เว็บไซต์จัดการการเช่าจักรยาน

สมาชิก

นางสาวริตา ฉิมน้อย
643020488-2 SECTION 2

นายศุภโชค คำตะลา
643020492-1 SECTION 1

นางสาวกัลยาณี สอนสิงห์
643021216-0 SECTION 2

ที่มาและความสำคัญ

มหาวิทยาลัยขอนแก่นมีขนาดพื้นที่ประมาณ 12,259 ไร่ สำหรับคนที่ไม่มียานพาหนะการเดินทางจะลำบากมาก เพื่อที่จะแก้ไขปัญหาเหล่านี้ทางผู้จัดทำถึงได้คิดหายานพาหนะที่จะช่วยให้ผู้คนเดินทางได้โดยสะดวกและไม่ต้องเสียเวลานาน ซึ่งก็คือจักรยาน เนื่องจากจักรยานสามารถขี่ได้ง่ายและคนส่วนใหญ่ขี่เป็น จึงเป็นยานพาหนะที่เหมาะสมที่สุดและจักรยานยังมีประโยชน์อีกมากมาย เช่น ถือเป็นการออกกำลังกายและยังช่วยลดการก่อมลพิษทางอากาศได้

จากที่กล่าวมาข้างต้น ทางผู้จัดทำจึงต้องการที่จะแก้ไขโดยการจัดทำเว็บไซต์จัดการการเช่าจักรยาน เพื่อให้ทุกคนสามารถขอยืมจักรยานได้ โดยจะใช้บัตรนักศึกษาหรือบัตรประชาชนในการขอยืม และสามารถขี่จักรยานได้ในมหาวิทยาลัยขอนแก่นเท่านั้น

วัตถุประสงค์

เพื่อช่วยให้นักศึกษามหาวิทยาลัยขอนแก่นสามารถประหยัดค่าใช้จ่ายในการเดินทางและสามารถเดินทางรอบมหาวิทยาลัยขอนแก่นได้อย่างสะดวก

ประโยชน์ที่คาดว่าจะได้รับ

01

นักศึกษาสามารถประหยัดค่าใช้จ่ายในการเดินทางรอบๆ มหาวิทยาลัยขอนแก่น

02

นักศึกษาสามารถเดินทางได้อย่างสะดวก

ทฤษฎีที่เกี่ยวข้อง

SOLID

หลักการที่ใช้ออกแบบซอฟต์แวร์เพื่อให้ระบบสามารถยืดหยุ่น โดยประกอบไปด้วย

1. Single Responsibility Principle มีหน้าที่รับผิดชอบหน้าที่เดียว
2. Open-Closed Principle โมดูลควรเปิดให้เพิ่มความสามารถแต่ไม่ควรปิดกั้นการแก้ไข
3. Liskov Substitution Principle หลักการของแทนที่ของลิสคอฟคือคลาสย่อยสามารถแทนที่คลาสหลักได้โดยที่ระบบยังคงทำงานได้ถูกต้อง
4. Interface Segregation Principle หลักการแบ่งอินเตอร์เฟซคือคลาสควรมีอินเตอร์เฟสที่เป็นเพียงส่วนหนึ่งของที่จำเป็นและไม่จำเป็นต้องสนับสนุนเมธอดที่ไม่ใช้
5. Dependency Inversion Principle คลาสควรขึ้นอยู่กับสิ่งที่เป็นส่วนหนึ่งของระบบไม่ควรขึ้นอยู่กับรายละเอียด

ทฤษฎีที่เกี่ยวข้อง

CRUD

แนวคิดพื้นฐานที่ใช้ในการเขียนโปรแกรมหรือใช้พัฒนาแอปพลิเคชันที่มีการปฏิบัติตามหลักการสำคัญ 4 ประการ ซึ่งประกอบไปด้วยส่วนที่จะทำหน้าที่สร้าง (Create) อ่าน (Read) อัปเดต (Update) และลบ (Delete) ดังนั้นแต่ละอัปเดตจะถูกแทนที่ด้วยคำสั่งหรือเมธอดที่เหมาะสมสำหรับการดำเนินการนั้นๆ

