

빈티지 서점 3차 최종 발표

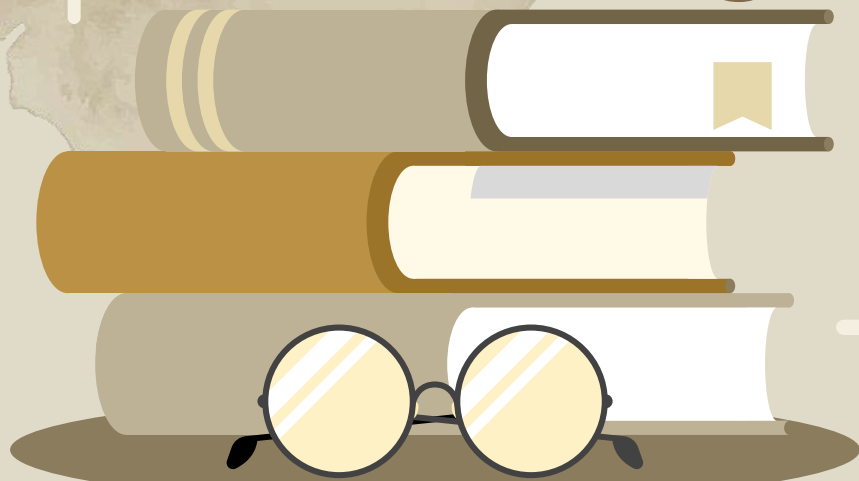
팀장 : 김동욱 2017112090

팀원 : 김동현 2016112103

김민석 2017211810

안형진 2016112130

정환훈 2017112112



목차

01. 팀원 소개

02. 회의록 (backlog)

03. 시나리오

04. UML 다이어그램

05. DB 설계

06. 내용 비교& 소스코드

07. 웹페이지 시연 화면

01

팀원 소개



팀 소개



빈티지서점

팀장 : 김동욱 2017112090

팀원 : 김동현 2016112103

김민석 2017211810

안형진 2016112130

정환훈 2017112112

02

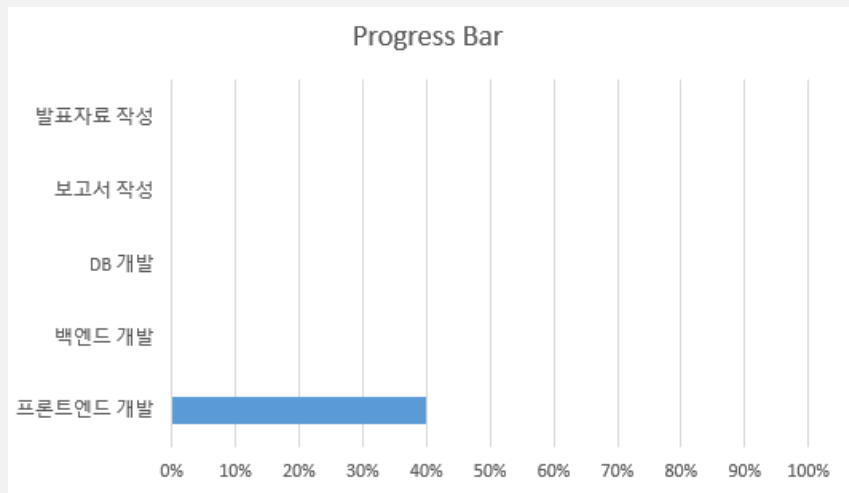
회의록 (backlog)



회의록 (backlog)

