



# MOTEIA

THE ORIGINAL APPLICATION BY

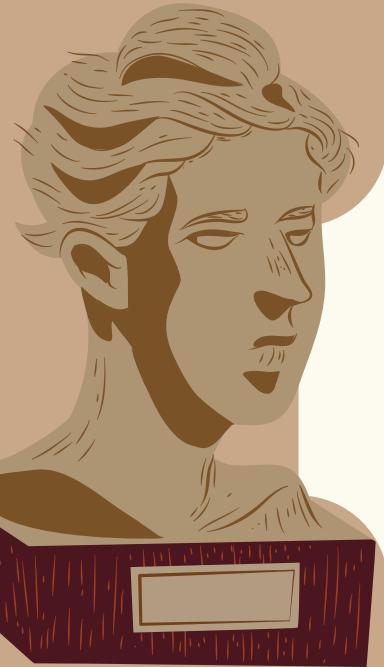
นักประดิษฐ์จีว G-9

# WEBSITE

# ຈອງໂຮງແຣນ ກີ່ພັກຕ່າງໆ



# SCOPE OF THE PROJECT



## บัญชีผู้ใช้งาน

User สามารถสร้างและใช้งานบัญชีส่วนตัวของตนเอง



## การกำลังดับคิว

ระบบสามารถระบุคิว และห้องว่างในโรงพยาบาลได้

## การค้นหาโรงพยาบาล

มีระบบในการ sort หาโรงพยาบาลที่ถูกต้องที่สุด หรือรีวิวดีที่สุด



## การชำระเงิน

User สามารถชำระเงิน และจองคิวได้ทันที



## การจองโรงแรม

ระบบสามารถจองได้ทันที เสมือนจองที่โรงแรม

# WELCOME TO MOTEUA

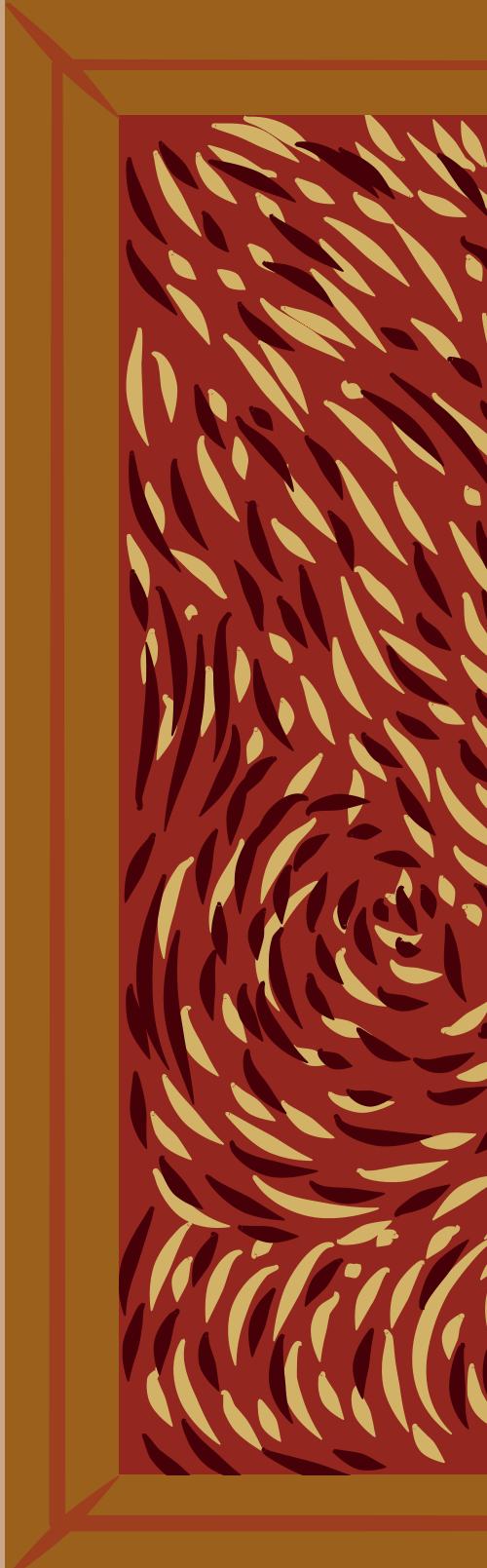
## PROBLEM

ปัญหาการเดินทางและการท่องเที่ยวแบบค้างคืนหรือระยะไกลนั้นสิ่งที่เรา  
จำเป็นต้องการหาคือสถานที่พักผ่อนที่ตรงตามต้องการดีและราคาถูกและ  
ปลอดภัย ซึ่งเป็นเรื่องที่ยากที่จะทำการจองได้สำเร็จทั้งเรื่องการหาสถานที่ที่จะ  
ต้องค้นหาทีละเว็บไซต์เรื่องการໂกรจองซึ่งเราไม่สามารถทราบได้เลยว่าเต็มคล้า  
ไม่ติดต่อไปก่อน เรื่องการชำระเงิน เป็นต้น เนื่องจากพวกราคาเสื่อมเหลือกึ่งการ  
ปัญหานั้นเราจึงจะทำเว็บแอพพลิเคชันที่ขับตอนในการจองที่ง่าย มีลิสต์  
โรงแรมให้เลือกมากมาย มีระบบชำระเงิน



