

“메이플 마켓” 프로젝트 개요서

프로젝트 목표:

기존 중고거래 플랫폼인 당근마켓의 1:1 거래 특성의 한계점을 극복하기 위해서
사용자들이 직접 벼룩시장을 열고 참여하는 과정을 통해
다대다 거래를 할수있도록 하는것을 목표로함.

프로젝트 기능:

- 1.회원가입 및 로그인
- 2.주변 벼룩시장 탐색기능
- 3.벼룩시장 개설기능
- 3-1. 인근 몇백미터 내에 중복으로 벼룩시장을 개설하지 못하도록 하는 기능
- 4.벼룩시장내 중고물품 등록기능(사진 첨부 및 텍스트 작성가능)
- 5.타인이 등록한 물건에 대해 찜하기 기능
- 6.내가 등록한 물품을 누가 찜했는지 확인하는 기능.
- 7.내 물건을 찜한 상대방의 거래내역 확인 기능.
- 8.내 물건을 찜한 상대방중 누구와 거래할지 선택하는 기능

팀 소개:

조장: 김병관(본인)

e-mail : gbds234@gmail.com

역할: 백엔드,DB,배포

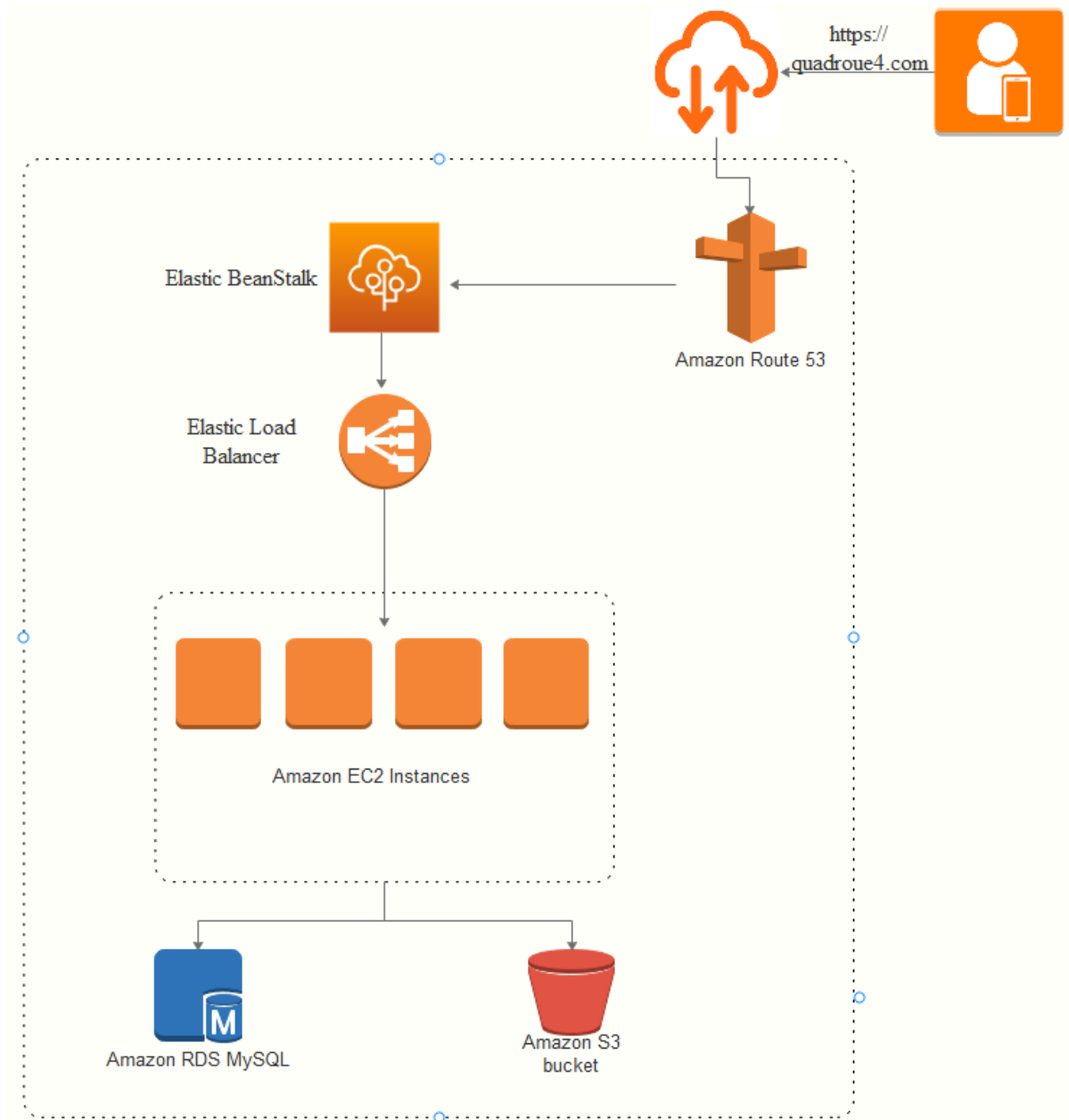
조원1: 강찬우

e-mail : wooju9812127@gmail.com

역할: 프론트엔드

시스템 구성:

(Spring boot 와 React.js 앱을 합쳐서 하나의 jar 파일로 만들고 aws elastic beanstalk에 업로드)



기능 상세 캡처

1.회원가입

아이디

woaju3434

비밀번호

.....

비밀번호 재확인

..a

8자 이상이어야 합니다

닉네임



9

1

2

3

4

5

6

7

8

9

0

q

w

e

r

t

y

u

i

o

p

a

s

d

f

g

h

j

k

l



z

x

c

v

b

n

m



비밀번호

.....

비밀번호 재확인

.....

닉네임

Fsg

생년월일

1998

3

12

성별

남

프로필 사진

FOR YOU

가입이 완료되었습니다!

