รายงานระบบจองคิวโรงพยาบาล QHospital

สมาชิกกลุ่ม

62010728	นายภูรินท์	บุญกระสินธ์
63010022	นายกฤต	รุ่งโรจน์กิจกุล
63010052	นายก้องเกียรติ	ชุนงาม
63010062	นางสาวกันต์กนิษฐ์	ทองเก๋ง
63010086	นางสาวกีรติกร	พลับพลา
63010226	นายชานน	เต็มกมลศิลป์
63010249	นางสาวญาณิศา	พงษ์เมธา
63010271	นางสาวณกมล	แซ่เฮง

นำเสนอ

ดร.ปริญญา เอกปริญญา

รายงานนี้เป็นส่วนหนึ่งของวิชา 01076024

สถาปัตยกรรมและการออกแบบซอฟต์แวร์

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Proposal

ปัญหา

เนื่องจากโรงพยาบาลเป็นสถานที่ที่ให้บริการเกี่ยวกับสุขภาพให้กับผู้ป่วย ทั้งการตรวจสุขภาพ การรักษาโรคหรือภัยที่เกิดจากอุบัติเหตุต่างๆ และยังมีเตียงสำหรับรับผู้ป่วยเพื่อเข้าพักรักษาตัวอีกด้วย ทำให้โรงพยาบาลนั้นเป็นสถานที่ที่มีคนเข้า-ออกอยู่ตลอดเวลา นอกจากนี้การรักษาในโรงพยาบาลนั้นก็ยังมีระดับความสำคัญในการเข้ารับรักษาอีกด้วย จึงทำให้การที่จะได้รับบริการกับทางโรงพยาบาลนั้นเป็นไปได้อย่างช้าหรือค่อนข้างใช้เวลา และแน่นอนว่าสำหรับทุกคนแล้วเวลาเป็นสิ่งที่มีค่าที่สุด ทุกคนล้วนต้องการที่จะได้รับการบริการกับโรงพยาบาลอย่างไวที่สุด โดยเฉพาะในช่วงเวลาที่ฉุกเฉินหรือเร่งด่วน ต่อให้ในปัจจุบันจะมีแอปพลิเคชันที่สามารถบอกถึงโรงพยาบาลที่อยู่ใกล้ผู้ป่วย รวมถึงจำนวนโรงพยาบาล ระยะทางหรือสภาพความคล่องของการจราจรบริเวณนั้นและเวลาที่ใช้ในการเดินทาง อย่างเช่น Google Map แต่ก็ยังไม่สามารถแสดงถึงจำนวนเตียงที่ว่างอยู่เพียงพอหรือไม่ จำนวนคิวที่มีอยู่ในขณะนั้น หรือจำนวนแพทย์และพยาบาลที่สามารถรองรับผู้ป่วยได้มีอยู่เท่าไหร่ได้ ทำให้ผู้ที่ต้องการรับการรักษากับทางโรงพยาบาลนั้นไม่สามารถคาดเดาเวลาที่จะได้เข้ารับการรักษากับทางโรงพยาบาลในแต่ละครั้งได้เลย

ดังนั้นจึงทำให้นักศึกษามีความคิดอยากที่จะทำโปรเจคนี้ขึ้นมาเพื่อเป็นเสมือนตัวกลางที่คอยรับข้อมูลว่าตอนนี้มีแพทย์ พยาบาล เตียงหรือคนไข้ที่อยู่ภายในโรงพยาบาลนั้นเป็นจำนวนเท่าไหร่บ้าง และ มีความพร้อมในการรับรักษาผู้ป่วยเพิ่มอีกจำนวนเท่าไหร่ เพื่อให้ผู้ที่ต้องการใช้บริการกับทางโรงพยาบาลนั้นสามารถตัดสินใจได้ง่ายขึ้น ว่าควรที่จะไปใช้บริการกับโรงพยาบาลไหนจึงได้รับการบริการได้อย่างรวดเร็วที่สุด