ทฤษฎีที่เกี่ยวข้อง

Design Patterns

เป็นพิมพ์เขียวสำหรับการออกแบบซอฟต์แวร์ โดย Design Pattern จะแบ่งออกเป็น 3 กลุ่ม

1. Creational Patterns เป็นกลุ่มที่ไว้ใช้สร้าง Object ในรูปแบบต่างๆ ให้มีความยืดหยุ่น และนำโค้ดมาใช้ซ้ำได้
2. Structural Patterns จะเป็นวิธีการนำ Object และ Class มาใช้งานร่วมกัน สร้างเป็นโครงสร้างที่มีความซับซ้อนยิ่งขึ้น โดยยังมีความยืดหยุ่นและทำงานได้อย่างมีประสิทธิภาพ
3. Behavioral Patterns เป็นวิธีการออกแบบการติดต่อกันระกว่าง Object ให้มีความยืดหยุ่นและสามารถติดต่อกันกันได้อย่างไม่มีปัญหา

เครื่องมือที่เกี่ยวข้อง

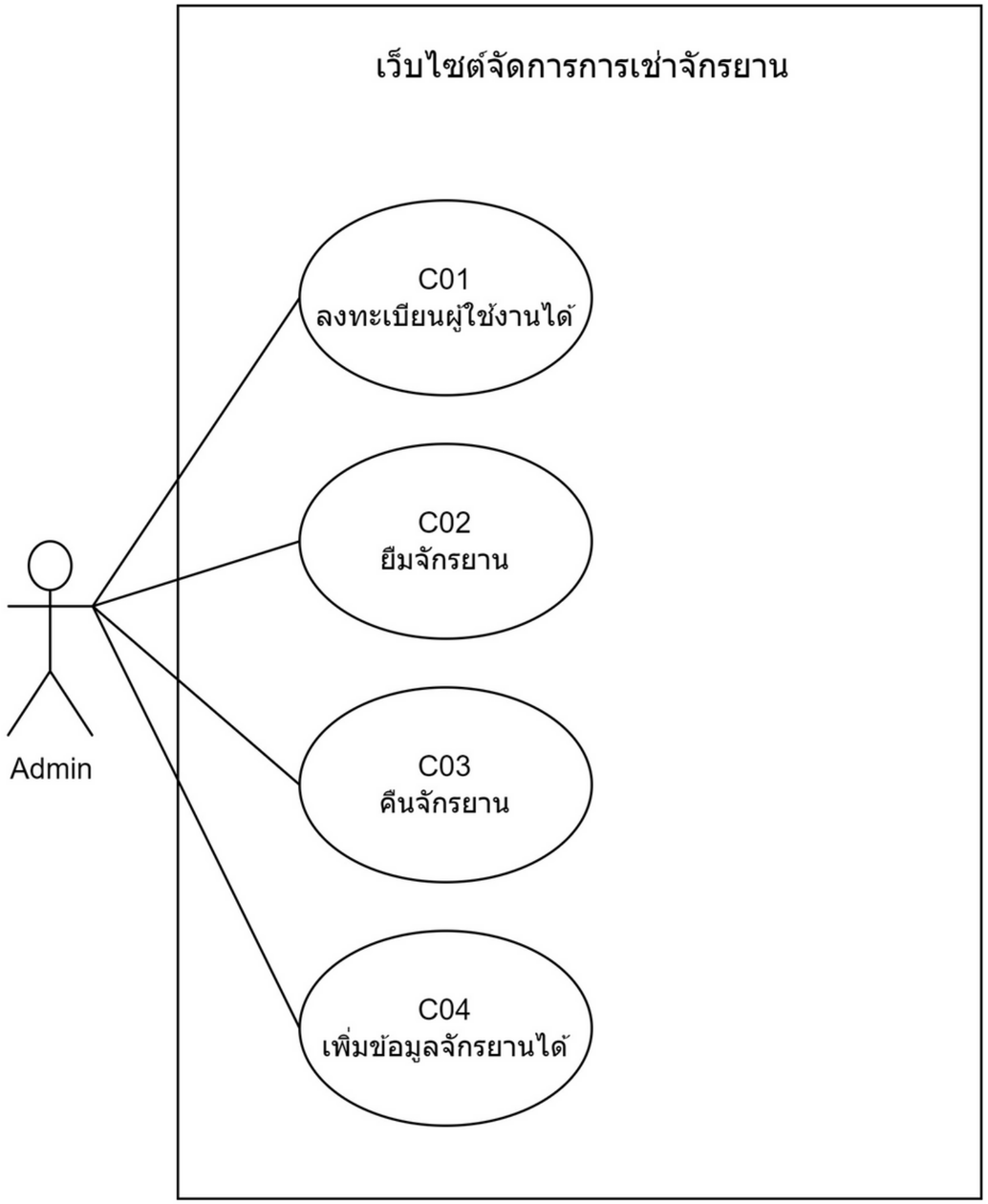
MySQL

ระบบจัดการฐานข้อมูลแบบข้อมูลเชิงสัมพันธ์ โดย MySQL มีหน้าที่จัดเก็บข้อมูลอย่างเป็นระบบ รองรับคำสั่งภาษา SQL เพื่อจัดการกับฐานข้อมูลโดยเฉพาะ

