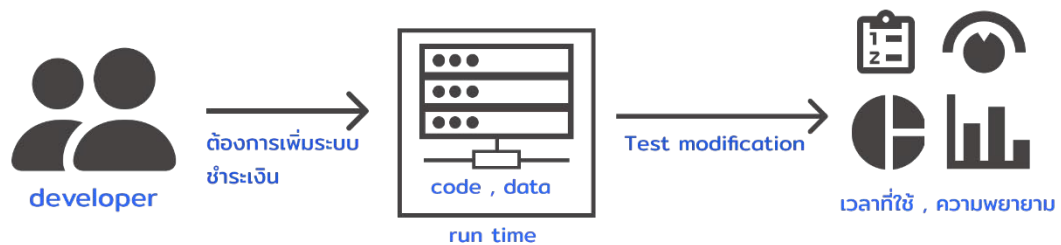
Source of stimulus	เจ้าของบริษัท HouseFlipper
Stimulus	ต้องการเพิ่มระบบชำระเงินจากธนาคาร 4QU
Artifacts	ระบบชำระเงิน
Environment	Development
Response	การเพิ่มระบบอยู่ในช่วงทดสอบ
Response measure	ระยะเวลาที่ใช้ในการเพิ่มระบบ

5. Performance



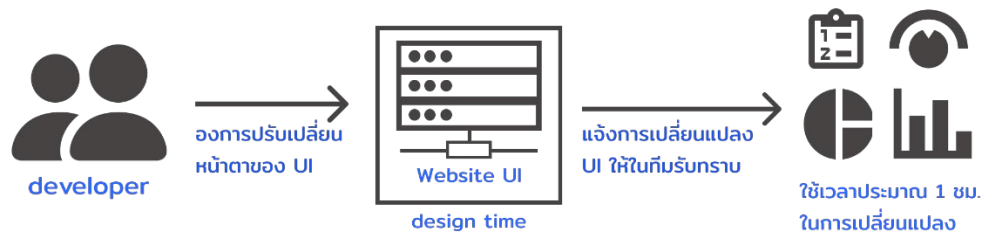
Source of stimulus	User request
Stimulus	คำขอการเข้าสู่ระบบ
Artifacts	ระบบเข้าสู่ระบบ
Environment	ในสถานะปกติ
Response	ส่ง response
Response measure	ความหน่วง (latency) , จำนวนที่ request ต่อ throughput,การจัดการ load balancer ของ container

6. Modification



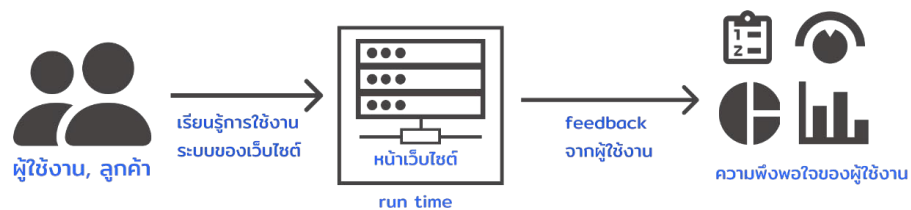
Source of stimulus	developer
Stimulus	ต้องการเพิ่มระบบชำระเงิน
Artifacts	code , data
Environment	run time
Response	Test modification
Response measure	เวลาที่ใช้ , ความพยายาม

7. Modification



Source of stimulus	Developer
Stimulus	ต้องการปรับเปลี่ยนหน้าตาของ UI
Artifacts	UI ของเว็บไซต์
Environment	ขณะ design time
Response	แจ้งการเปลี่ยนแปลง UI ให้ในทีมรับทราบ
Response measure	ใช้เวลาประมาณ 1 ชม. ในการเปลี่ยนแปลง