# HOW TO FIX IT?

กลุ่มผู้พัฒนาได้เลิ�งเห็นปัญหาเหล่านี้ และ  
เริ่มที่จำพัฒนาแอพพลิเคชัน Motela  
เพื่อตอบโจทย์ปัญหา และผู้ใช้งานสามารถ  
ใช้แอพพลิเคชัน Motela ของทางเราได้  
อย่างสะดวกสบาย และง่ายมากที่สุด



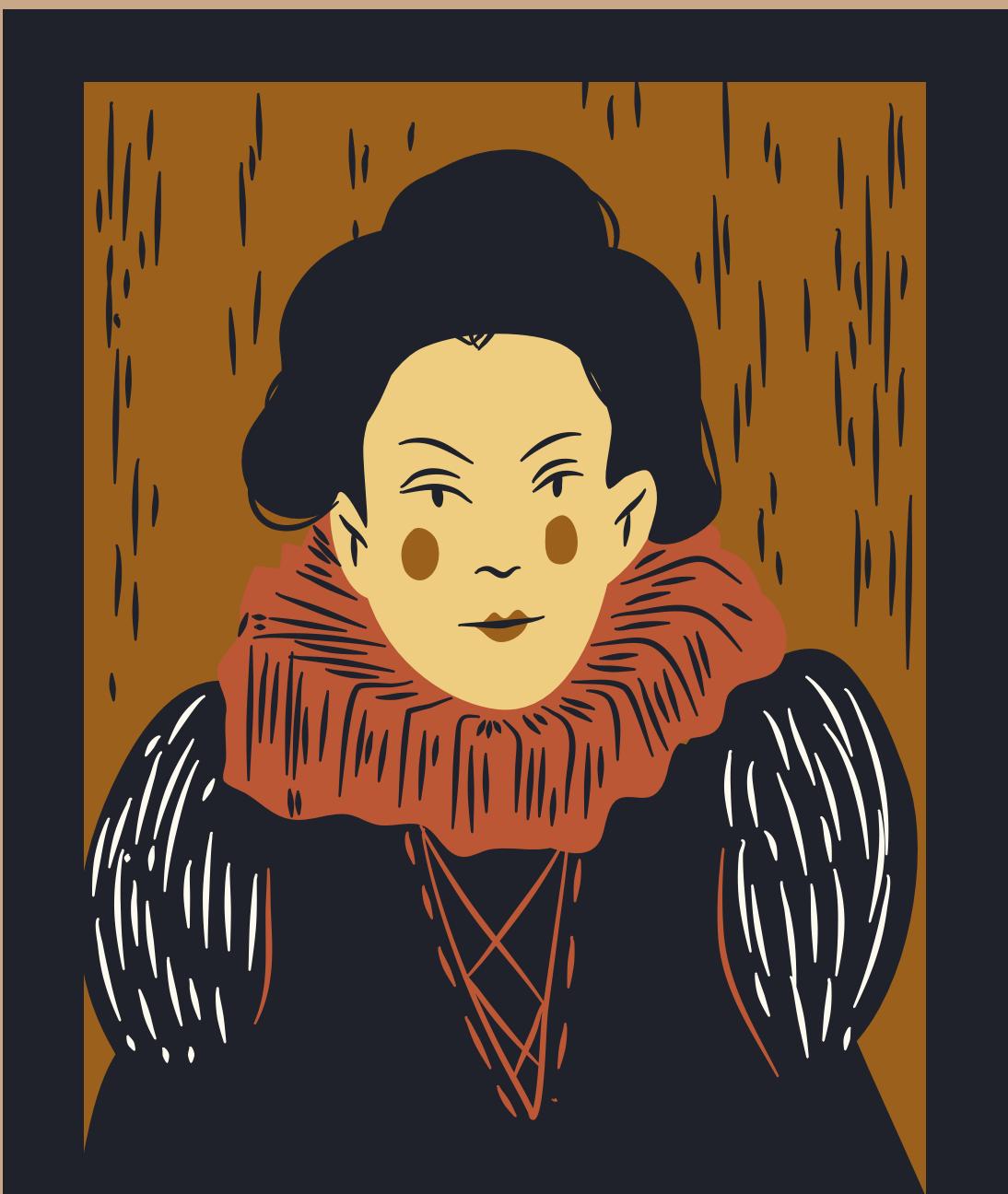
# DESIGN PATTERN



# PROTOTYPE

```
export class UserService {  
    constructor(@InjectRepository(UserEntity)private readonly userRepository: Repository<UserEntity>){  
    }  
    private users:User[]=[];  
    getUsers(){  
        return this.users.map((user) => new SerializedUser(user));  
    }  
}
```

ปัญหาที่พบคือ ไม่สามารถประกาศ entity เป็น global ได้  
จากนั้นจึงแก้ปัญหาด้วย วิธี prototype  
Original entity ถูกสร้างขึ้นและมีการสร้าง entity ขึ้นมาใหม่โดยอ้างอิงจาก original entity และentity ที่ถูกสร้างขึ้นมาใหม่สามารถถูกเรียกใช้ใน function อื่นได้



```
async createOffer(createOfferDto: CreateOfferDto, hotelId: string) {  
    const offer = new HotelOffer();  
    const hotel = this.hotelRepository.findOneBy({ hotelId });  
    offer.offerId = createOfferDto.offerId;  
    offer.emptyRoom = createOfferDto.emptyRoom;  
    offer.totalRoom = createOfferDto.totalRoom;  
    offer.offerPrice = createOfferDto.offerPrice;  
    offer.hotel = await hotel;  
    this.offerRepository.save(offer);  
    return offer;  
}  
  
async getAll() {  
    const hotel = await this.hotelRepository  
        .createQueryBuilder('hotel')  
        .leftJoinAndSelect('hotel.hotelOffer', 'hotelOffer')  
        .getMany();  
    return hotel;  
}
```