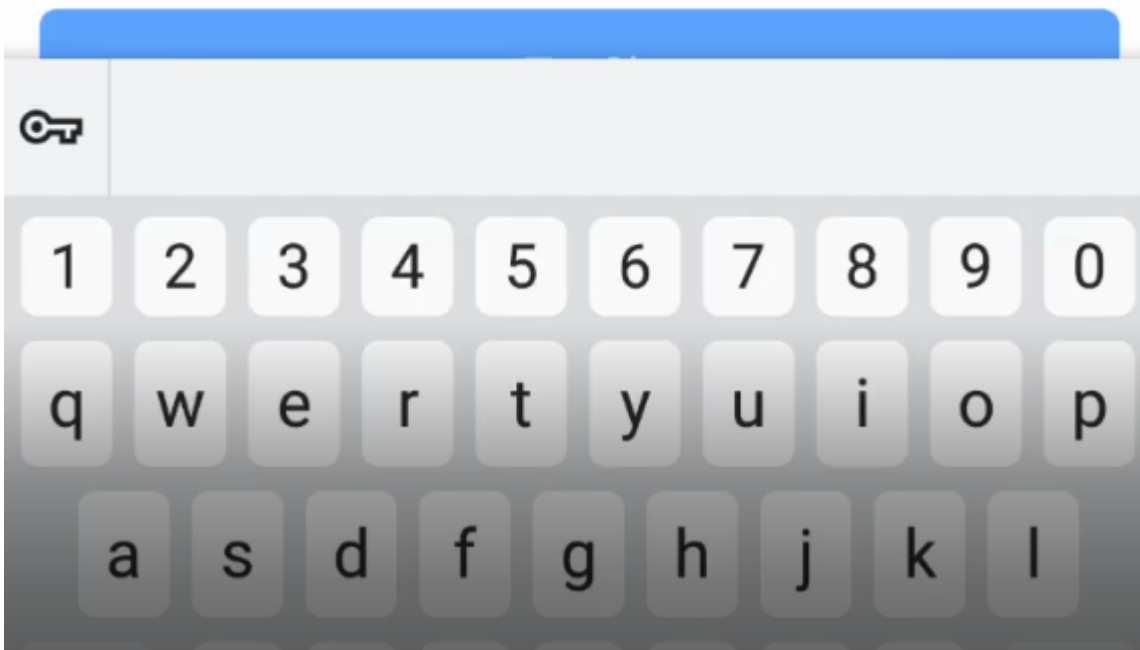
확인

Welcome to market

로그인

woaju3434

.....2|



2.장터 등록

장터 이름

판매자:장터와 물품등록

3. 물건등록



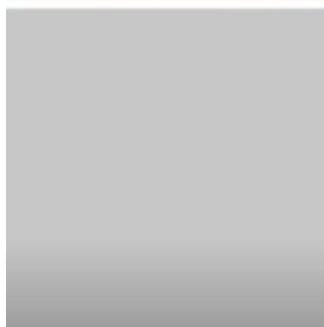
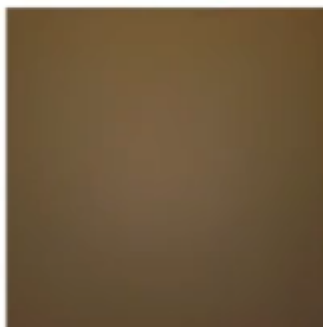
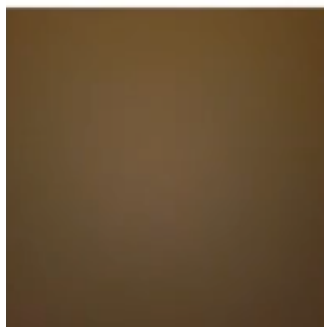
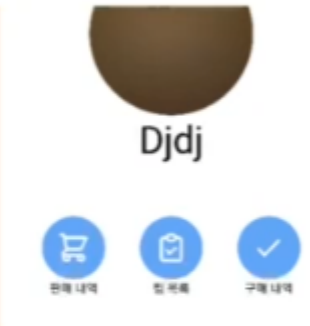
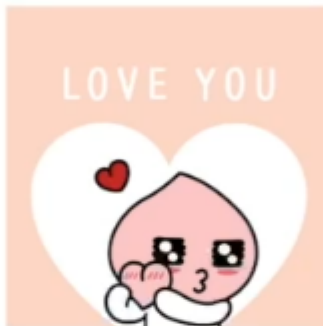
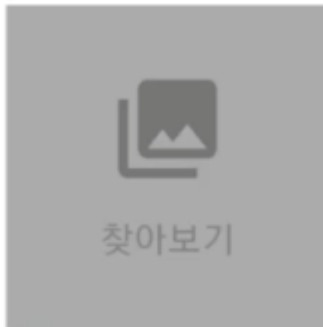
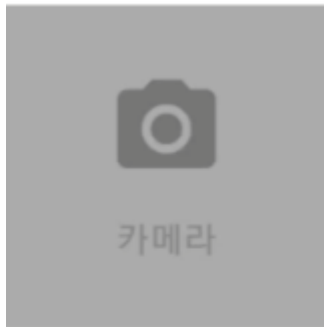
물건 이름

가격

판매 시간

10:00 ▼

설명





물건 이름

가격

판매 시간

설명

아이템등록이 완료되었습니다!

확인

물건 이름

연필깍이

가격

2000

판매 시간

10:00

설명

연필깍이



물품 등록

4.현재 내가 팔고있는 물품 리스트 확인 기능



5. 장터에 등록된 물품 확인 기능



 : 현재 위치
  : 다른 장터의 위치



10시



연필깎이

2000

14시



6.찜하기 (하트에 불이 들어오면 내가 찜한 물건임을 확인가능)