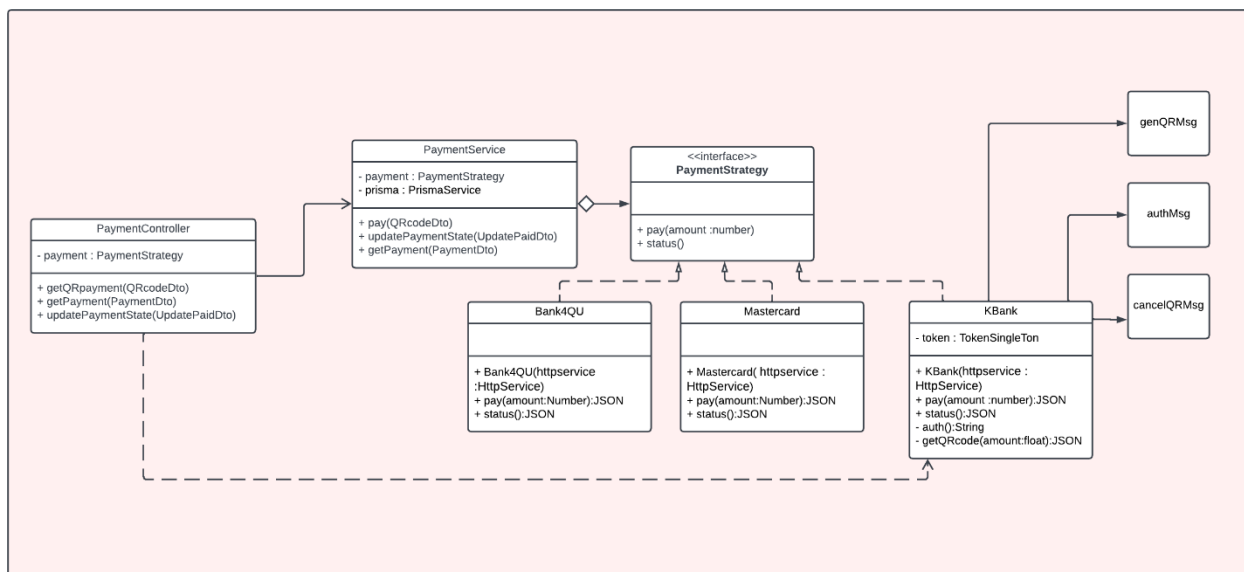
8. Usability



Source of stimulus	ผู้ใช้งาน / ลูกค้า
Stimulus	เรียนรู้การใช้งานระบบของเว็บไซต์
Artifacts	หน้าเว็บไซต์
Environment	runtime
Response	feedback จากผู้ใช้งาน
Response measure	ความพึงพอใจของผู้ใช้งาน

Design Pattern

1. Strategy



ปัญหา : เพื่อแก้ปัญหาการเลือกช่องทางการชำระเงินในอนาคตซึ่งจะเกิดในช่วง runtime

วิธีการใช้ : เพิ่ม interface ชื่อ PaymentStrategy ซึ่งถูก implement โดย คลาส Bank4QU, Mastercard และ KBank หลังจากนั้น ให้คลาส PaymentService สร้าง instance ของ PaymentStrategy เพื่อให้ controller ซึ่งเป็นส่วนที่ติดต่อกับ client ดำเนินการต่อ

ผลจากการใช้งาน : การเพิ่มช่องทางการชำระเงินเพิ่มได้โดยง่าย , การแก้ไขปรับปรุงโค้ดทำได้ง่ายขึ้น

ตัวอย่างโค้ด

```
1 export class KBank implements PaymentStrategy{
2   private readonly logger = new Logger(KBank.name);
3   private bankID = "KBank"
4   private token
5   constructor(private readonly httpService:HttpService){
6     //this.httpService = new HttpService()
7     // this.token = TokenSingleton.getInstance()
8   }
9   public async pay(amount: number) {
10    this.token = TokenSingleton.getInstance()
11    if (this.token.getToken() == null){
12      this.token.setToken(await this.auth())
13      TokenSingleton.objectTimeout()
14    }
15    return await this.getQRcode(amount)
16  }
```