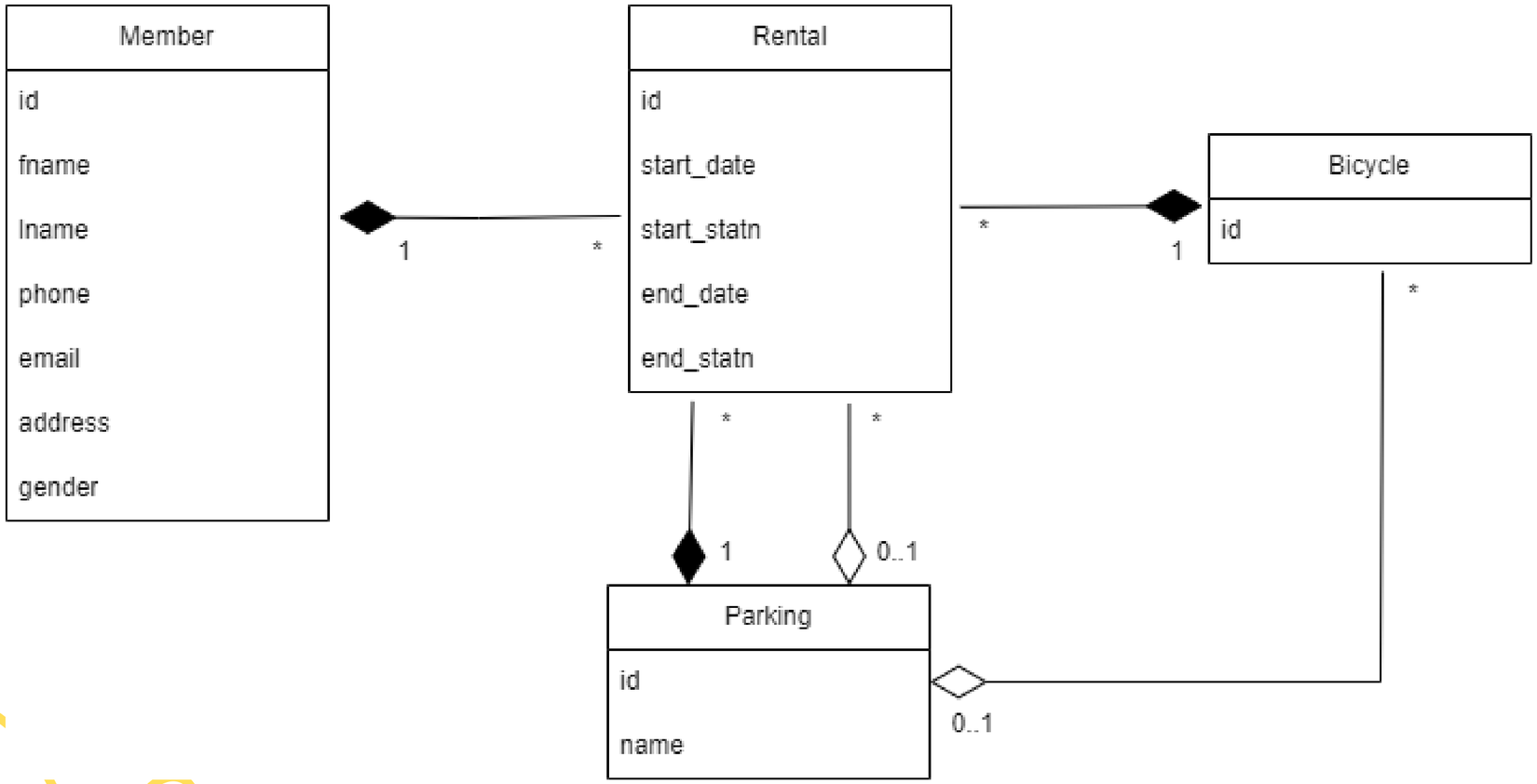
spring boot

Framework ใน Spring อันหนึ่ง ช่วยให้สร้าง Web application ได้ง่ายขึ้น เพราะมี Auto Configuration และสามารถใช้งานได้ทันที