market



Chanw12

판매자

연필깎이

2000원 

연필깎이

구매자:찜하기 및 찜목록 확인




market



Chanw12

판매자

연필깎이

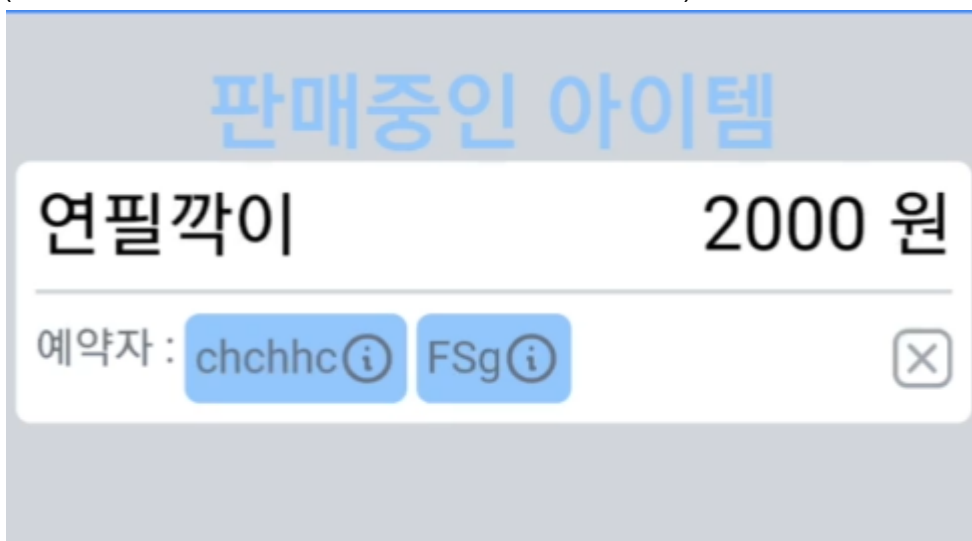
2000원 

연필깎이

7.내가 찜한 물건 리스트 확인 기능



8. 내 물건을 찜한 사람들을 확인하는 기능
(예약자중 한명을 클릭하면 거래상대방으로 선택됨.)



9. 내가 찜한 물건들중 거래자로 확인됐는지 확인하는 기능
(초록색 하트 및 체크 표시)



10. 거래당일 벼룩시장에서 자신의 착장정보를 입력하는 기능(서로가 알아볼수 있도록)



11.위치 기반으로 실제 거래일자에 장터에 왔는지 확인하는 기능.