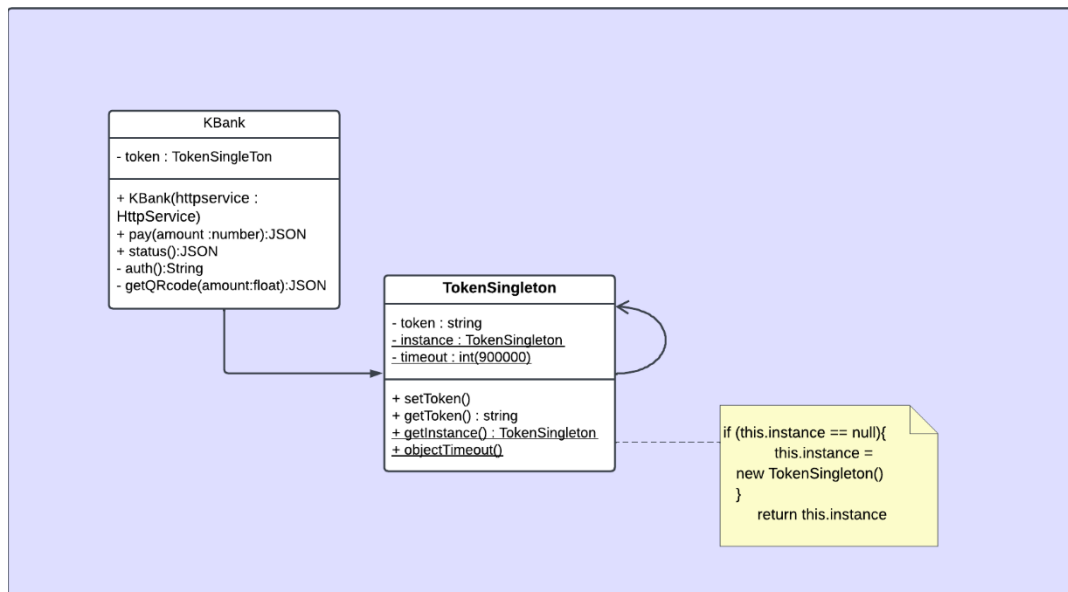


```
1 export interface PaymentStrategy {  
2     pay(amount: number)  
3     status()  
4 }
```



```
1 export class Bank4QU implements PaymentStrategy{  
2     private bankID = "4QU"  
3     private readonly logger = new Logger(Bank4QU.name);  
4     constructor(private httpService: HttpService){  
5  
6     }  
7     public async pay(amount: number) {  
8         let date = await new Date().toLocaleString('sv-SE').split(' ')  
9         const url = await this.getqrURL(amount)  
10        return url  
11    }
```


2. Singleton




ปัญหา : การใช้งาน Token ของธนาคารกสิกรมีอายุ 30 นาที (แต่ทางเราเลือกกำหนดให้มีอายุ 15 นาที)มีการกำหนดครั้งการขอ (request) ไว้เพื่อให้เกิดประโยชน์สูงสุด และเพื่อให้ทุกครั้งที่ถูกค้าหรือผู้ใช้งานที่ต้องการจะชำระเงินเข้าถึงToken เดียวกัน

วิธีการใช้ : สร้าง class `TokenSingleton` ซึ่งมี attribute ชนิด static และกำหนดการเข้าถึงให้เป็น private เพิ่ม static method `getInstance()` และ `objectTimeout()` เพื่อควบคุมพฤติกรรมของ token