ปัญหาที่พบคือ ไม่สามารถรู้ได้ว่า hotel  
ไม่สามารถรู้ได้ว่าภายในมี offer ใดบ้าง  
จากนั้นจึงแก้ปัญหาด้วย วิธี composite  
Composite ทำให้โยง relation ไปทาง  
object ที่เล็กกว่าโดยใช้ query

# COMPOSITE



# MEDIATOR

```
@Entity()
export class Hotel_offer {
    @PrimaryGeneratedColumn('uuid')
    offerId: string;

    @Column()
    emptyRoom: number;

    @Column()
    totalRoom: number;

    @Column()
    offerPrice: number;

    @ManyToOne(() => Hotel, (hotel) => hotel.hotelOffer)
    @JoinColumn({ name: 'hotel_Id' })
    public hotel: Hotel;

    @OneToMany(() => Room, (room) => room.hotelOffer)
    public room: Room[];
}
```

```
@Entity()
export class Room {
    @PrimaryGeneratedColumn('uuid')
    roomId: string;

    @Column()
    offerId: string;

    @Column()
    roomNum: string;

    @Column({ default: true })
    isEmpty: boolean;

    @ManyToOne(() => Hotel_offer, (hotelOffer) => hotelOffer.room)
    @JoinColumn({ name: 'roomOffer_Id' })
    public hotelOffer: Hotel_offer;

    @OneToOne(() => Booking, (booking) => booking.room)
    public booking: Booking;
}
```

```
@Entity()
export class Hotel {
    @PrimaryGeneratedColumn('uuid')
    hotelId: string;

    @Column()
    hotelName: string;

    @Column()
    hotelContact: string;

    @Column()
    hotelLocation: string;

    @Column()
    hotelInfo: string;

    @Column()
    farFromMe: number;

    @Column()
    price: number;

    @Column()
    rating: string;

    @OneToMany(() => Hotel_offer, (hotelOffer) => hotelOffer.hotel, {
        cascade: true,
    })
    public hotelOffer: Hotel_offer[];

    @OneToOne(() => Picture, (picture) => picture.hotel)
    @JoinColumn({ name: 'picture_Id' })
    public picture: Picture;
}
```

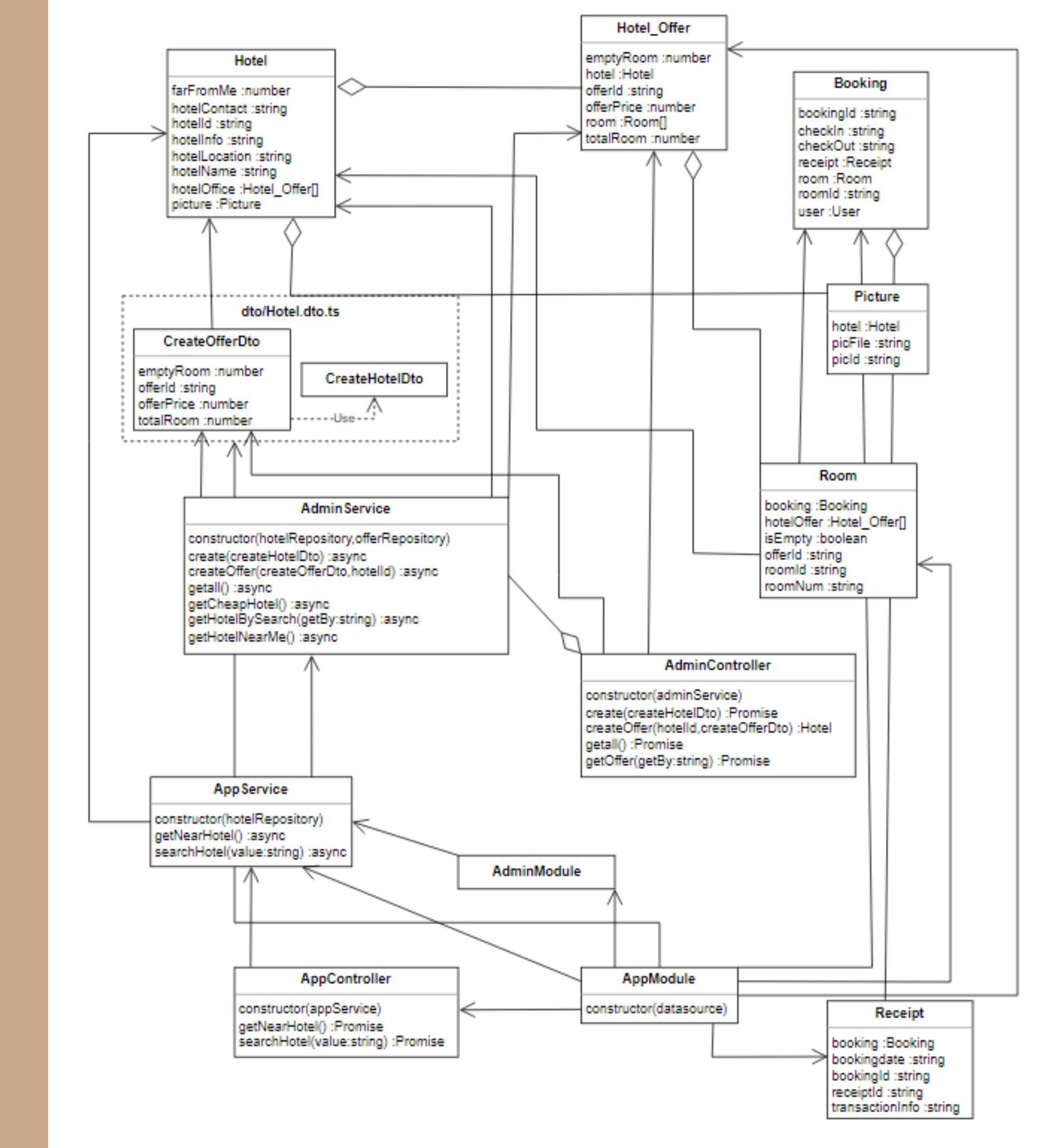
ปัญหาที่พบคือ หา object ตัวรองไม่เจอ  
จากนั้นจึงแก้ปัญหาด้วย วิธี Mediator  
Mediator ทำให้สามารถเชื่อมต่อ object  
ตัวกลางกับตัวรองได้



