



# Hotel Flipper

Group 8





# Hotel Flipper คืออะไร ?

Web Application ที่ทำให้โรงแรมสามารถจัดหาพนักงานที่มีความชำนาญในแต่ละด้าน  
มาช่วยในงานต่างๆ





## ทำไมต้อง Hotel Flipper ?

เพราะ Hotel Flipper ของเราเป็นผู้ที่จะมาช่วยจัดการกับปัญหา ให้กับโรงแรมทั้งขนาดเล็ก และขนาดใหญ่ ได้อย่างมีประสิทธิภาพ ด้วยทีมงานฝีมือดีในแต่ละด้าน ที่ได้ผ่านการคัดสรรมาอย่างพิถีพิถัน ทำให้การันตีได้เลยว่าผลลัพธ์ได้ออกมาแน่น ต้องเป็นที่น่าพึงพอใจสำหรับผู้ใช้งานได้อย่างแน่นอน





## โรงแรมผู้รับบริการ



## ข้อดี Hotel Flipper



## ผู้ให้บริการ

### ให้บริการครบวงจร

มีบริการให้เลือกหลากหลาย ตอบโจทย์ทุกความต้องการของโรงแรม

### ราคาประหยัดกว่า

ลดรายจ่ายการจ้างแม่บ้านประจำ ด้วยการจ้างผู้บริการจาก Hotel Flipper

### รายได้ดี มีงานทำ

การันตีรายได้ต่อชั่วโมง

### ฝึกอบรมโดยผู้เชี่ยวชาญ

ไม่มีประสบการณ์ไม่ใช่ปัญหาในการเป็นส่วนหนึ่งกับพวคเรา

# Service

รวมรวมบริการทำความสะอาดครบวงจรไว้ที่นี่  
มีการเตรียมและฝึกฝนพนักงาน ทั้งด้านกาย  
ด้านใจ และ ความคิด ให้มีมาตรฐานที่เท่ากัน  
และเสมอต้นเสมอปลาย ใช้เทคโนโลยี  
เครื่องมือที่ทันสมัย เพื่อแก้ไขทุกปัญหาให้คุณ



## COVID-19

ระบบการพ่นน้ำยาแบบ  
ละอองฟอย เดินทางไปใน  
อากาศ เกาะบนพื้นผิว และ  
ข้าเชื้อโรค แบคทีเรีย

## กำจัดไรฝุ่น

ใช้เครื่องดูดฝุ่นที่มี  
ประสิทธิภาพสูง เพื่อการัน  
ตีได้ว่าจะปราศจากฝุ่น  
อย่างแท้จริง

## ความปลอดภัย

ดูแลรักษาความปลอดภัย  
ให้กับบริษัทห้างร้านต่างๆ  
หรือกระทั่งที่อยู่อาศัย ทั้ง  
เวลากลางวันและกลางคืน

## กั่วไป

การทำความสะอาดกั่วไป  
ภายใน เช่น บัด กวาด เช็ด  
ถู ถูดฝุ่น ล้างห้องน้ำ นำ  
ขยะไปทิ้ง เป็นต้น