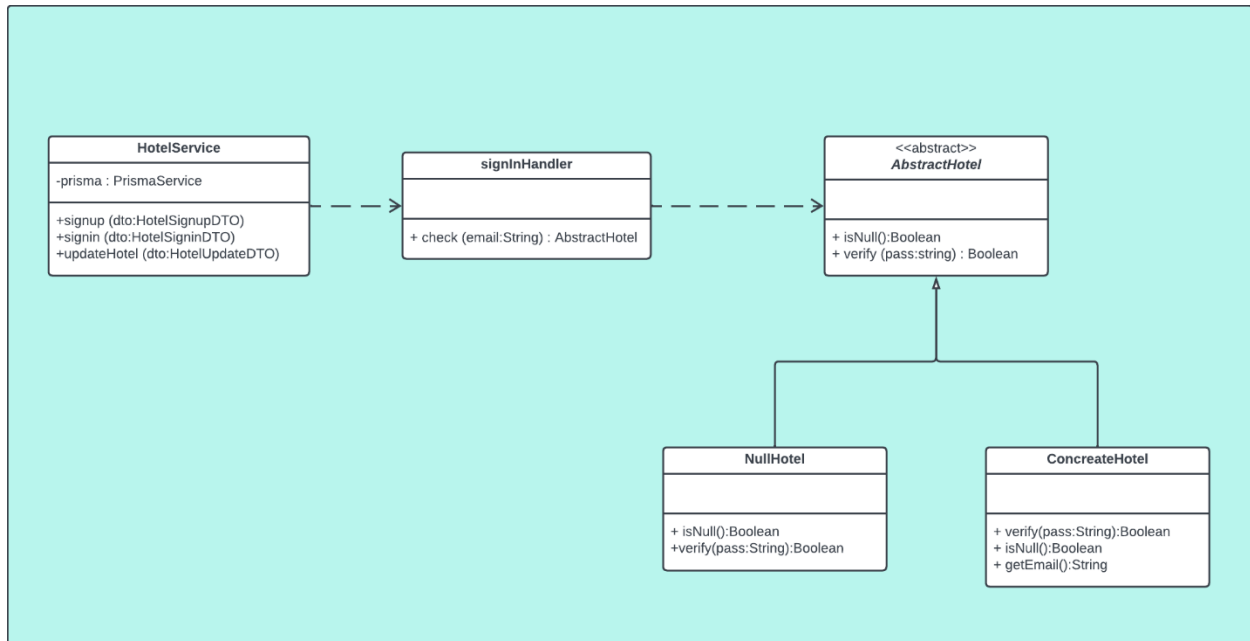
ผลจากการใช้งาน : มีแค่ Token เดียวในช่วงเวลา , ลูกค้าเข้าถึง Token เดียวกัน

```
1 public async pay(amount: number) {
2     this.token = TokenSingleton.getInstance()
3     if (this.token.getToken() == null){
4         this.token.setToken(await this.auth())
5         TokenSingleton.objectTimeout()
6     }
7     return await this.getQRcode(amount)
8 }
```



```
1  export class TokenSingleton {
2      private static instance = null
3      private token
4      private static timeout = 900000
5      constructor(){
6      }
7
8      public static getInstance() : TokenSingleton{
9          if (this.instance == null){
10             this.instance = new TokenSingleton()
11          }
12          return this.instance
13      }
14
15      public setToken(token:string){
16          this.token = token
17      }
18      public getToken():string{
19          return this.token
20      }
21
22      public static objectTimeout(){
23          if (this.instance != null) {
24              setTimeout(() => {
25                  this.instance = null
26              }, this.timeout);
27          }
28      }
29
30 }
```

3. Null Object



ปัญหา : การเข้าสู่ระบบเมื่อ email ไม่ถูกต้อง การที่ต้องส่งข้อความกลับไปยังหาผู้ใช้งานหรือลูกค้าทำได้ยาก เพราะต้องมาตั้งเงื่อนไขในส่วนของ service และ เพื่อต้องการเช็คก่อนที่จะทำ method อื่นๆ ว่า email นั้นมีจริงหรือไม่ กล่าวคือ เคยสมัครบัญชีกับทางระบบของเราแล้วหรือไม่

วิธีการใช้ : เพิ่ม abstract class ชื่อ `AbstractHotel` ซึ่งมีเมธอดนามธรรมเพื่อให้ระบุว่า object นี้เป็น null หรือไม่ และตรวจสอบว่ารหัสผ่านของอีเมลนั้นถูกต้องหรือไม่ คลาสนี้ถูกสืบทอดโดย `NullHotel` และ `ConcreateHotel` หลังจากนั้น สร้างคลาส `signInHandler` ที่มีเมธอด `check()` เพื่อทำการเช็คว่ามี email นั้นมีหรือไม่ หากมีก็ให้สร้าง instance ของ `ConcreateHotel` ออกไป หากไม่ให้สร้าง instance ของ `NullHotel`