วิธีการแก้ปัญหา

กลุ่มนักศึกษาจึงได้คิดการแก้ปัญหาดังกล่าวด้วยการสร้างระบบคิว โดยเป็นเว็บไซต์ที่ผู้ป่วยสามารถจองคิวการเข้ารักษากับโรงพยาบาลที่ต้องการได้ ซึ่งในเว็บไซต์จะมีหน้าแสดงโรงพยาบาลที่อยู่ใกล้กับผู้ป่วย และในแต่ละโรงพยาบาลมีข้อมูลต่างๆ เช่น จำนวนคิวที่มีผู้คนจองในปัจจุบัน

จำนวนเตียงผู้ป่วย แพทย์ พยาบาลที่มี ระยะทางระหว่างผู้ป่วยกับโรงพยาบาล ระยะเวลาในการเดินทางไปโรงพยาบาลโดยประมาณ เป็นต้น สำหรับเป็นตัวช่วยในการตัดสินใจการจองคิวของผู้ป่วย นอกจากนั้นจะมีระบบจัดลำดับโรงพยาบาลตามหัวข้อที่เลือก เพื่อช่วยให้ผู้ป่วยค้นหาโรงพยาบาลที่ต้องการได้ง่ายขึ้น โดยหัวข้อที่ใช้ในการจัดลำดับสามารถแบ่งได้เป็น 2 กลุ่มย่อย ได้แก่

กลุ่มหัวข้อที่เป็นปัจจัยภายในโรงพยาบาล เช่น จำนวนผู้ป่วยที่โรงพยาบาลรองรับ, จำนวนแพทย์ พยาบาลที่พร้อมรับผู้ป่วย , รูปแบบโรค อาการเจ็บป่วยที่โรงพยาบาลรักษาได้ เป็นต้น

กลุ่มหัวข้อที่เป็นปัจจัยภายนอกโรงพยาบาล เช่น จำนวนคิวในโรงพยาบาล , ระยะทางในการเดินทางไปโรงพยาบาล , เวลาที่ใช้ในการเดินทางโดยประมาณ , ระดับความรุนแรงของอาการเจ็บป่วยของผู้ป่วย เป็นต้น

โดยข้อมูลบางส่วนจะมีการอัพเดทภายในช่วงเวลาที่สั้น เพื่อให้แสดงข้อมูลตรงกับข้อมูลในปัจจุบันเร็วที่สุด และเพื่อให้ผู้ป่วยมีข้อมูลในการเลือกจองคิวได้แม่นยำมากยิ่งขึ้น

ผู้ป่วยที่ต้องการจองคิวกับโรงพยาบาลหนึ่ง จะมีระบบสร้างคิวให้ผู้ป่วย โดยระบบจะทำการเก็บข้อมูลเกี่ยวกับผู้ป่วยเบื้องต้น เช่น เพศ อายุ ระดับความรุนแรงของอาการ คำอธิบายลักษณะอาการเบื้องต้น ให้แพทย์ พยาบาล หรือเจ้าหน้าที่ที่ดูแลคิวของโรงพยาบาลได้ทราบล่วงหน้า และอาจช่วยให้เตรียมการดูแลรักษากับผู้ป่วยที่มีความรุนแรงของอาการสูงได้ทัน