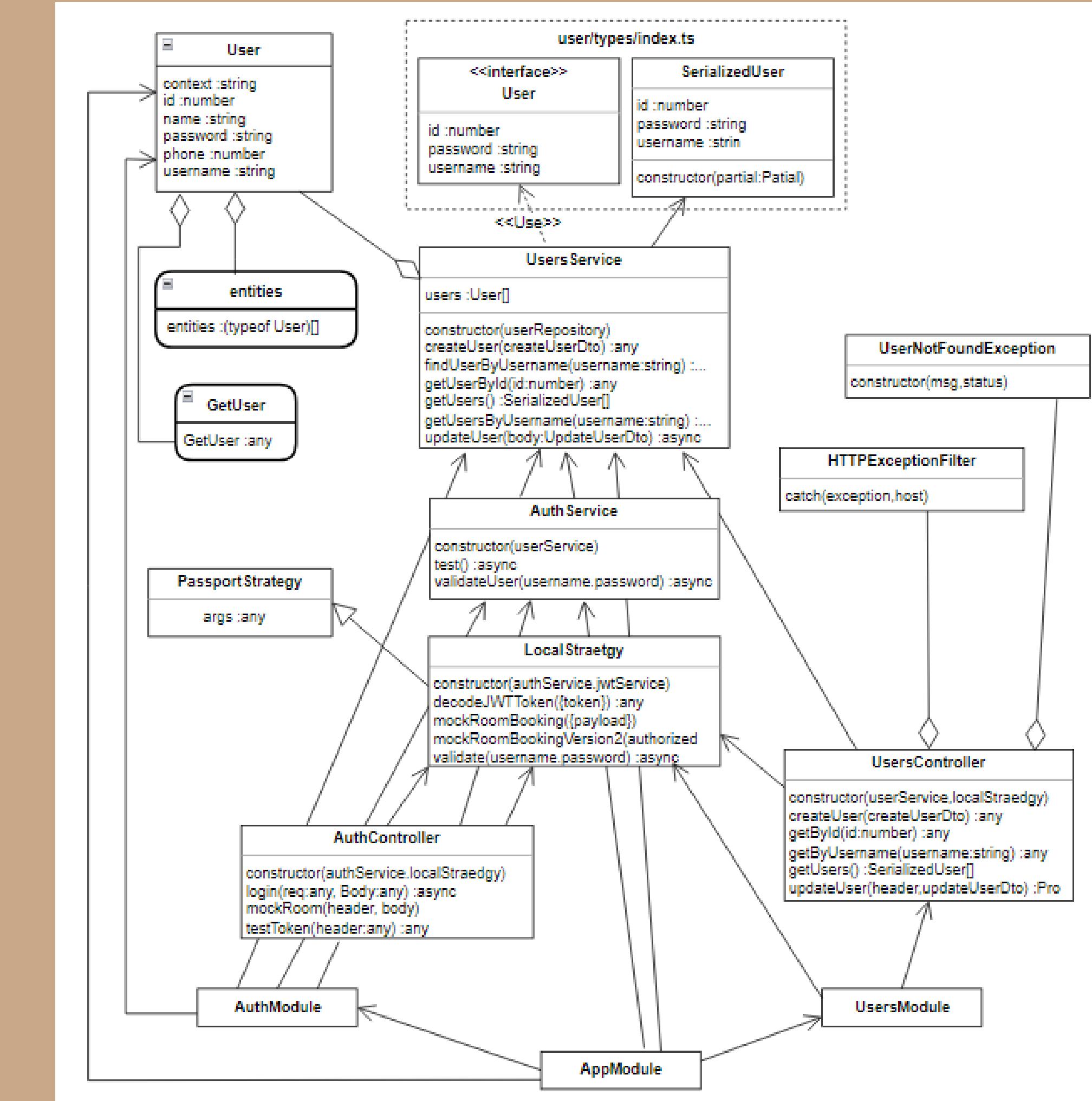
# SOFTWARE ARCHITECTURE AND DESIGN



# HOTEL



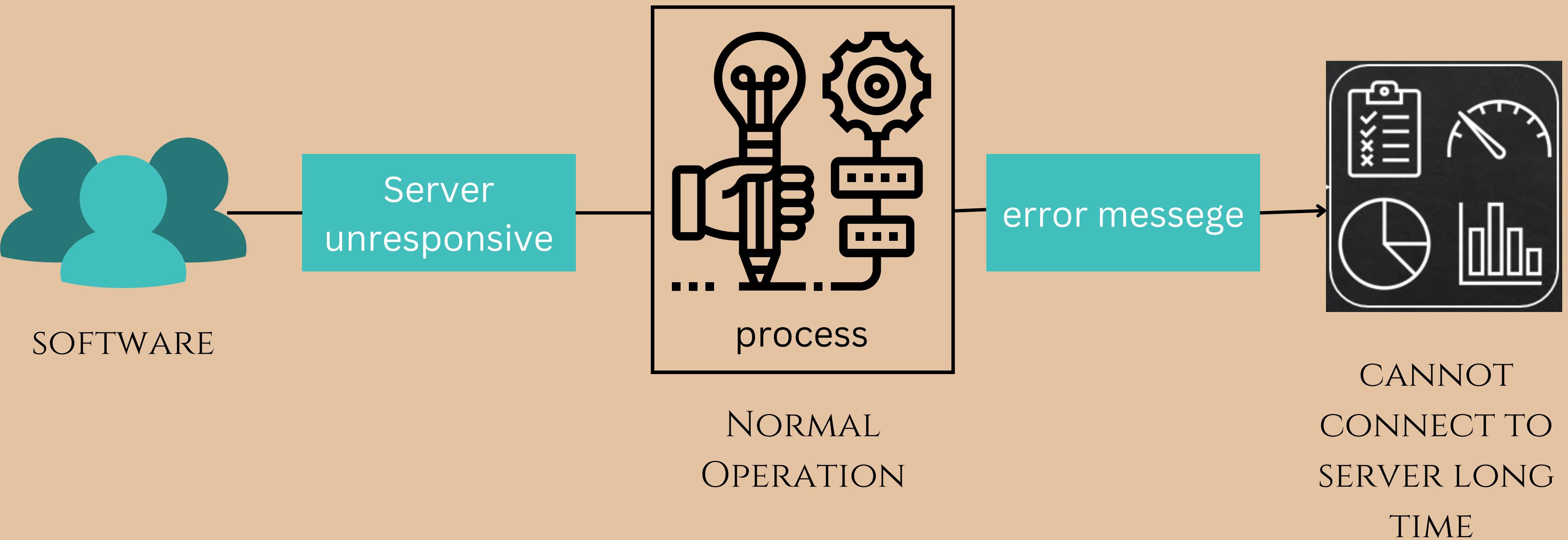
# USER



# QUALITY ATTRIBUTE SCENARIOS



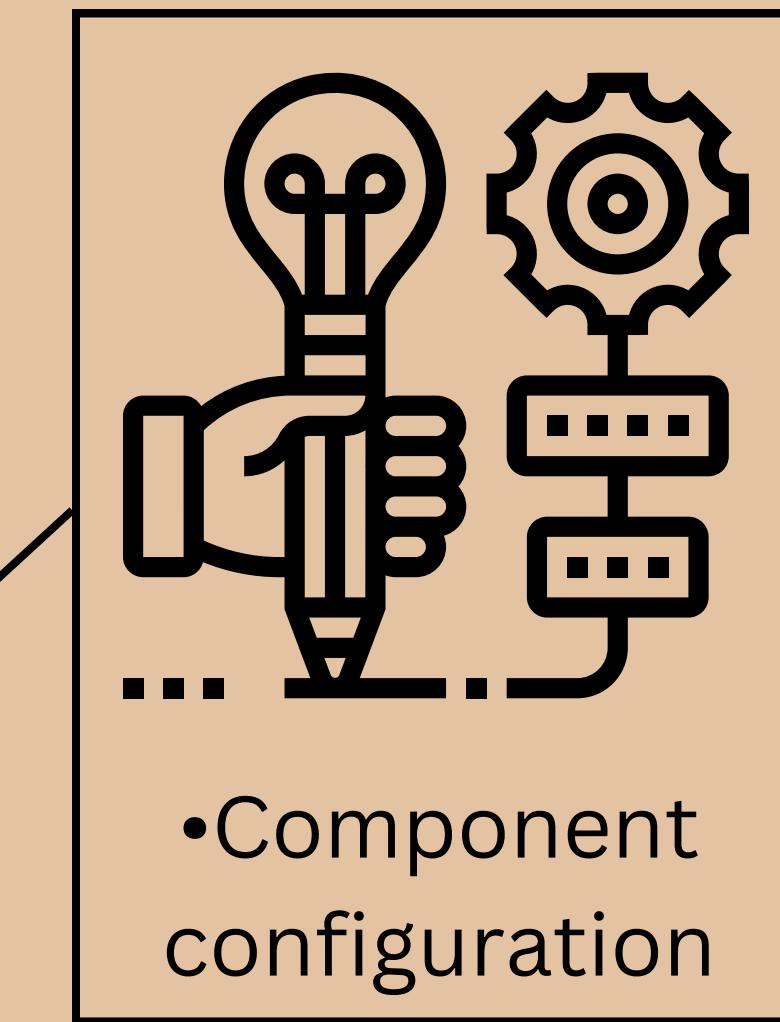