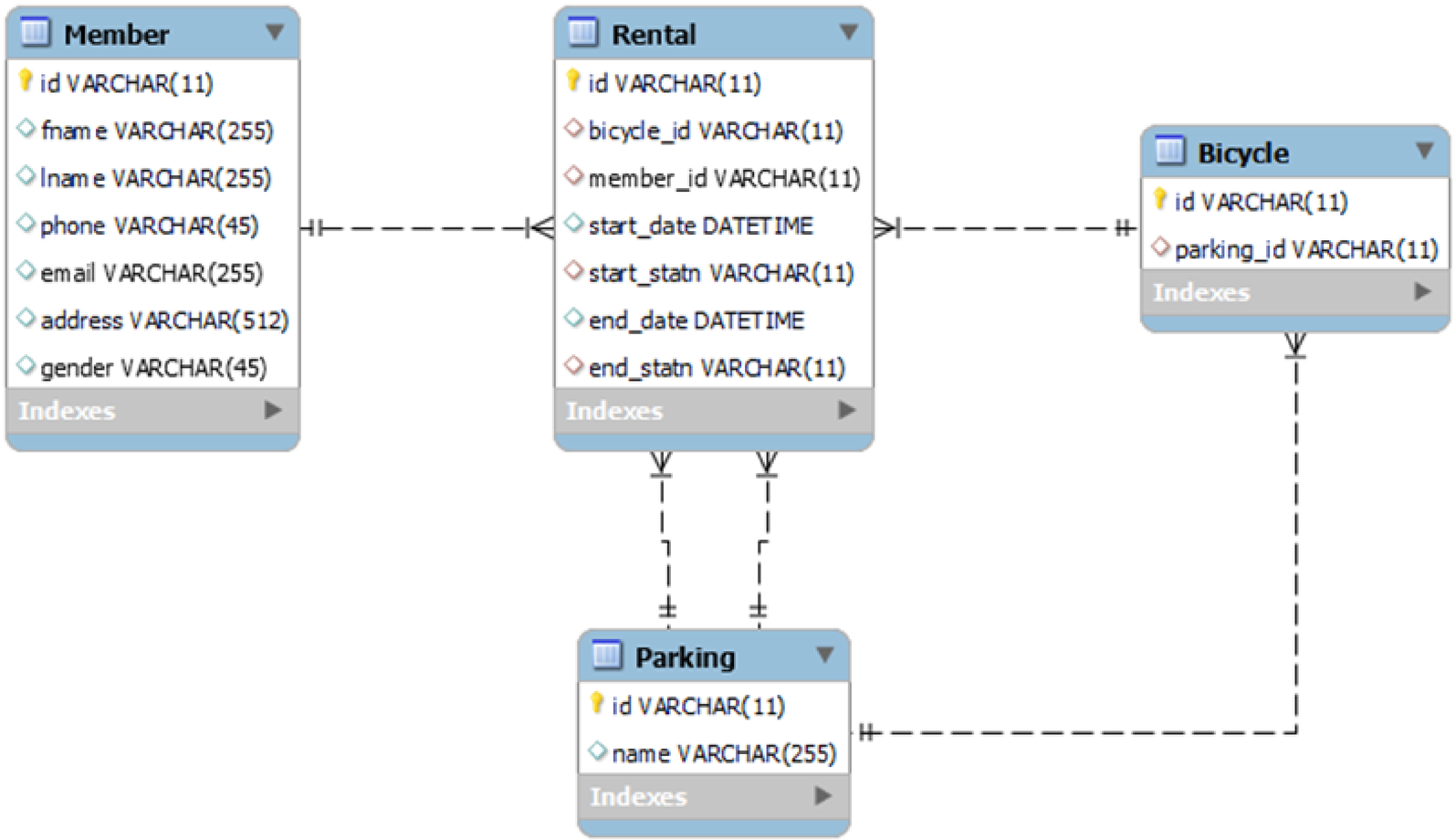
UML USE CASE DIAGRAM



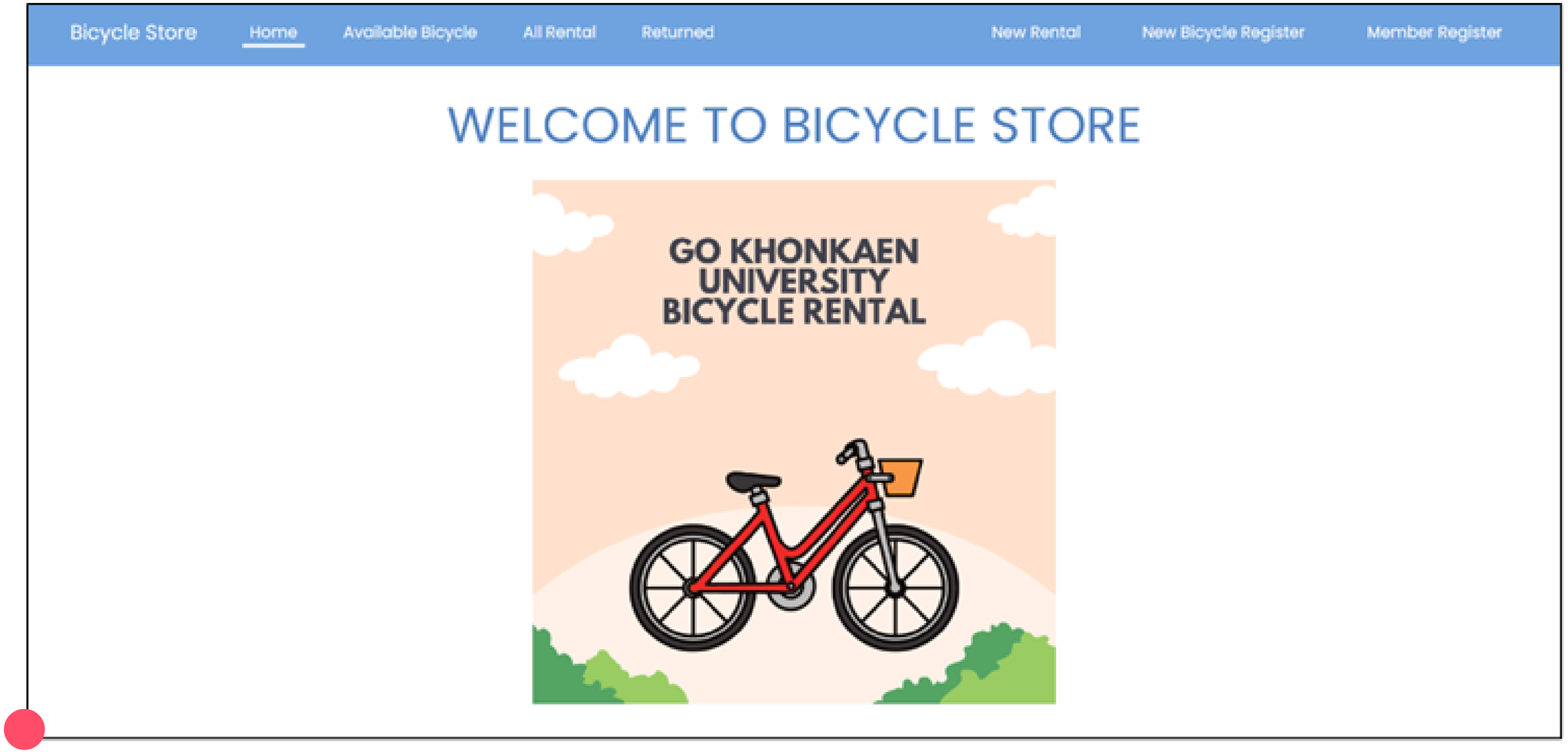
CLASS DIAGRAM



ER DIAGRAM


Member
id VARCHAR(11)
fname VARCHAR(255)
lname VARCHAR(255)
phone VARCHAR(45)
email VARCHAR(255)
address VARCHAR(512)
gender VARCHAR(45)
Indexes
Rental
id VARCHAR(11)
bicycle_id VARCHAR(11)
member_id VARCHAR(11)
start_date DATETIME
start_statn VARCHAR(11)
end_date DATETIME
end_statn VARCHAR(11)
Indexes
Bicycle
id VARCHAR(11)
parking_id VARCHAR(11)
Indexes
Parking
id VARCHAR(11)
name VARCHAR(255)
Indexes

DESIGN USER INTERFACE



PATTERN ใน SPRING ที่ใช้ในระบบ

Class BicycleService

```
11  import com.cp.repository.BicycleRepository;
12
13  @Service
14  public class BicycleService {
15      @Autowired
16      private BicycleRepository biRepo;
17
18      public boolean isExistByBicycleId(String biId) {