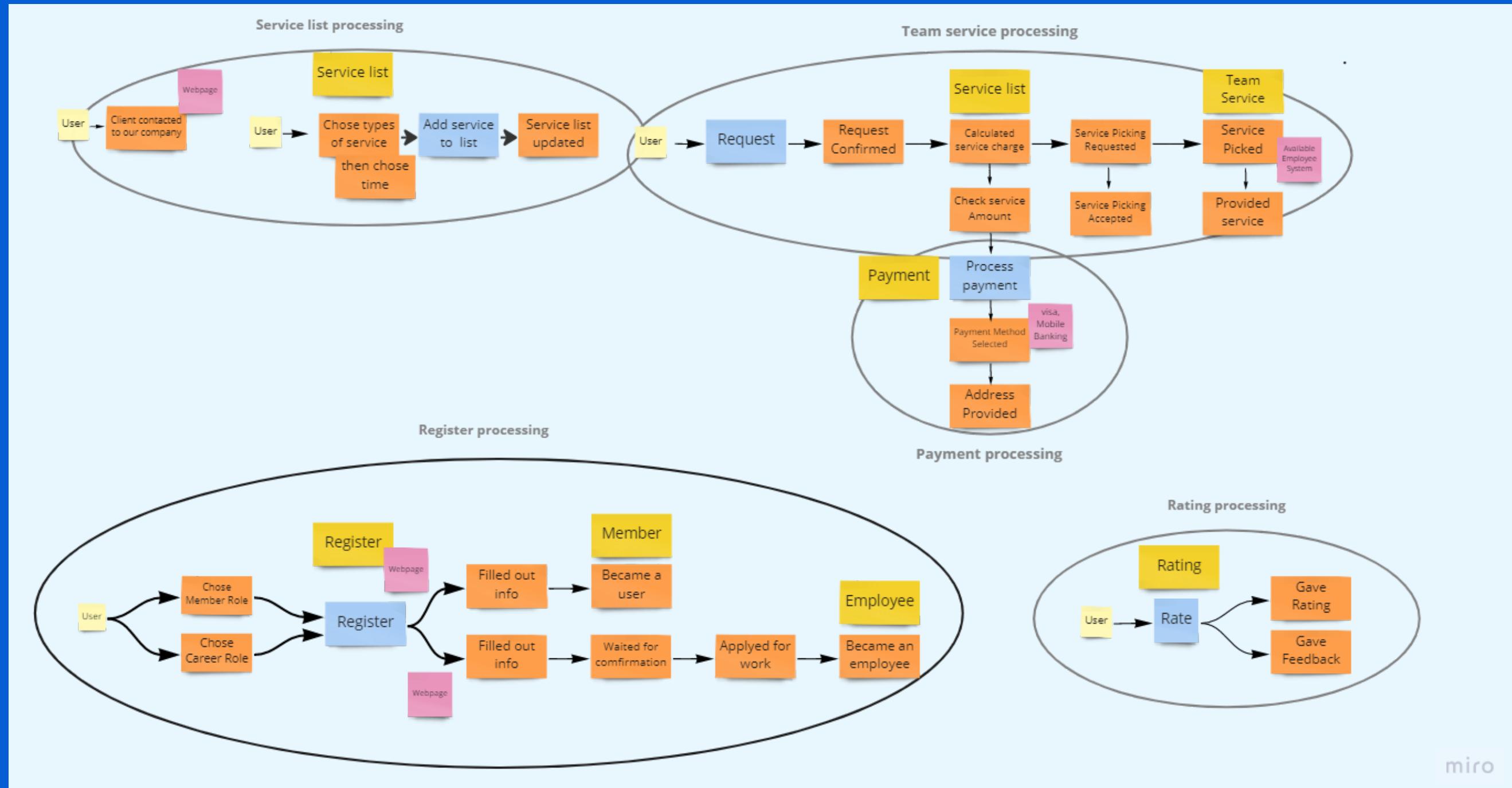
## ซ่อมบำรุง

การซ่อมบำรุงรักษา กั่วไป  
เช่น ซ่อมเครื่องปรับอากาศ  
รวมไปถึงการจัดและ  
ตกแต่งสวน