# AVAILABILITY: MONITOR



# INTEGRABILITY: CONFIGURE BEHAVIOR



- Integrate new version of existing component

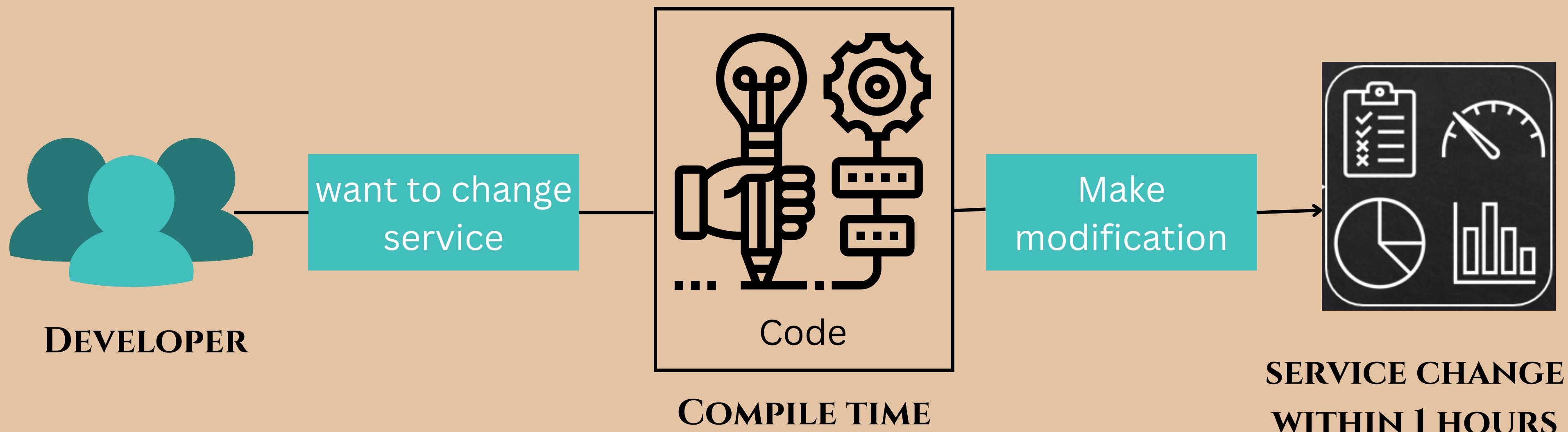


Changes are completed



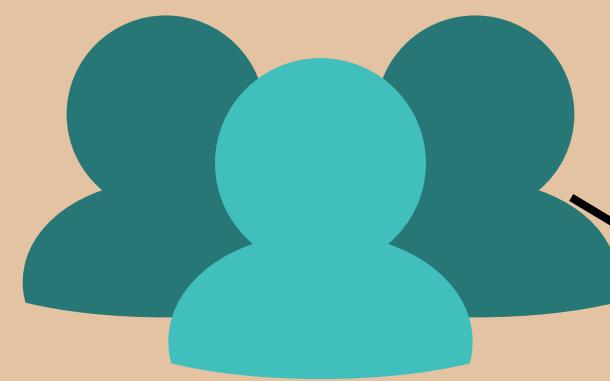