19          Optional bi = biRepo.findById(biId);
20          return bi.isEmpty();
21      }
22
23      public List<Bicycle> findAllBicycle() {
24          return (List<Bicycle>) biRepo.findAll();
25      }
26
27      public List<Bicycle> findAllByParking(Parking parking) {
28          return (List<Bicycle>) biRepo.findByParking(parking);
29      }
30      public List<Bicycle> findAllAvailable(){
31          return (List<Bicycle>) biRepo.findByParkingNotNull();
32      }
33
34      public Bicycle findBicycleById(String biId) {
```

- ในบรรทัดที่ 15 ได้เรียกใช้ @Autowired เพื่อช่วยในการสร้างและจัดการ injection ของคลาส BicycleRepository ให้กับตัวแปร biRepo
- หลักการ Dependency Injection Principle (DI) ใช้ในคลาส BicycleService ได้เรียกใช้คลาส BicycleRepository เพื่อทำงานกับฐานข้อมูลและยังใช้ตาม
- หลักการ Low-level Component คลาส BicycleRepository เป็นส่วนที่อยู่ต่ำกว่า และให้ BicycleService เป็นส่วนที่อยู่สูงกว่า High-level Component ซึ่งเป็นหลักปฏิบัติที่ถูกต้องตามหลัก Dependency Injection Principle

Class BicycleRepository

```
5   import org.springframework.data.repository.CrudRepository;
6
7   import com.cp.entity.Bicycle;
8   import com.cp.entity.Parking;
9   import com.cp.entity.Rental;
10
11  public interface BicycleRepository extends CrudRepository<Bicycle, String>{
12      List<Bicycle> findByParking(Parking parking);
13      List<Bicycle> findByParkingIsNull();
14      List<Bicycle> findByParkingNotNull();
15  }
16
```

- ได้เรียกใช้ Autowired ในส่วนบรรทัดที่ 12 โดยให้ทำการ findByParking จากคลาส Parking
- ได้ใช้หลักการ Dependency Inversion (DI) เช่น บรรทัดที่ 12 คือทำหน้าที่รับคลาส Parking เข้ามา และสร้าง object parking โดยในส่วนโค้ดตรงนี้ไม่ได้สร้างตัวแปรขึ้นใหม่แต่รับผ่าน Spine Container