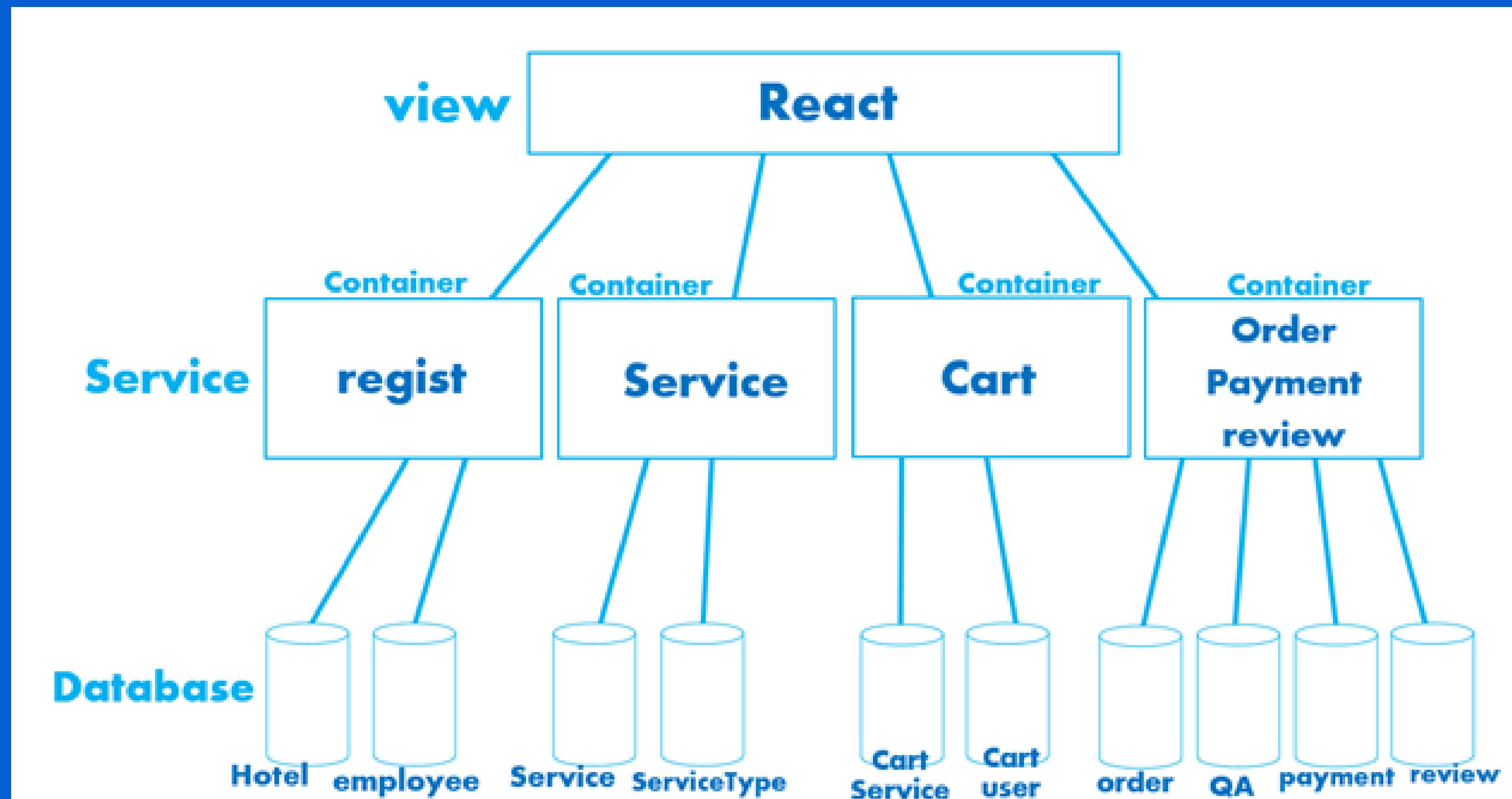




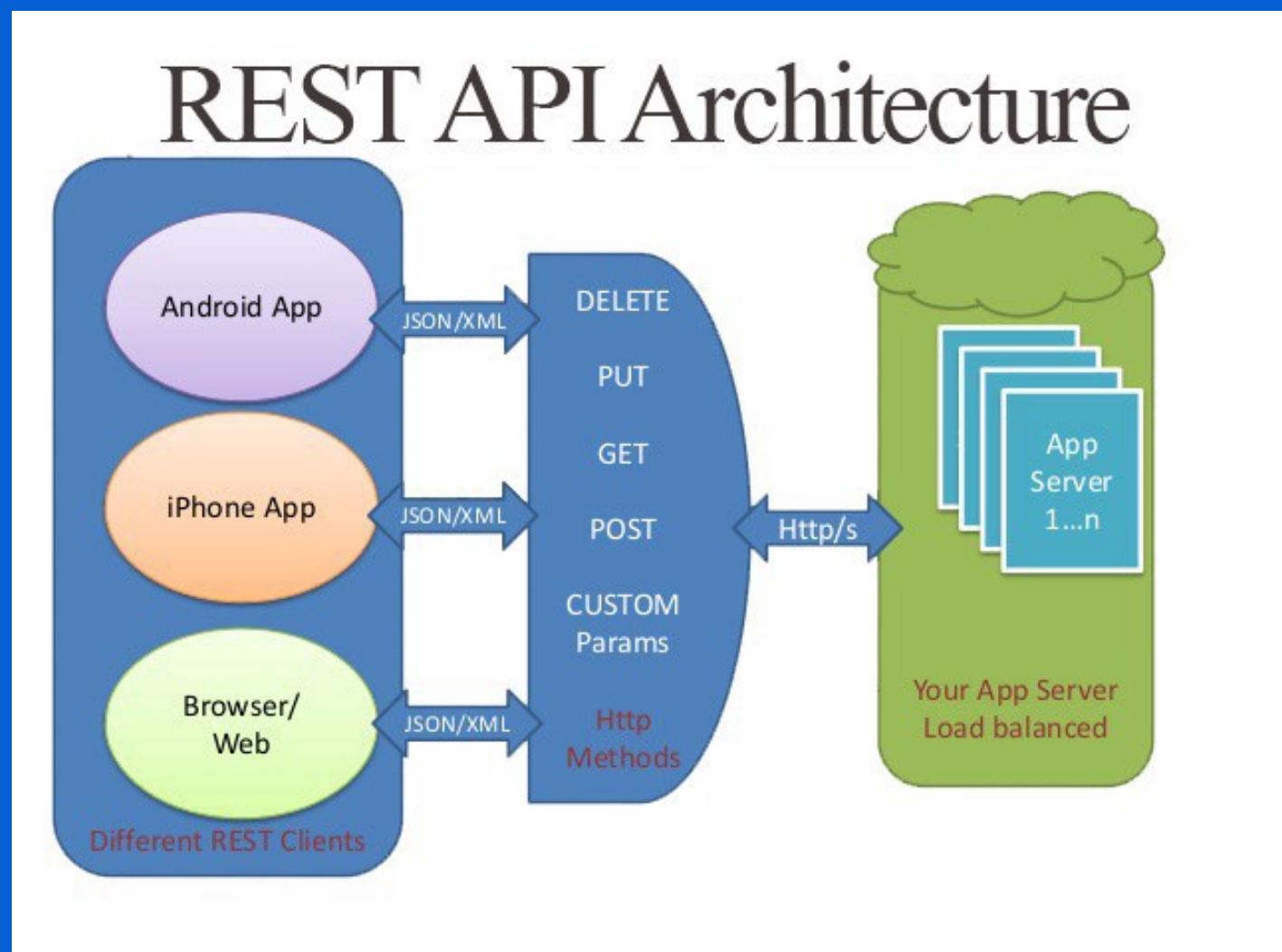
# Bounded Context



# Software Architecture



# REST API





# Unified Modeling Language (UML)

PAYMENT



CART  
SERVICE



SERVICE  
TYPE

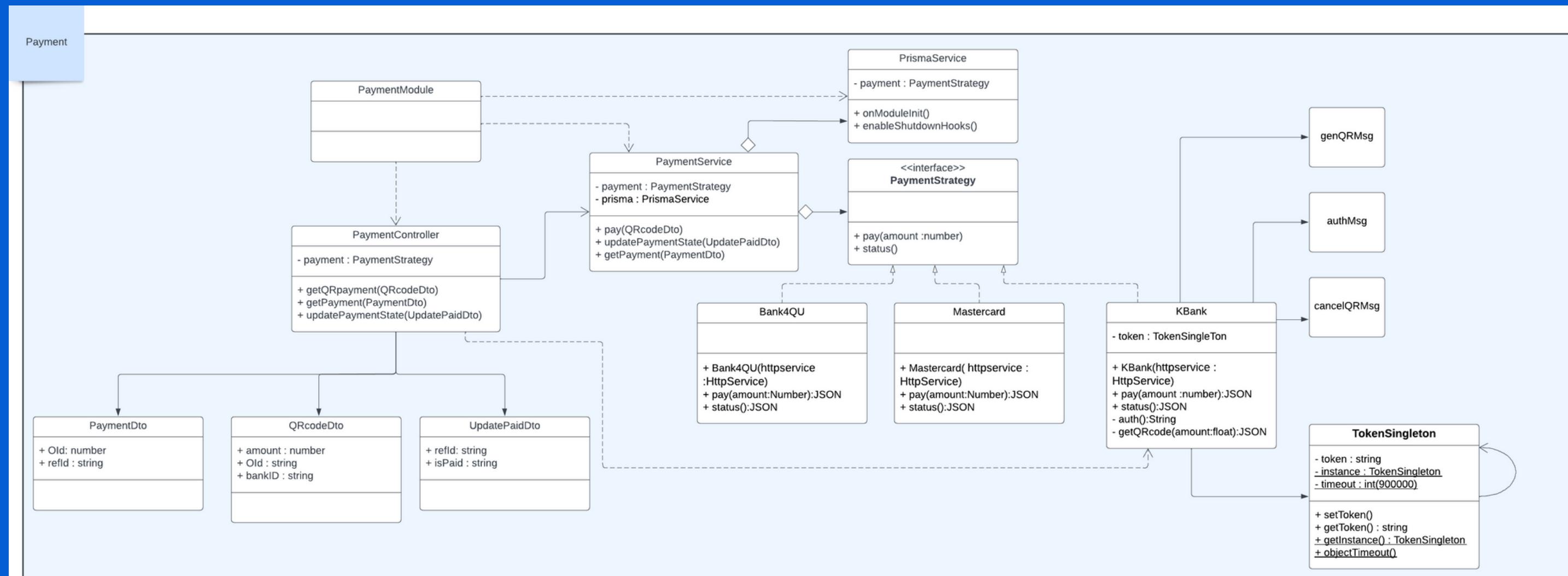


COMPONENT  
DIAGRAM