RUNTIME

# MODIFIABILITY :SPLIT MODULE



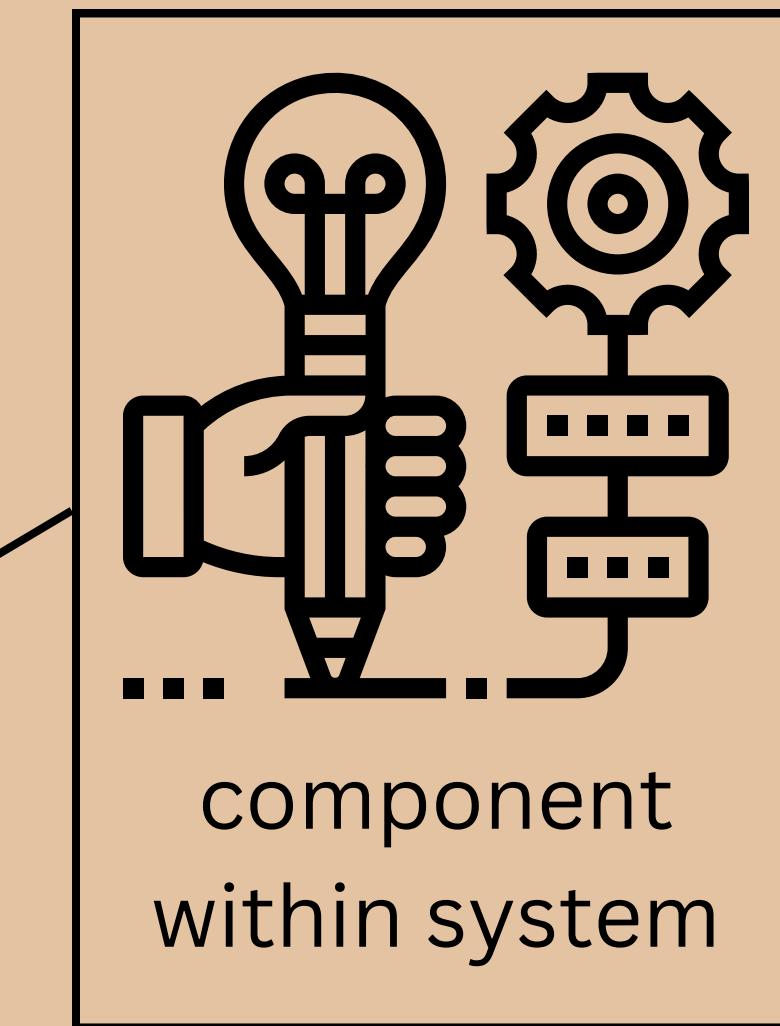
# PERFORMANCE :

## LIMIT EVENT RESPONSE



**USER  
REQUEST**

A periodic event arrives at a predictable interval.