Software Architecture

Client-Server & N-tier

เป็นโครงสร้างซอฟต์แวร์ที่ประกอบด้วย 2 ส่วน คือ Client และ Server โดย

- Client จะเป็นผู้ส่ง Request ไปหา Server ก่อน

- Server ทำหน้าที่ตอบ Response กลับไปหา Client

ในกรณีเว็บไซต์ของพวกเรา

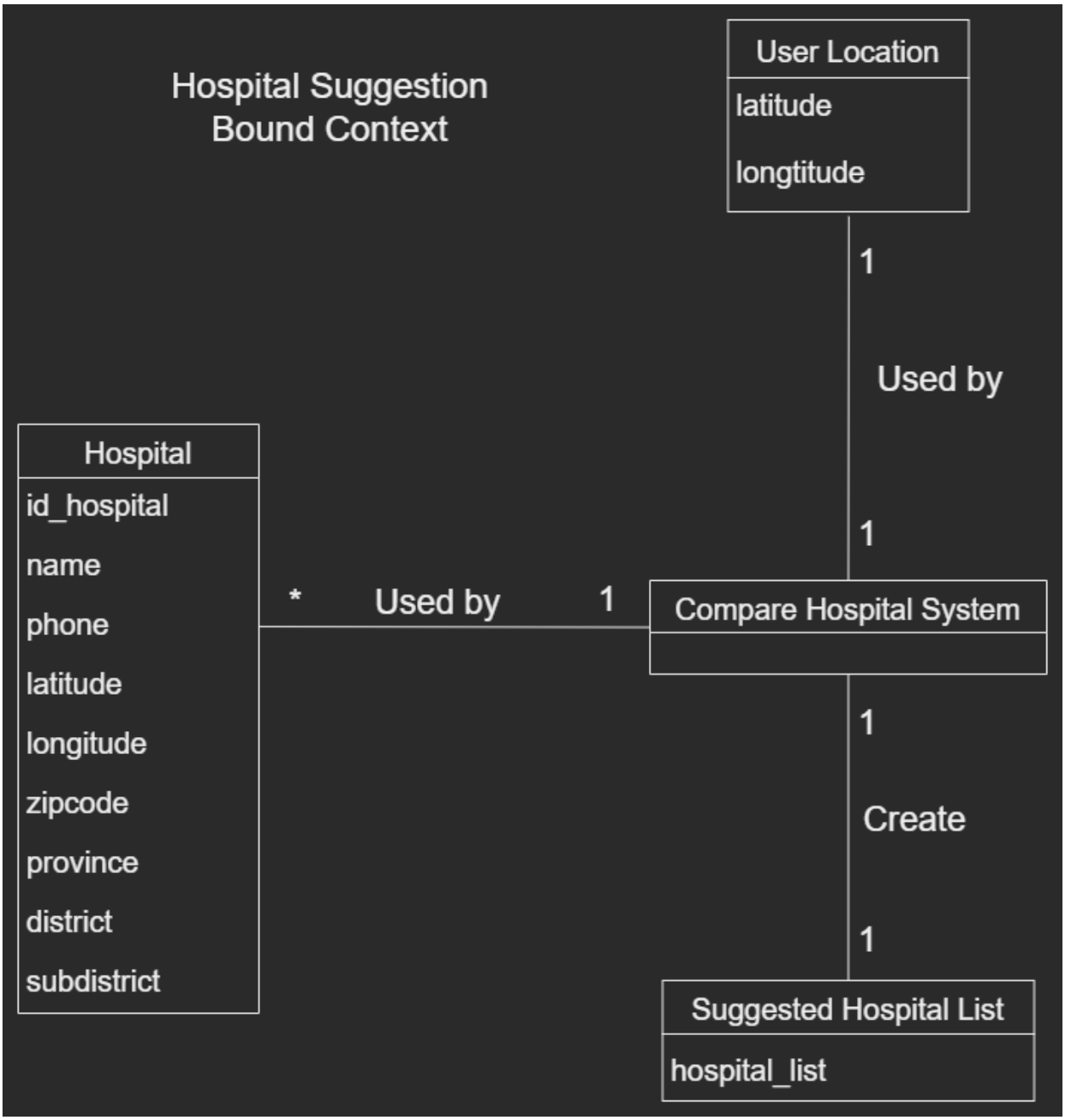
- ผู้ใช้ที่เข้ามาใช้งานเว็บไซต์ผ่านบราวเซอร์ ถือเป็น Client