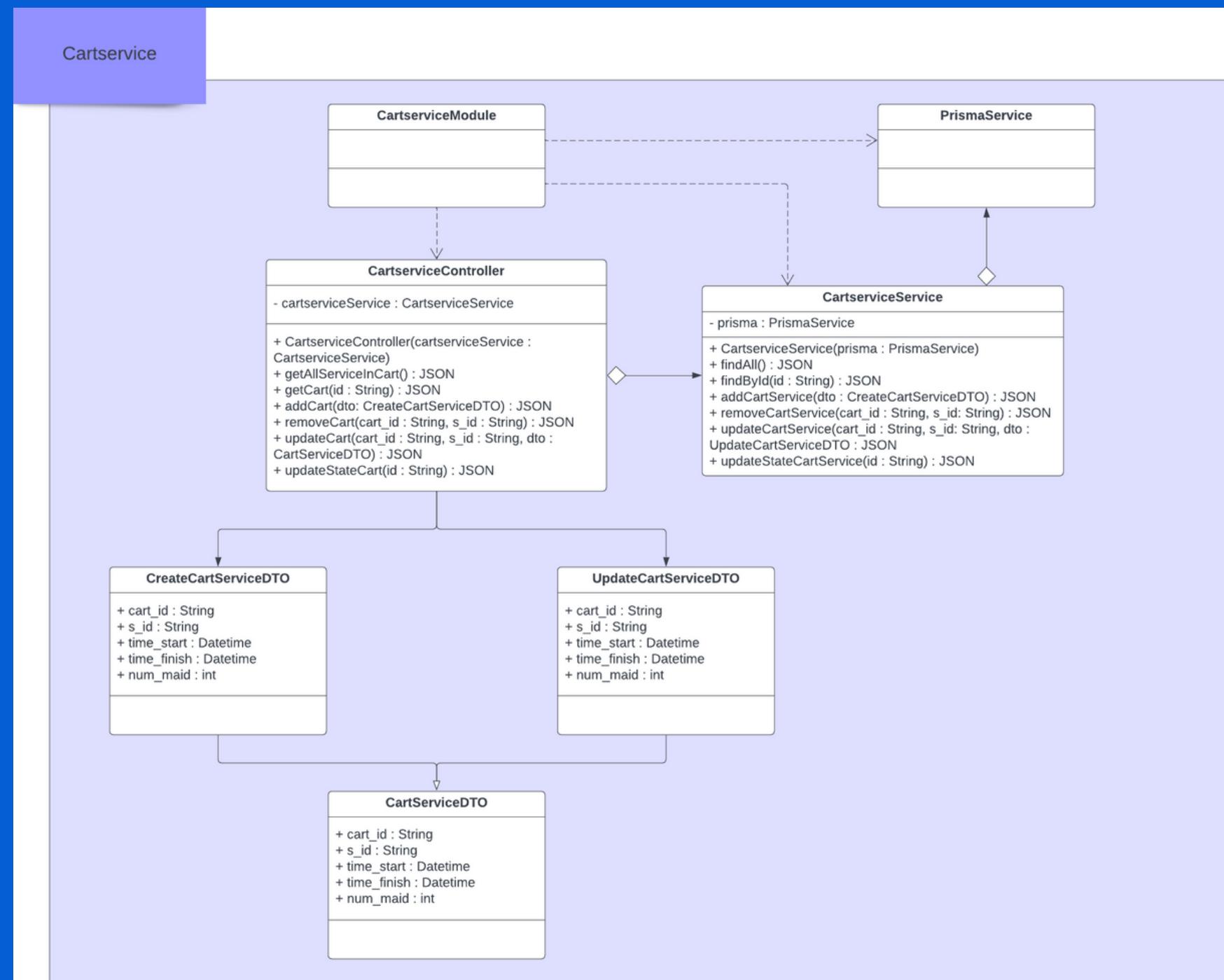


# Payment UML

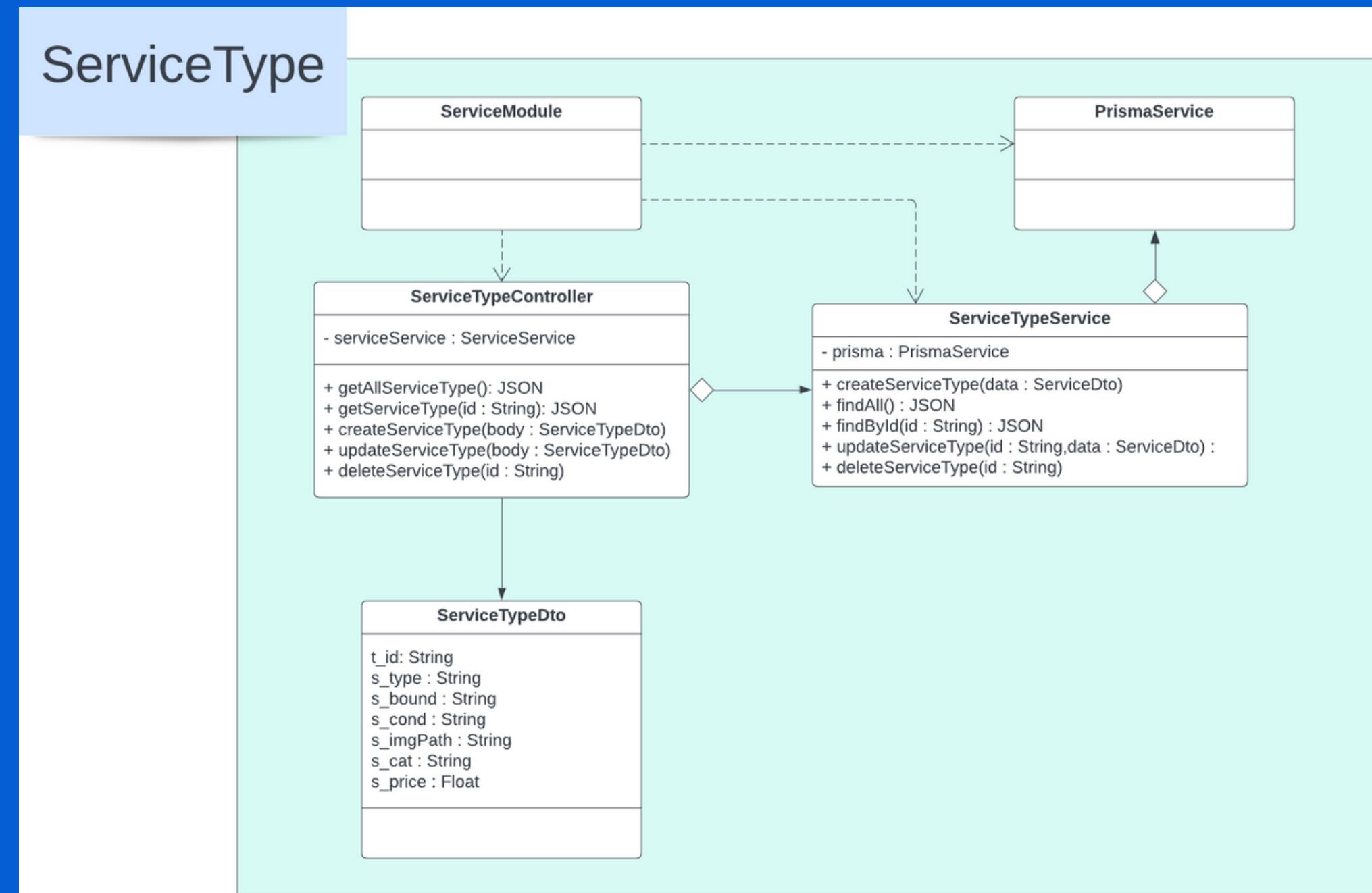




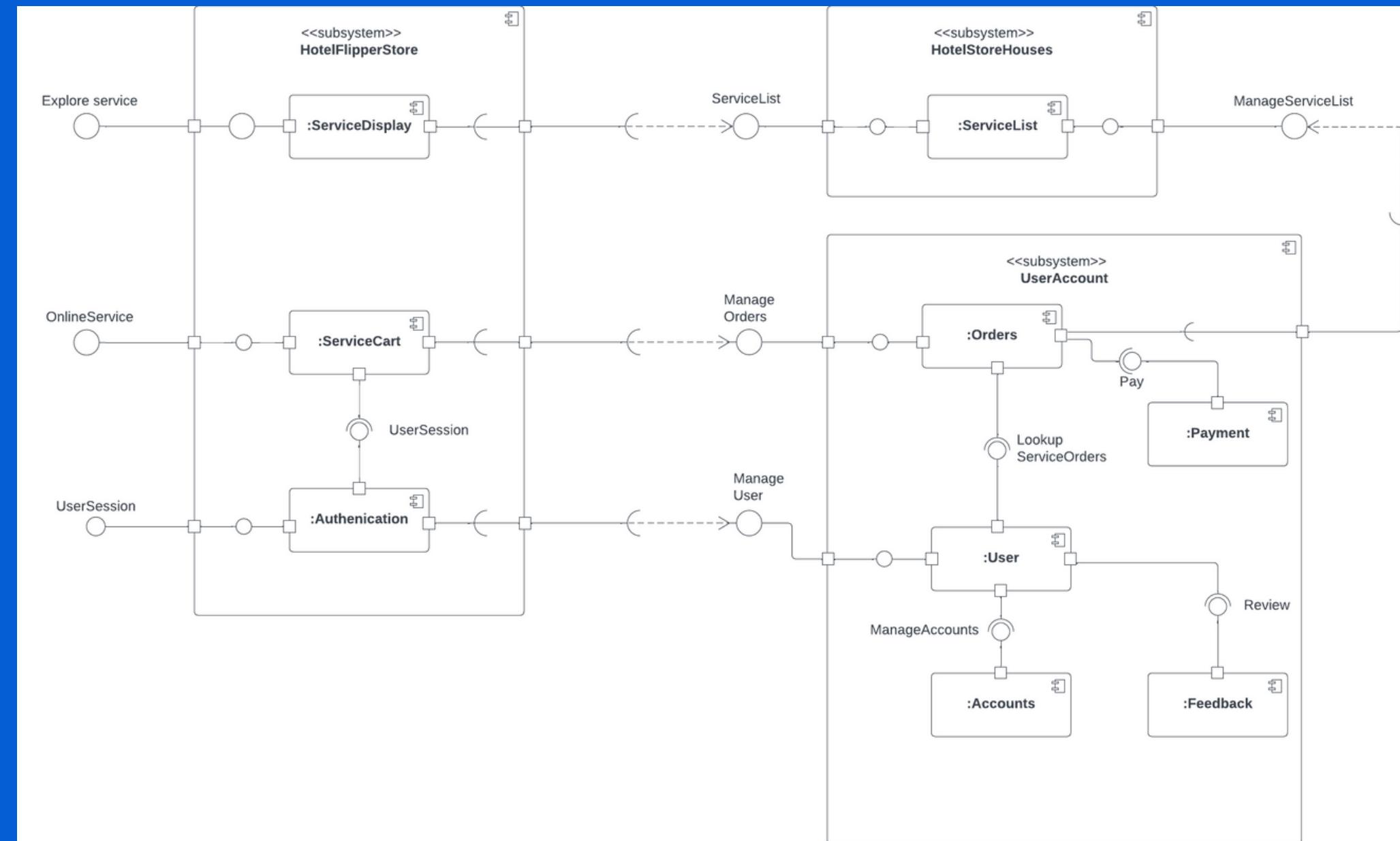
# Cart Service UML



# Service Type UML



# Component Diagram



# Design Patterns

## 01

### STRATEGY

ปัญหา :

เพื่อแก้ปัญหาการเลือกช่อง  
การการชำระเงินในอนาคตซึ่งจะ<sup>จะ</sup>  
เกิดในช่วง runtime

ผลจากการใช้งาน :

- เพิ่มช่องทางการชำระเงินได้  
โดยง่าย
- การแก้ไขปรับปรุงโค้ดทำได้  
ง่ายขึ้น

Behavioral Patterns

## 02

### SINGLETON

ปัญหา :

มีการกำหนดครั้งการขอ  
(request ) และให้ทุกครั้งที่ผู้ใช้  
งานต้องการจะชำระเงินเข้าถึง  
Token เดียวกันได้

ผลจากการใช้งาน :