market

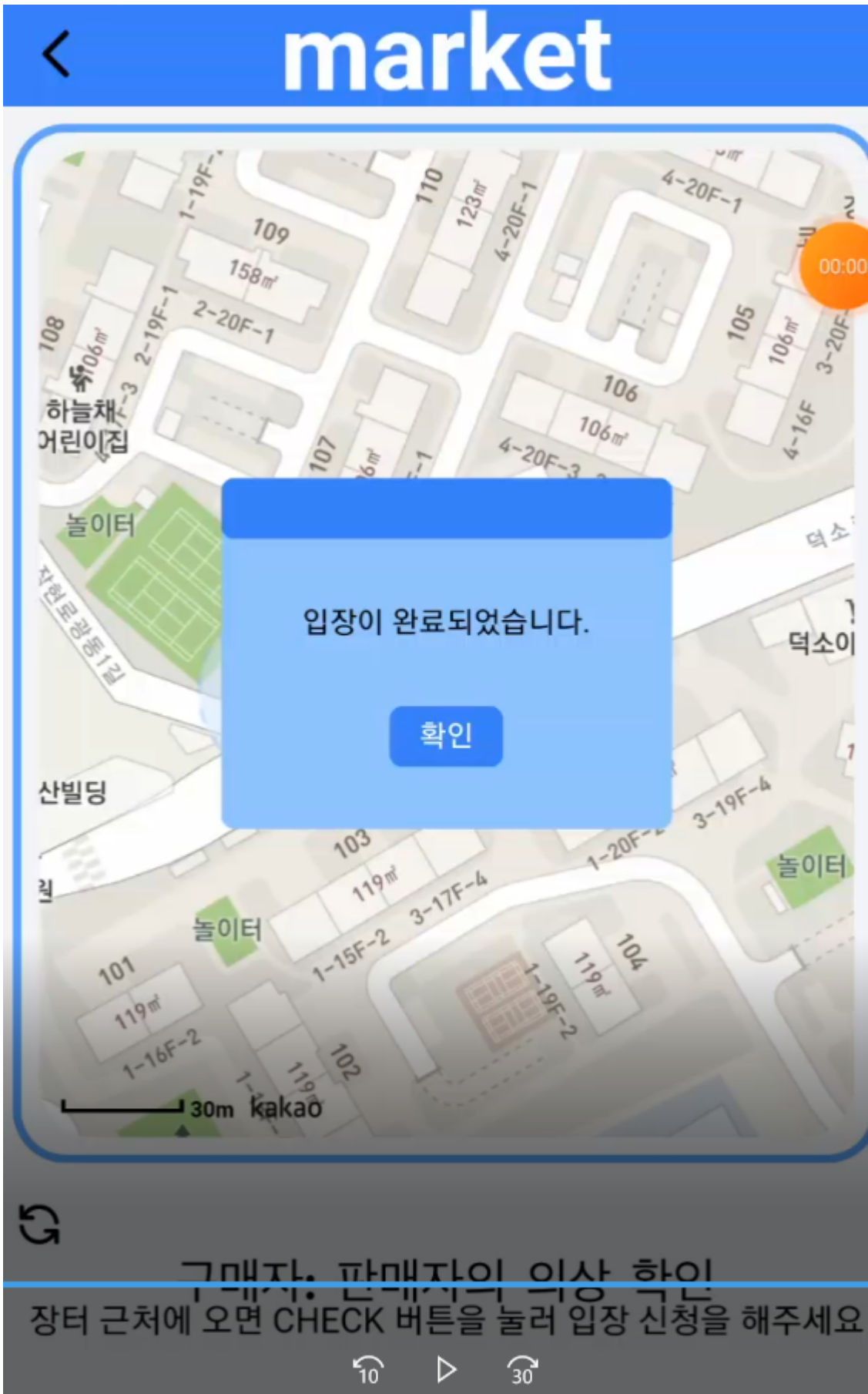


장터 근처에 오면 CHECK 버튼을 눌러 입장 신청을 해주세요

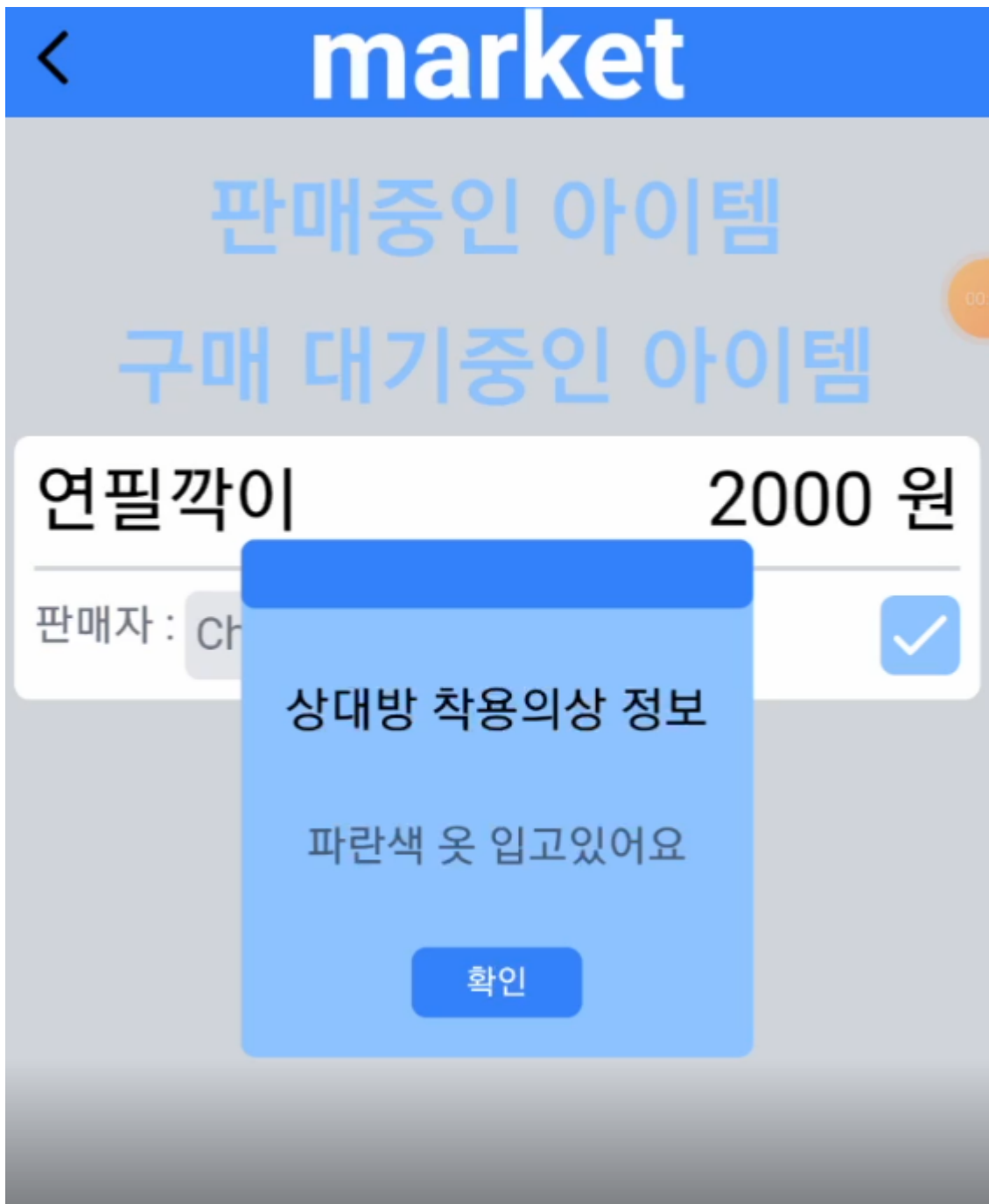
CHECK



장터 근처에 오면 CHECK 버튼을 눌러 입장 신청을 해주세요



12. 거래 물건 대상의 착장정보를 확인하는 기능



13. 거래내역이 업데이트되는 기능



소스코드 예시
(모든 코드는 따로
첨부하겠습니다.)