- เว็บเซิร์ฟเวอร์ของเว็บไซต์ ถือเป็น Server

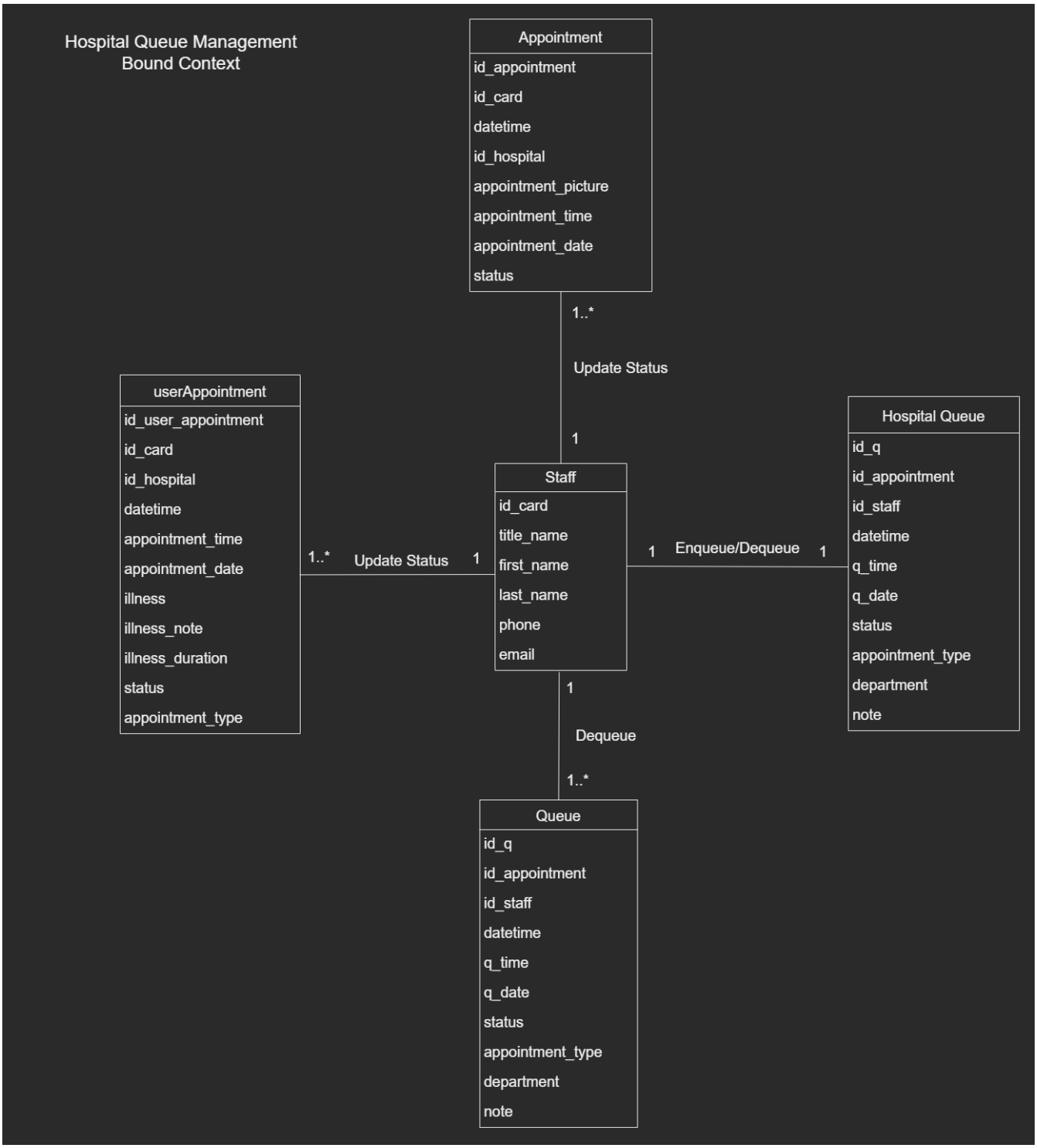
Software Design

Domain Model Diagram

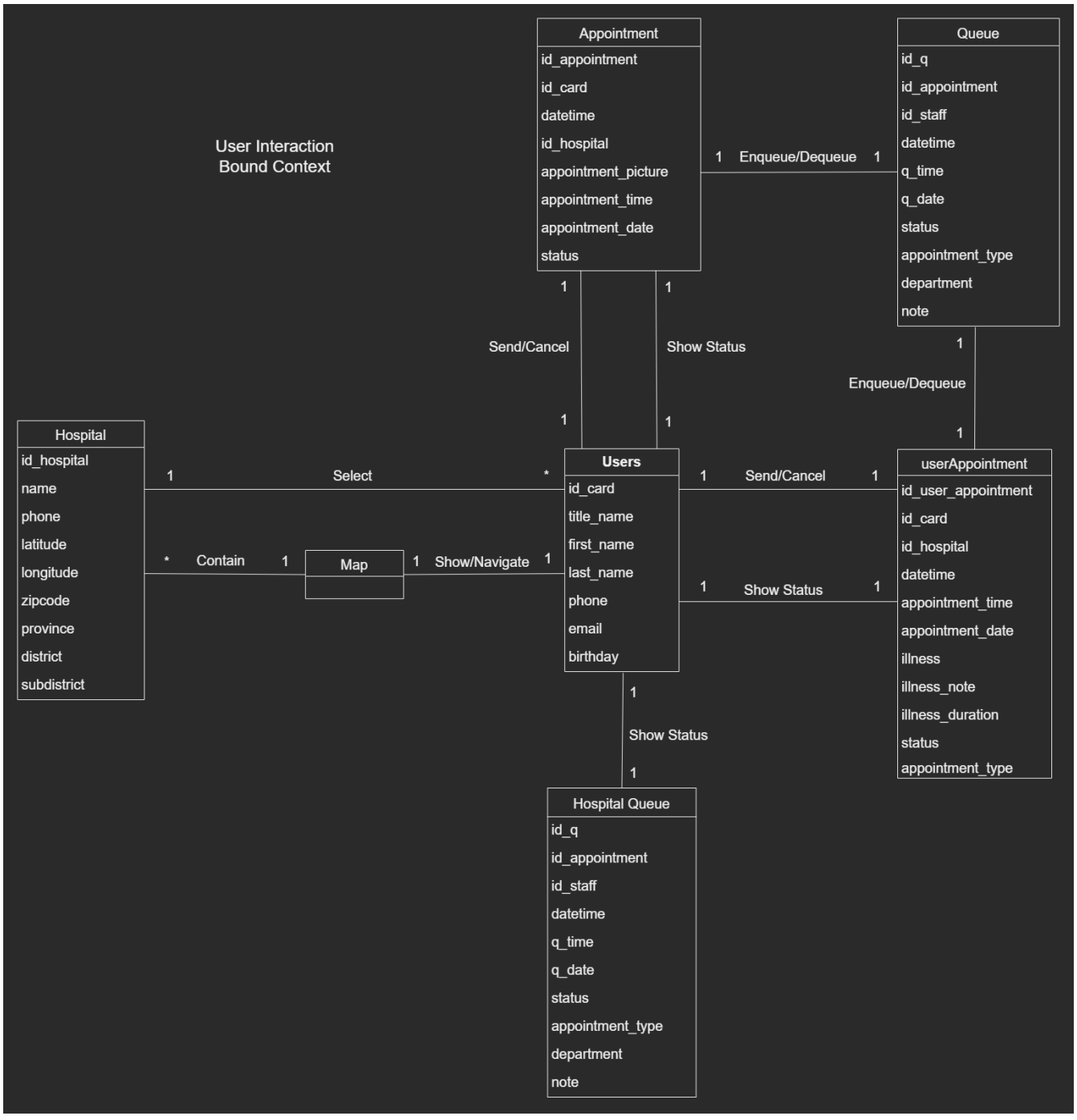
- **Hospital Suggestion Bound Context**



- **Hospital Queue Management Bound Context**



- **User Interaction Bound Context**



Design Patterns

Dependency Injection

ปัญหาที่พบ : Class หรือ Dependency ที่เกิดขึ้นระหว่าง 2 Class ใดๆ และปัญหาเรื่องความไม่ยืดหยุ่น (inflexible) ที่เกิดขึ้น หรือไม่ว่าจะเป็นปัญหาเรื่องความเปราะบาง (brittle) เมื่อมี Class แล้วต้องการ Parameter เพิ่ม