- 1 Token ต่อช่วงเวลา
- ลูกค้าเข้าถึง Token  
เดียวกัน

Creational Patterns

## 03

### NULL OBJECT

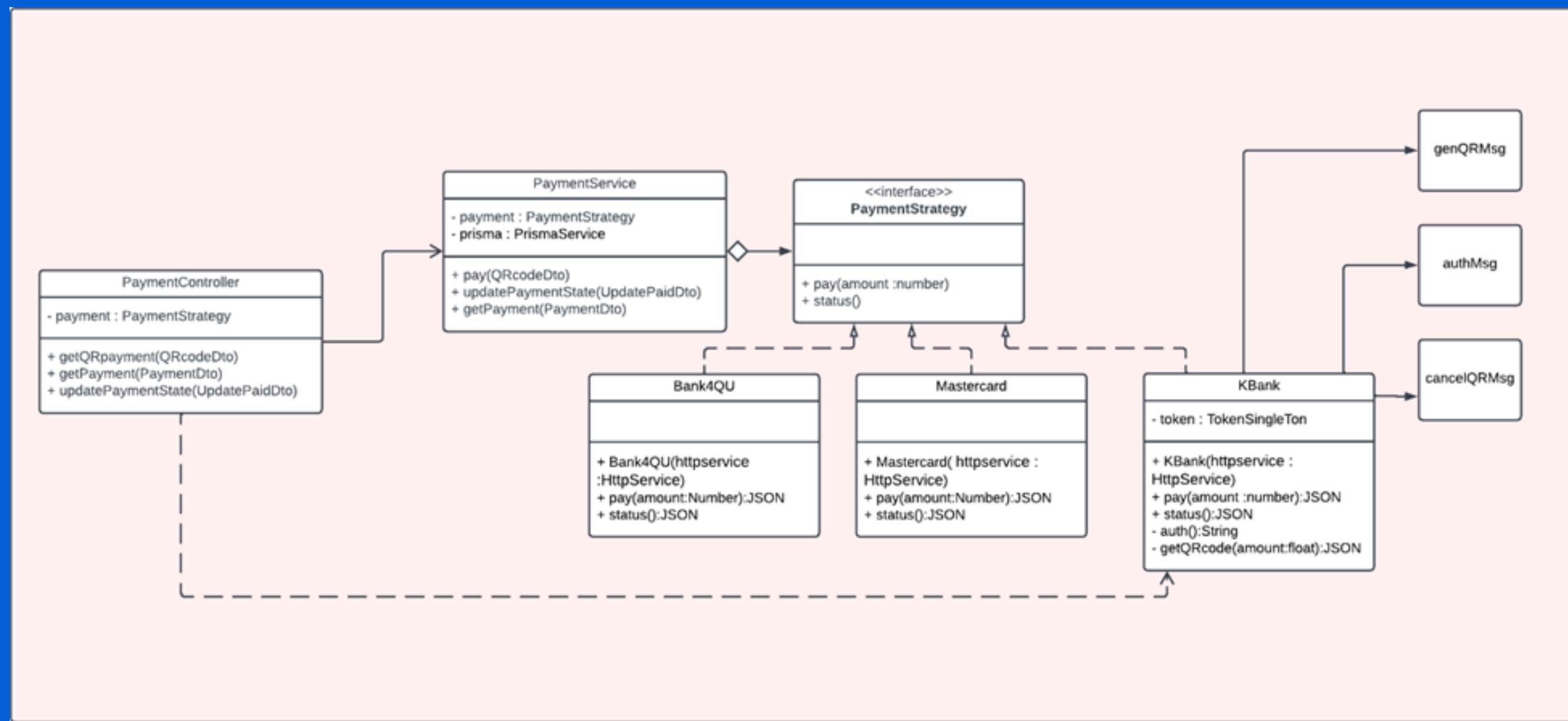
ปัญหา :