Class Bicycle

```java
@Entity
@Table(name="bicycle")
public class Bicycle {
    @Id
    private String id;
    @JsonIgnore
    @ManyToOne(optional=true)
    @JoinColumn(name = "parking_id")
    private Parking parking;
    @JsonIgnore
    @OneToMany(targetEntity=Rental.class, mappedBy="bicycle",
            cascade=CascadeType.ALL, fetch = FetchType.LAZY)
    private List<Rental> rental;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public Parking getParking() {
        return parking;
    }
    public void setParking(Parking parkingSpace) {
        this.parking = parkingSpace;
    }
    public List<Rental> getRental() {
        return rental;
    }
    public void setRental(List<Rental> rental) {
        this.rental = rental;
    }
```

- ใช้หลักการ Dependency Inversion (DI) เช่น บรรทัดที่ 25 คือทำหน้าที่รับคลาส Parking เข้ามาและสร้าง object parking โดยในส่วนโค้ดตรงนี้ไม่ได้สร้างตัวแปรขึ้นใหม่แต่รับผ่าน Spine Container

- ใช้ Annotation เพื่อกำหนดความสัมพันธ์ระหว่าง Bicycle class กับ Parking class และ Rental class ตามลำดับ

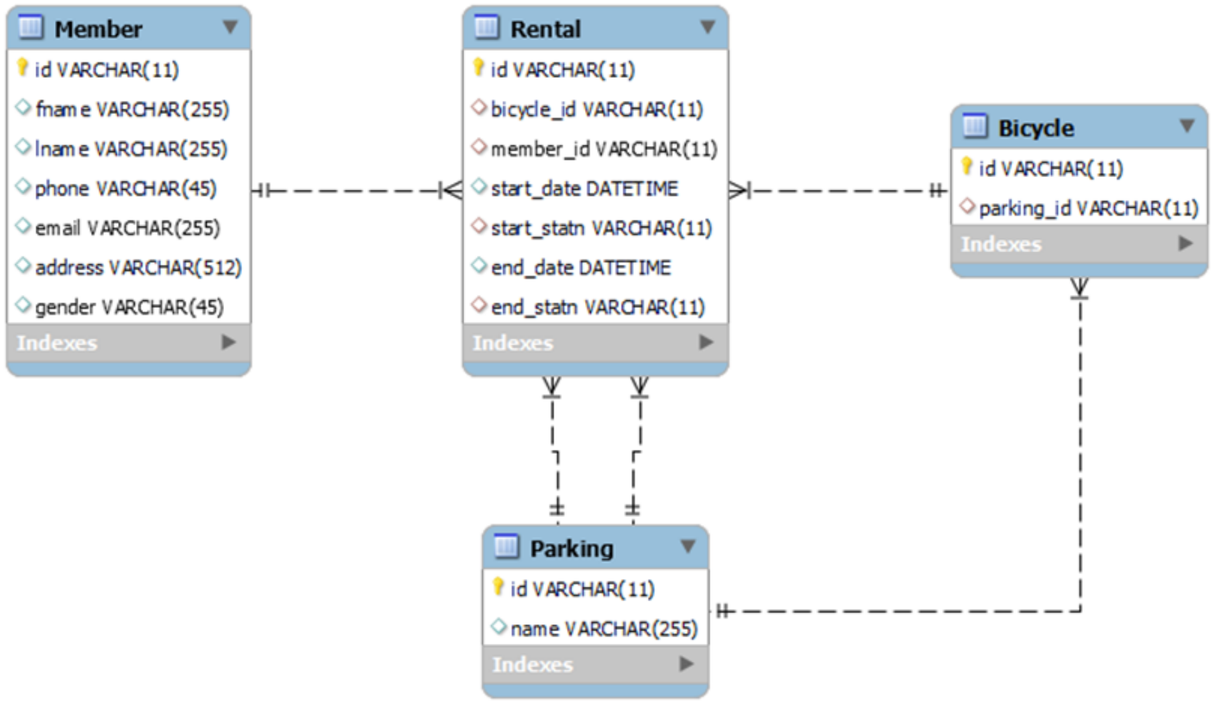
Class MainController

```java
@Controller
public class MainController {
    @Autowired
    private MemberService memService;
    @Autowired
    private BicycleService biService;
    @Autowired
    private RentalService renService;
    @Autowired
    private ParkingService parkService;

    @GetMapping("/")
    public String homeMenu(Model model) {
        return "home";
    }
    @GetMapping("/bi_edit")
    public String biEditMenu(Model model) {
        return "bicycleEdit";
    }
    @GetMapping("/bi_list")
    public String biListMenu(Model model) {
        //List<Bicycle> biList = biService.findAllBicycle();
        List<Bicycle> biList = biService.findAllAvailable();
        List<Parking> parkList = parkService.findAllParking();
        model.addAttribute(attributeName:"biList", biList);
        model.addAttribute(attributeName:"parkList", parkList);
        return "bicycleList";
    }
    @GetMapping("/bi_reg")
    public String biRegMenu(Model model) {
        AddBicycle ab = new AddBicycle();
        List<Parking> parkingList = parkService.findAllParking();
        model.addAttribute(attributeName:"bicycle", ab);
        model.addAttribute(attributeName:"parkingList", parkingList);
        return "bicycleRegister";
```