item 엔티티

```

@Getter
@Setter
@Entity
public class Item {

    @Id @GeneratedValue
    @Column(name="ITEM_ID")
    private Long itemId;
    private String itemName;
    private Long price;
    private String description;
    private boolean soldOut = false;
    private boolean bought = false;
    private int sellingTime = 10;
    private String imagePath;

    @ManyToOne
    @JoinColumn(name="Owner_ID")
    private Member owner;

    @ManyToOne
    @JoinColumn(name="ConfirmedMember_ID")
    private Member confirmedMember;

    @ManyToOne
    @JoinColumn(name="MARKET_ID")
    private Market market;

    @OneToMany(mappedBy = "reserveItems", cascade = CascadeType.REMOVE)
    private List<ITEM_MEMBER_RESERVE_RELATION> reserveMembers;
}

```

member 엔티티 코드 예시

```

@Getter
@Setter
@Entity
public class Member {

    @Id @GeneratedValue
    @Column(name = "MEMBER_ID")
    private Long memberId;

    @OneToMany(mappedBy = "owner")
    protected List<Item> myItems = new ArrayList<>();

    @OneToMany(mappedBy = "confirmedMember")
    private List<Item> confirmedItems = new ArrayList<>();

    @OneToMany(mappedBy = "reserveMember")
    private List<ITEM_MEMBER_RESERVE_RELATION> reserveItems;

    @OneToMany(mappedBy = "boughtMember")
    private List<ItemBought> boughtItems;

    @OneToMany(mappedBy = "soldMember")
    private List<ItemSoldout> soldoutItems;

    @NotEmpty
    private String loginId; //로그인 ID
    @NotEmpty
    private String name; //사용자 이름
    @NotEmpty
    private String password;

    private String fasion;

    private String imagePath;

    @ColumnDefault("0")
    private Long reportedCount;

    private Long reservedMarket;
}

```


멤버는 여러개의 아이템을 찜할수 있으며,
아이템은 여러 사람이 찜할수 있기 때문에 다대다 관계를 가지므로
멤버와 아이템 사이의 엔티티를 하나 만들었음.

```
@Getter
@Setter
@Entity
public class ITEM_MEMBER_RESERVE_RELATION {

    @Id
    @GeneratedValue
    private Long id;

    @ManyToOne
    @JoinColumn(name="RESERVED_ITEM")
    private Item reserveItems;

    @ManyToOne
    @JoinColumn(name="RESERVED_MEMBER")
    private Member reserveMember ;
}
```

내가 구매한 아이템 내역 엔티티

```
@Entity
@Getter
@Setter
public class ItemBought {

    @Id
    @GeneratedValue
    private Long id;

    @ManyToOne
    @JoinColumn(name="member")
    private Member BoughtMember;

    private String itemName;
    private Long itemPrice;
    private String imagePath;
}
```

내가 판매한 아이템 내역 엔티티

```

@Entity
@Getter
@Setter
public class ItemSoldout {

    @Id
    @GeneratedValue
    private Long id;

    @ManyToOne
    @JoinColumn(name="SoldMember")
    private Member SoldMember;

    private String itemName;
    private Long itemPrice;
    private String imagePath;
}

```

벼룩시장 엔티티(벼룩시장내 아이템 엔티티들은 벼룩시장의 id를 fk로 갖고있음)

```

@Getter
@Setter
@Entity
public class Market {

    @Id @GeneratedValue
    @Column(name = "MARKET_ID")
    private Long marketId;

    @OneToMany(mappedBy = "market")
    private List<Item> items = new ArrayList<>();

    private double latitude;
    private double longitude;
    //public Market(String[] itemsInfo){
    //    items = itemsInfo;
    //}
}

```

거래일자에 벼룩시장 입장한 기록 엔티티
(미완성이지만 미참여 인원에 대한 신고기능을 위함)

```

@Entity
@Getter
@Setter
public class MarketEnter {
    @Id
    @GeneratedValue
    private Long id;

    private Long memberId;

    private Long marketId;
}