การแก้ไข : ใช้ Framework ที่เป็น Dependency Injection มาเพื่อแก้ไขปัญหา

ส่วนของโค้ดที่มีการใช้งาน :

```
@Injectable()
export class QueueService extends ResponseHandler {
    constructor(
        @InjectRepository(Queue)
        private readonly queueRepository: Repository<Queue>,
        @InjectRepository(Appointment)
        private readonly appointmentRepository: Repository<Appointment>,
        @InjectRepository(UserAppointment)
        private readonly userAppointmentRepository: Repository<UserAppointment>,
        @InjectRepository(Staff)
        private readonly staffRepository: Repository<Staff>,
    ) {
        super();
    }
```

```
@Injectable()
export class RegisterService extends ResponseHandler {
    constructor(
        @InjectRepository(Registration)
        private readonly registrationRepository: Repository<Registration>,
        @InjectRepository(User)
        private readonly userRepository: Repository<User>,
    ) {
        super();
    }
```

Adapter

ปัญหาที่พบ : เนื่องด้วย Library ที่เราใช้งานมีการนำมีการปรับเปลี่ยนการทำ Middlewares แตกต่างไปจาก Request กับ Response มาใช้งานในส่วนอื่น ฉะนั้นจึงต้องมีการ implement class เพื่อให้ตรงกับรูปแบบที่เราต้องการนำไปใช้งานต่อใน Flow การทำงานแบบเดิม

การแก้ไข : มีการนำ adapter มาช่วยให้ทำงานต่อกันได้

ส่วนของโค้ดที่มีการใช้งาน :

```
import { MiddlewareConsumer } from '../middleware/middleware-consumer.interface';
export interface NestModule {
    configure(consumer: MiddlewareConsumer): any;
}
```

```
@Module({
    imports: [
        TypeOrmModule.forRoot(configService.getTypeOrmConfig()),
        AuthModule,
        TestModule,
        AppointmentModule,
        HospitalModule,
        MapModule,
        QueueModule,
        RegistrationModule,
        StaffModule,
        TimelistModule,
        UserAppointmentModule,
        UsersModule,
    ],
})
export class AppModule implements NestModule {
    configure(consumer: MiddlewareConsumer) {
        consumer
            .apply(LoggerMiddleware)
            .forRoutes({
                path: '*',
                method: RequestMethod.ALL,
            });

        // apply auth
        consumer
            .apply(AuthMiddleware)
            .exclude('api/(register|login)')
            .forRoutes({
                path: '*',
                method: RequestMethod.ALL,
            })
    }
}
```

Template Method

ปัญหาที่พบ : Response มี Pattern การ Response ที่ค่อนข้างซ้ำกัน

การแก้ไข : นำ Template Method มาใช้ในการสร้าง Response แต่ละแบบ

ส่วนของโค้ดที่มีการใช้งาน :

```
export abstract class ResponseAbstract {
    public abstract responseBadRequest();
    public abstract responseUnauthorized();
    public abstract responseForbidden();
    public abstract responseMethodNotAllowed();
    public abstract responseConflict();
}
```

Quality Attribute Scenarios

Availability

Scenario 1:

Source of Stimulus: Server ของเว็บไซต์

Stimulus: ไม่ตอบสนองต่อ request

Artifact: Process

Environment: ช่วงเวลาทำงานปกติ

Response:

- แจ้งเตือนให้ผู้ดูแลทราบ
- หยุดการทำงานของ server เพื่อทำการแก้ไขข้อผิดพลาดที่เกิดขึ้น