- ใช้ @Autowired เพื่อใช้ dependency ของ MemberService, BicycleService, RentalService และ ParkingService เข้ามาในคลาส MainController
- MainController class ไม่ได้สร้าง MemberService, BicycleService, RentalService, และ ParkingService แต่รับ dependency เหล่านั้นเข้ามาจาก Spring container
- MVC คลาส MainController เป็น @controller ทำหน้าที่รับ input จาก user และส่ง request ไปยัง service ต่างๆ เช่น MemberService, BicycleService, RentalService และ ParkingService ส่วน view จะทำหน้าที่แสดงผลข้อมูลที่ได้จาก controller

ฐานข้อมูลใน MYSQL (DATABASE)



- ตาราง Rental มีความสัมพันธ์แบบ Many to One กับ Member และ Bicycle ในส่วนของ Parking จะมีความสัมพันธ์ 2 แบบ ได้แก่ จุดเช่า (start_statn) และ จุดคืนรถ (end_statn)

- ตาราง Bicycle มีความสัมพันธ์แบบ Many to One กับ Parking คือรถหลายคันสามารถจอดในที่จอดเดี๋ยวกันได้

การสร้างระบบเว็บโดยใช้ SPRING BOOT EDITOR

```
57    <div class="container my-5 p-5">
58        <h4 class="text-center">Register</h4>
59        <!--save มาจากคลาส Repo นะ-->
60        <form action="#" class="col-md-4 offset-md-4" th:action="@{/add_bi}" method="post" th:object="${bicycle}">
61            <label for="name" class="form-label">ลานจอดรถ</label>
62            <select th:field="*{parking_id}">
63                <option value="NULL">เลือกลานจอด</option>
64                <option th:each="i : ${parkingList}" th:value="${i.id}" th:text="${i.name}">
65                </option>
66            </select>
67            <div class="mb-3">
68                <label for="name" class="form-label">จำนวน</label>
69                <input type="text" th:field="*{amount}" class="form-control" name="name">
70            </div>
71            <center><button type="submit" class="btn btn-primary">Submit</button></center>
72        </form>
73    </div>
```

- ในส่วนของ Model : โค้ด HTML จะใช้ object bicycle ซึ่งมาจาก controller ซึ่งเป็น model เพื่อแสดงข้อมูลจักรยานและรับ input จาก user โดยใช้ตัวแปร bicycle ดังนี้ th:object="${bicycle}"

- ในส่วนของ View : โค้ด HTML จะทำหน้าที่แสดงข้อมูลจักรยานและรับ input จาก user โดยใช้ thymeleaf template engine เช่น <html lang="en" xmlns:th="https://www.thymeleaf.com">

แผนการดำเนินงาน

ลำดับ	ขั้นตอนการดำเนินงาน	ระยะเวลาดำเนินงาน						
		กันยายน				ตุลาคม		
		1	2	3	4	1	2	3
1	กำหนดหัวข้อโปรเจค							
2	ศึกษาและรวบรวมข้อมูล							
3	ออกแบบระบบ							
4	จัดทำเว็บไซต์และฐานข้อมูล							
5	ทดสอบระบบ							
6	จัดทำรูปเล่มรายงาน							
7	นำเสนอ							

THANK YOU