การเช็คว่าผู้ใช้งานเคยสมัครบัญชี  
กับทางระบบไว้หรือไม่นั้นคุณได้  
ยาก

Behavioral Patterns



# 01 STRATEGY



# ຕົວຢ່າງໂຄດ

```
 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     }  
}
```

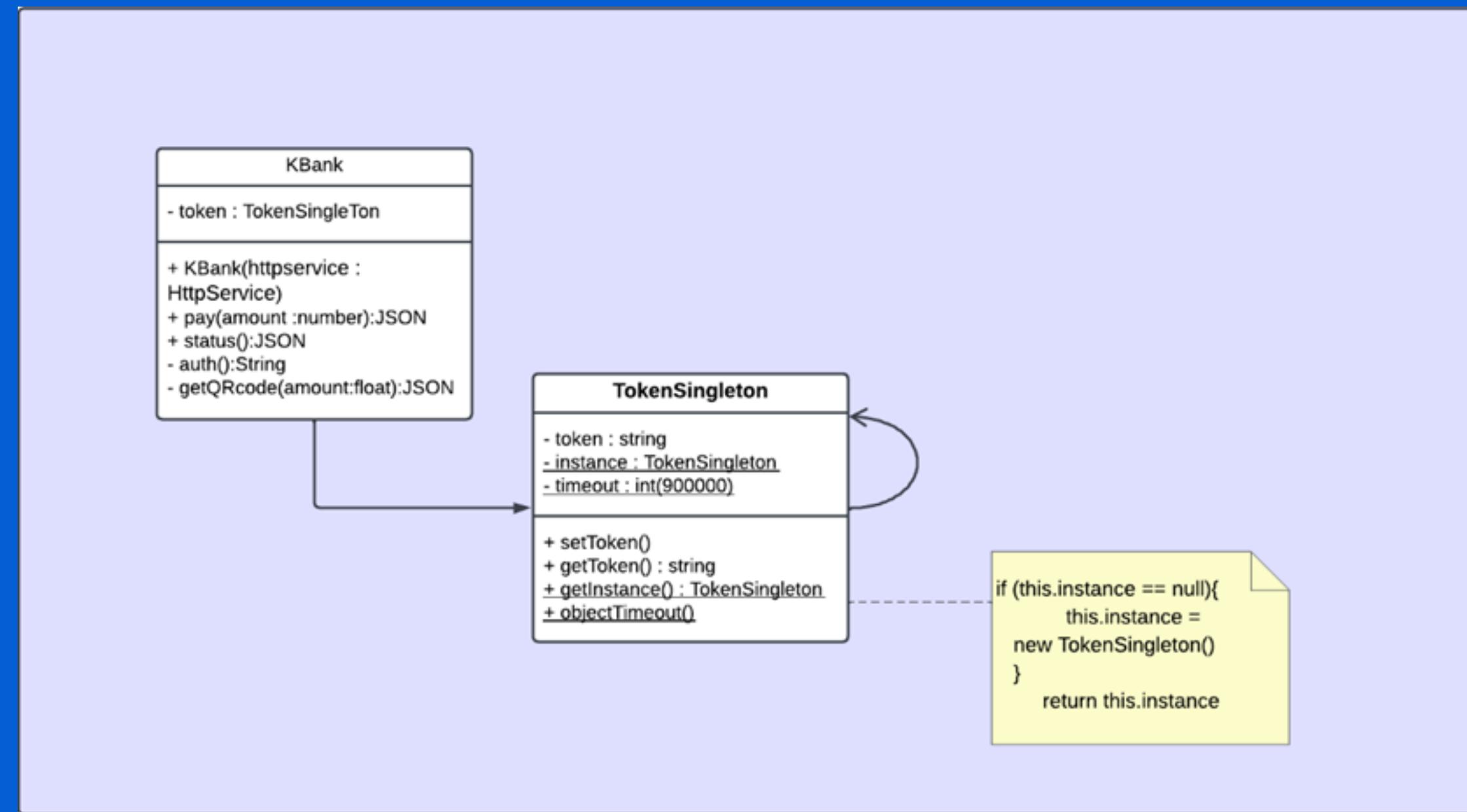
```
 1 export class Bank4QU implements PaymentStrategy{  
 2     private bankID = "4QU"  
 3     private readonly logger = new Logger(Bank4QU.name);  
 4     constructor(private httpService:HttpService){  
 5     }  
 6     public async pay(amount: number) {  
 7         let date = await new Date().toLocaleString('sv-SE').split('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

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



# 02

## SINGLETON



# ຕົວຢ່າງໂຄດ

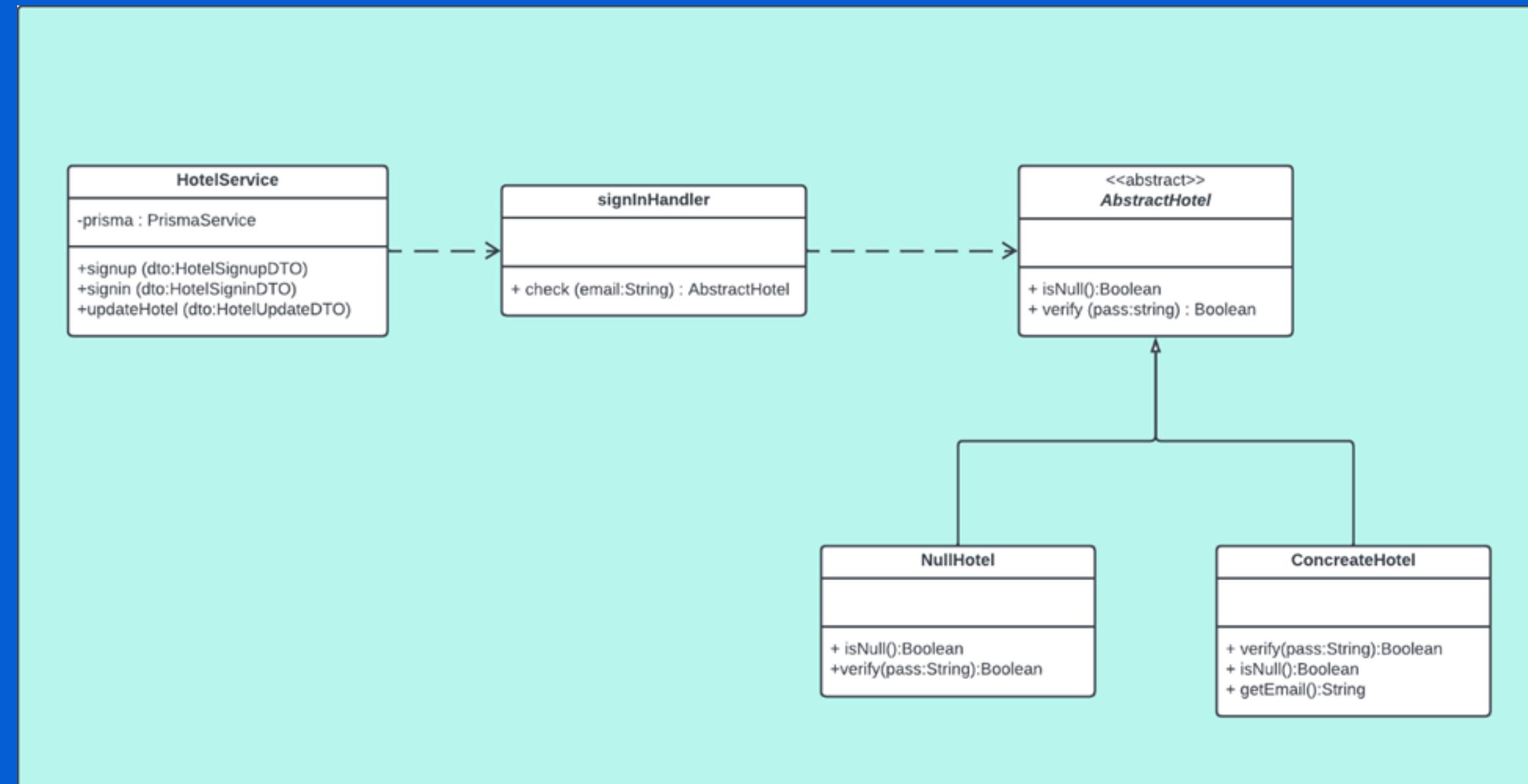
```
● ● ●  
1 public async pay(amount: number) {  
2     this.token = TokenSingleton.getInstance()  
3     if (this.token.getToken() == null){  
4         this.token.setToken(await this.auth())  
5         TokenSingleton.objectTimeout()  
6     }  
7     return await this.getQRcode(amount)  
8 }
```

```
● ● ●  
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 }
```



# 03

## NUL OBJECT



# ຕົວຢ່າງໂຄດ