Response Measure: ระยะเวลาที่ server ไม่สามารถใช้งานได้จะไม่เกิน 1 ชั่วโมง

Scenario 2:

Source of Stimulus: คนดูแลพัฒนา backend ของเว็บไซต์

Stimulus: แก้ไขโค้ดผิดรูปแบบ

Artifact: Process

Environment: ช่วงเวลาทำงานปกติ

Response:

- แจ้งเตือนให้ผู้พัฒนาทราบ
- ปิดระบบเว็บไซต์เพื่อทำการแก้ไขโค้ดที่เกิดข้อผิดพลาด

Response Measure: ระยะเวลาปิดเว็บไซต์และทำการแก้ไขจะใช้ไม่เกิน 30 นาที

Scenario 3:

Source of Stimulus: ฮาร์ดแวร์ของ Server เว็บไซต์

Stimulus: เครื่อง Server ดับ

Artifact: Communications channels

Environment: ช่วงเวลาทำงานปกติ

Response:

- บันทึกข้อผิดพลาด
- แจ้งเตือนให้ผู้ดูแลทราบบน monitor
- ปิดการให้บริการ server ที่เกิดข้อผิดพลาดและทำการแก้ไข

Response Measure: ระยะเวลาในการแก้ไขจะใช้ไม่เกิน 1 ชั่วโมง

Integrability

Scenario 1:

Source of Stimulus: ตลาดคอมโพแนนท์

Stimulus: Integrate เวอร์ชั่นใหม่ของคอมโพแนนท์ที่มีอยู่ในระบบ

Artifact: ระบบทั้งหมด

Environment: หลังเปิดการทำงานระบบ

Response: ผ่านการทดสอบ integrate และ integrate สำเร็จ

Response Measure: ใช้เวลาในการทำไม่เกิน 14 วันทำงาน

Modifiability

Scenario 1:

Source of Stimulus: ทีมผู้พัฒนา

Stimulus: ต้องการเพิ่มฟังก์ชันการทำงานใหม่

Artifact: โค้ด

Environment: ช่วงเวลาการเขียนโค้ด

Response: เพิ่มฟังก์ชันการทำงานใหม่ได้สำเร็จ

Response Measure: ไม่ส่งผลกับการทำงานส่วนอื่นๆ

Performance

Scenario 1:

Source of Stimulus: ผู้ใช้งาน 2000 คน

Stimulus: ทำการจองคิวโรงพยาบาล 900 รายการภายในระยะเวลา 1 นาที

Artifact: ระบบจัดการคิวโรงพยาบาล

Environment: ช่วงเวลาทำงานปกติ

Response: ระบบจัดลำดับคิวโรงพยาบาลได้ทั้งหมด

Response Measure: ระยะเวลาในการจัดลำดับคิวเฉลี่ยไม่เกิน 5 วินาที

Scenario 2:

Source of Stimulus: ผู้ใช้งาน 300 คน

Stimulus: ทำการยกเลิกคิวที่จอง 300 รายการภายในระยะเวลา 3 นาที

Artifact: ระบบจัดการคิวโรงพยาบาล

Environment: ช่วงเวลาทำงานปกติ

Response: ระบบนำคิวออกจากคิวโรงพยาบาลได้ถูกต้องทั้งหมด

Response Measure: ระยะเวลาในการตรวจสอบและจัดลำดับคิวใหม่เฉลี่ยไม่เกิน 10 วินาที

Scenario 3:

Source of Stimulus: ผู้ใช้งาน 200 คน

Stimulus: ทำการเข้าดูตำแหน่งโรงพยาบาลใน Google map 150 ครั้งใน 1 นาที

Artifact: ระบบเชื่อมต่อ Google map

Environment: ช่วงเวลาทำงานปกติ

Response: ระบบแสดงตำแหน่งโรงพยาบาลใน Google map ได้ถูกต้อง

Response Measure: ระยะเวลาในการค้นหาและแสดงตำแหน่งโรงพยาบาลเฉลี่ยไม่เกิน 5 วินาที