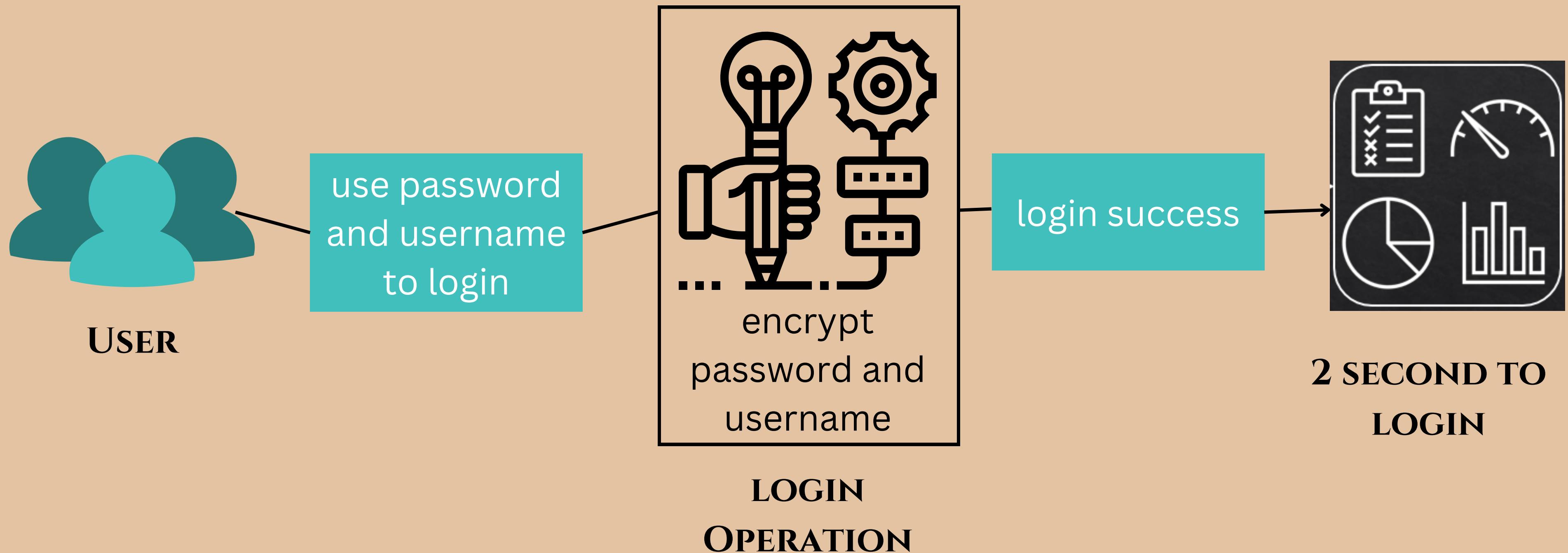
**peak mode**

System returns a response

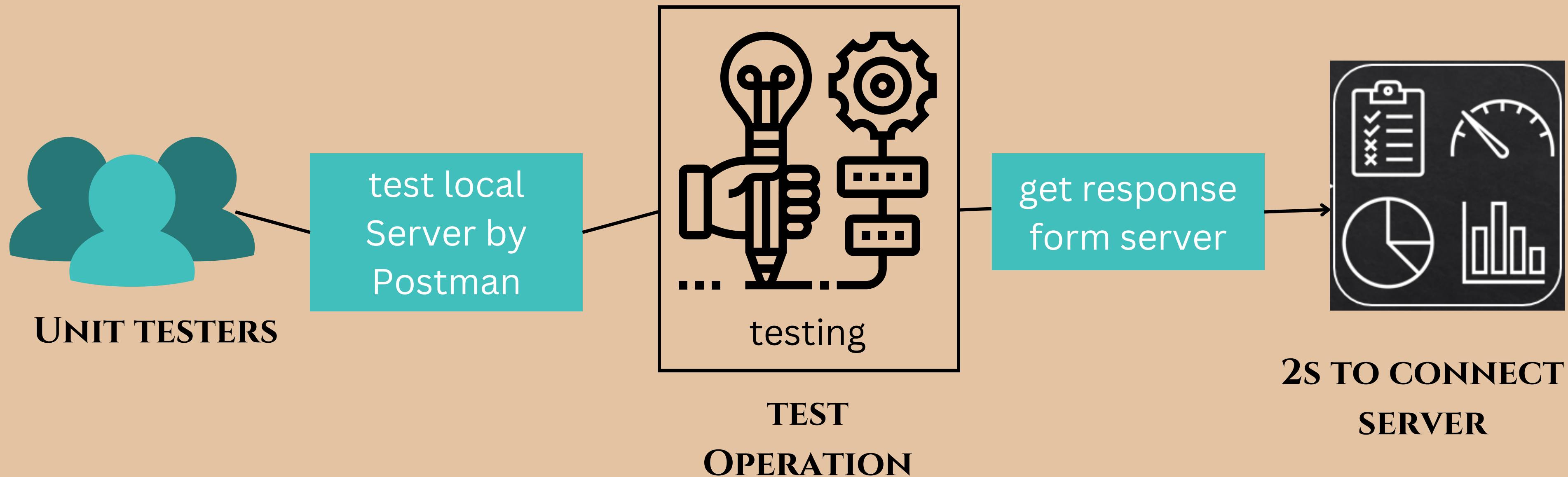


**RESPONSE TIME  
LATE  $\geq 2\text{ s}$**

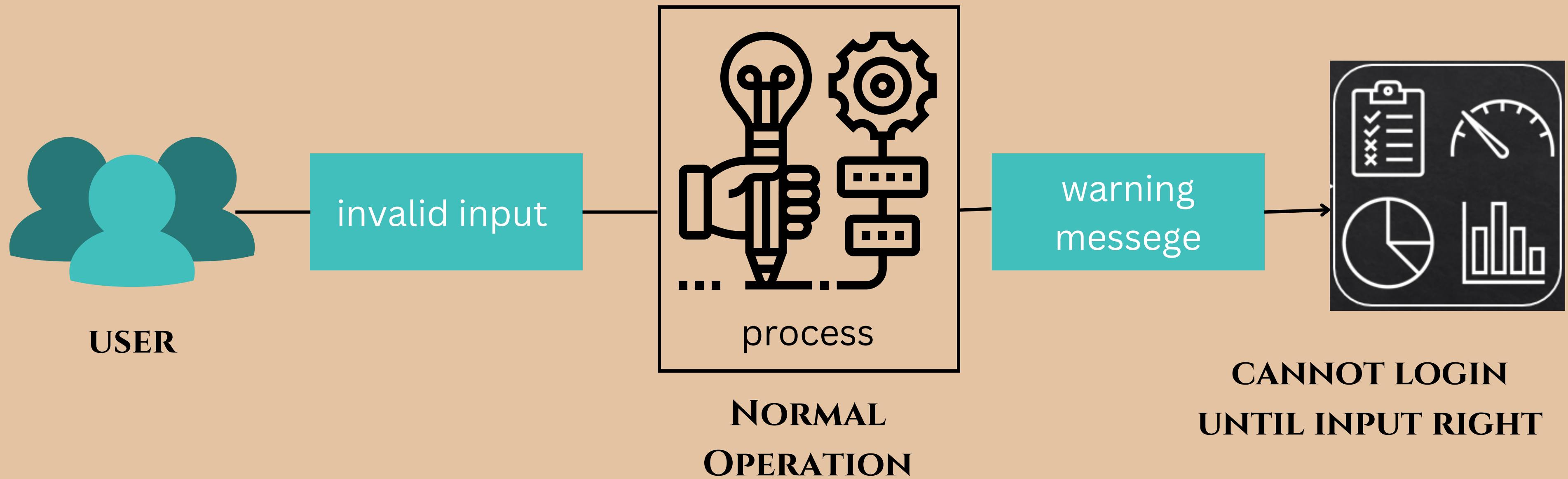