```
1 export abstract class AbstractHotel{  
2     protected hotel  
3  
4     abstractisNull():boolean  
5     abstract verify(pass:string):Promise<Boolean>  
6 }
```

```
1 import { ForbiddenException } from "@nestjs/common";  
2 import { AbstractHotel } from "./AbstractHotel";  
3  
4 export class NullHotel extends AbstractHotel {  
5     constructor() {  
6         super()  
7     }  
8  
9     public isNull(): boolean {  
10        throw new ForbiddenException('Credentials incorrect');  
11    }  
12  
13    public async verify(pass): Promise<Boolean> {  
14        return false  
15    }  
16}  
17  
18 }
```

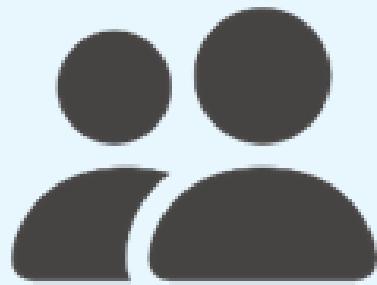
```
1 import { AbstractHotel } from "./AbstractHotel"  
2 import * as argon2 from "argon2";  
3  
4 export class ConcreateHotel extends AbstractHotel{  
5     constructor(hotel){  
6         super()  
7         this.hotel = hotel  
8     }  
9  
10    public isNull(): boolean {  
11        return false  
12    }  
13    public getEmail(): string {  
14        return this.hotel  
15    }  
16    public async verify(pass): Promise<Boolean> {  
17        return await argon2.verify(this.hotel.hash, pass)  
18    }  
19 }
```



# Quality Attributes Scenarios

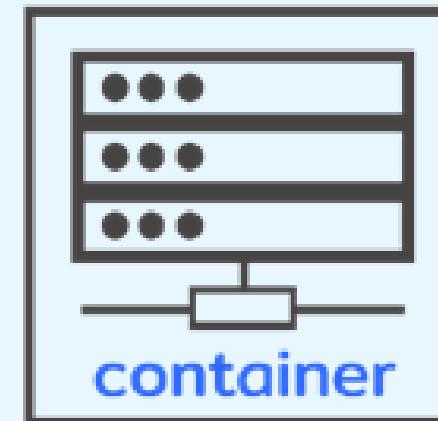


## 1 AVAILABILITY (REDUNDANT SPARE)



Developer

container เกิดการปิดตัวลง,  
การผลักกันเป็น active



Shutdown,  
Normal operation

จัดการโภนดที่มีปัญหา



เวลาที่ระบบใช้พื้นที่

# Quality Attributes Scenarios



2

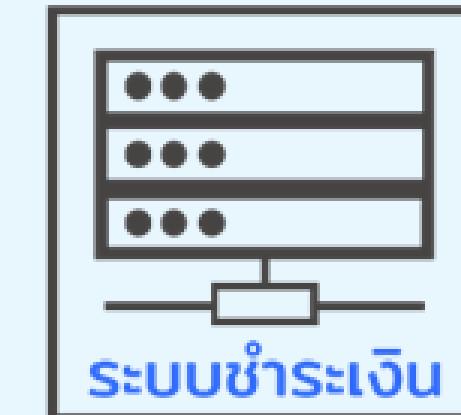
INTEGRABILITY



เจ้าของบ้าน  
HouseFlipper



ต้องการเพิ่มระบบชำระบัญชี  
จากธนาคาร 4QU



Development



การเพิ่มระบบอยู่ใน  
ช่วงทดสอบ



ระยะเวลาที่ใช้ในการเพิ่มระบบ

# Quality Attributes Scenarios

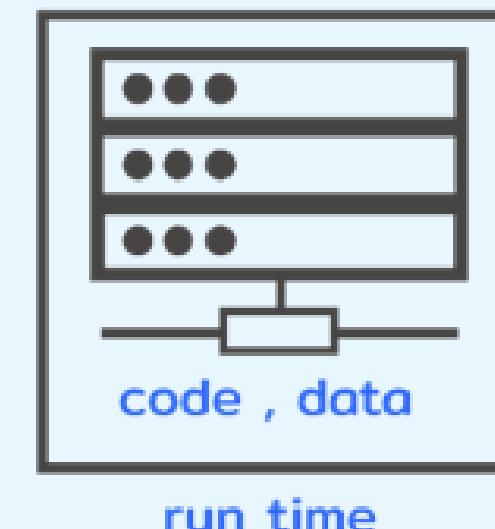


## 3

### MODIFICATION



ต้องการเพิ่มระบบ  
ชำรุดเสื่อม



Test modification



เวลาที่ใช้ , ความพยายาม

# Quality Attributes Scenarios

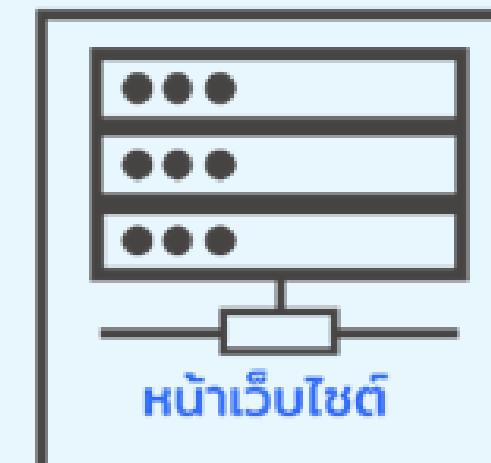


## 4 USABILITY



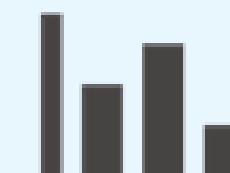
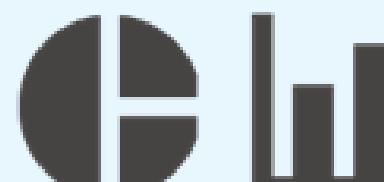
ผู้ใช้งาน, ลูกค้า

เรียนรู้การใช้งาน  
ระบบของเว็บไซต์



run time

feedback  
จากผู้ใช้งาน



ความพึงพอใจของผู้ใช้งาน

# DEMO



# THANK YOU