```

신고기록 엔티티

(미완성이지만 장터가 끝난 이후 위의 장터입장 기록들을 통해 검증하고 사용자에게 페널티를 부과할 계획이었음.)

```

@Entity
@Getter
@Setter
public class Report {
    @Id
    @GeneratedValue
    private Long id;

    private Long reportMember;

    private Long reportedMember;

    private Long marketId;
}

```

데이터 jpa를 이용한 db 접근 객체 memberRepository
(다른 repository 객체들도 이와 동일)

```

@Transactional
public interface MemberRepository {
    Member save(Member member);
    Optional<Member> findById(Long id);
    //Optional<Member> findByName(String name);
    List<Member> findAll();
    Optional<Member> findByLoginId(String loginId);

    void reportHandle();
}

```

내가 판매하는 물건들을 조회하는 MemberService.java 코드 예시

```

public List<ItemDTO> getItems(HttpServletRequest request){ //현재 내가 내놓은 물건 내역
    HttpSession session = request.getSession(false);
    List<ItemDTO> itemDTOS = new ArrayList<>();
    if(session == null){
        return itemDTOS;
    }
    Member loggedMember = (Member) session.getAttribute(SessionConst.LOGIN_MEMBER);

    if(loggedMember == null){
        return itemDTOS;
    }
    Member realMember = memberRepository.findById(loggedMember.getMemberId()).get();
    List<Item> items = realMember.getMyItems();
    items.stream().forEach(item -> itemDTOS.add(
        new ItemDTO(item, awsS3Service.downloadFile(item.getImagePath()), fasionDescription: null)));
    return itemDTOS;
}

```

벼룩시장에 물건을 등록하는 MarketService.java 코드 예시

```

public String saveItem(Long marketId, String itemName, Long price, int sellingTime, MultipartFile photo, String description, HttpServletRequest request) throws IOException {
    HttpSession session = request.getSession(false);
    if(session == null){
        return "Not Logged In";
    }
    Member loggedMember = (Member) session.getAttribute(SessionConst.LOGIN_MEMBER);
    if(loggedMember == null){
        return "Not Logged In";
    }

    Item item = new Item();
    Market market = marketRepository.findById(marketId).get();
    item.setDescription(description);
    item.setItemName(itemName);
    item.setPrice(price);
    item.setOwner(loggedMember);
    item.setMarket(market);
    item.setSellingTime(sellingTime);

    String fileName = awsS3Service.uploadFile(photo);
    item.setImagePath(fileName);
    itemRepository.save(item);

    return "OK";
}

```

memberController.java 예시

```

@GetMapping("/member/items") //현재 내가 내놓은 물건 내역
public List<ItemDTO> getSellingItems(HttpServletRequest request) { return memberService.getItems(request); }

@GetMapping("/member/reserveitems") // 현재 내가 찜한 물건 내역
public List<ItemDTO> getReserveItems(HttpServletRequest request) { return memberService.getReserveItems(request); }

@GetMapping("/member/mySellingReservedItems") //현재 내가 판매하는 물건중 거래예약된 내역
public List<ItemDTO> mySellingReservedItems(HttpServletRequest request){
    return memberService.getSellingItems(request);
}

@GetMapping("/member/myBuyingReservedItems") //현재 내가 찜한 물건중 거래예약된 물건 내역
public List<ItemDTO> myBuyingReservedItems(HttpServletRequest request){
    return memberService.getBuyingItems(request);
}

@GetMapping("/member/mySoldout") //내 판매 내역
public List<ItemSoldBoughtDTO> mySoldoutItems(@RequestParam Long memberId, HttpServletRequest request){
    return memberService.getMySoldouts(memberId, request);
}

@GetMapping("/member/myBought") //내 구매 내역
public List<ItemSoldBoughtDTO> myBoughtItems(@RequestParam Long memberId, HttpServletRequest request){
    return memberService.getMyBoughts(memberId, request);
}

@GetMapping("/member/myprofile")
public MemberDTO getProfile(HttpServletRequest request) { return memberService.myProfile(request); }

@GetMapping("/member/profile")
public MemberDTO getProfile(@RequestParam Long memberId, HttpServletRequest request){
    return memberService.memberProfile(memberId, request);
}

```