# SECURITY: ENCRYPT DATA



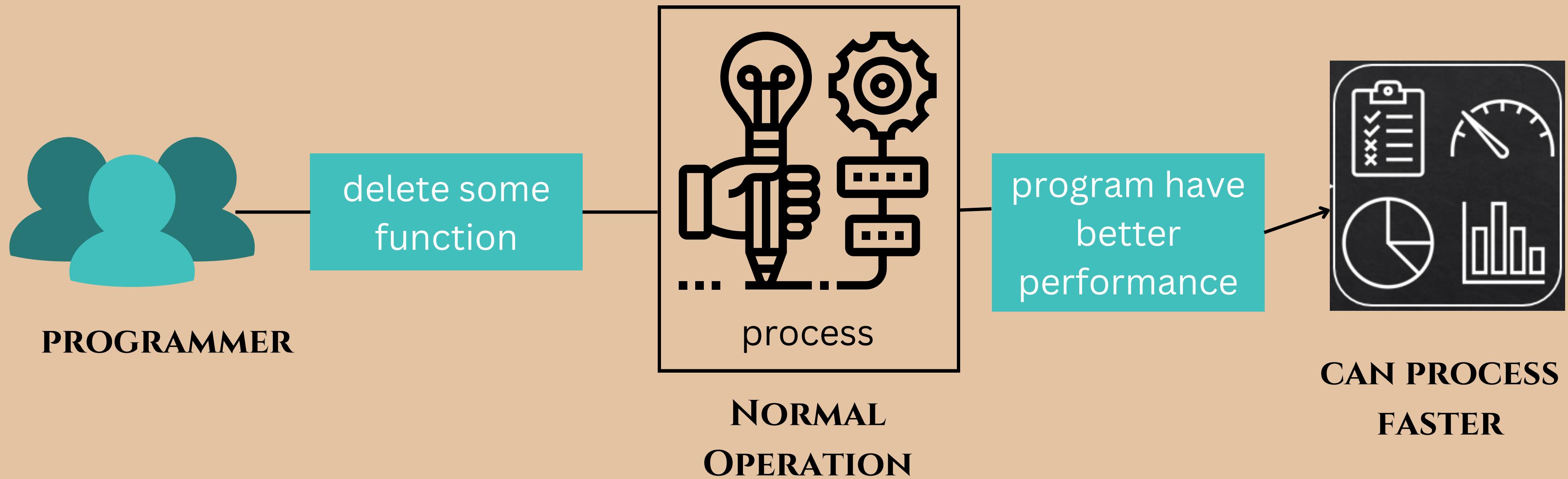
# TESTABILITY : LOCALIZE STATE STORAGE



# AVAILABILITY: EXCEPTION DETECTION



# AVAILABILITY: GRACEFUL DEGRADATION



# DEMO!