11월 9일

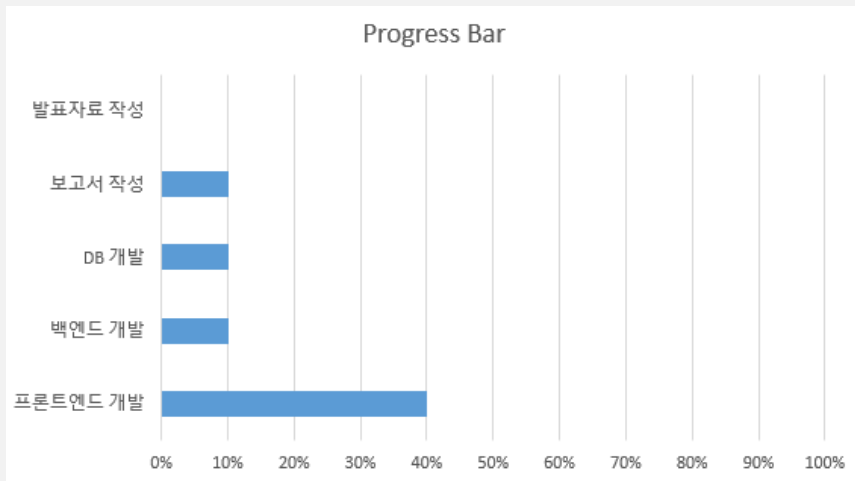
- 2차 발표에 대한 피드백을 공유
- Kotlin 프레임워크 조사
- 각 페이지 별로 HTML+CSS를 통해 UI 틀 잡기



회의록 (backlog)

11월 13일

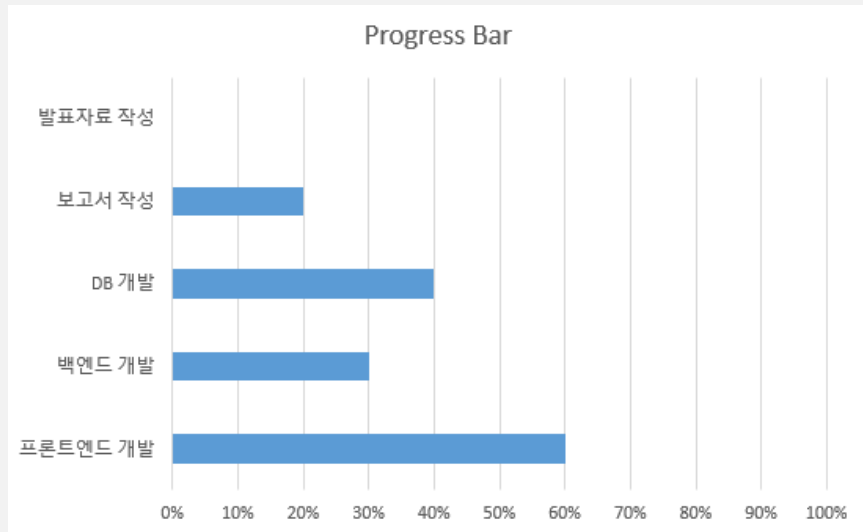
- Kotlin+Spring으로 프레임워크 결정
- 사용할 DB를 MY SQL로 결정
- 메인페이지 기능 구현 계획
- CSS 수정 및 보완점 계획



회의록 (backlog)

11월 16일

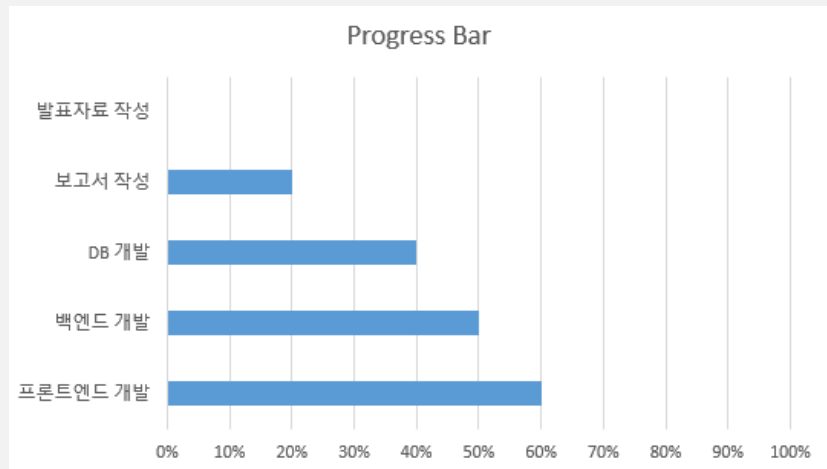
- DB에 필요한 테이블 조사 후 스키마 생성
- 메인페이지 기능 구현 별 어려웠던 점 공유
- CSS 통일 및 프론트엔드와 백엔드 연결



회의록 (backlog)

11월 20일

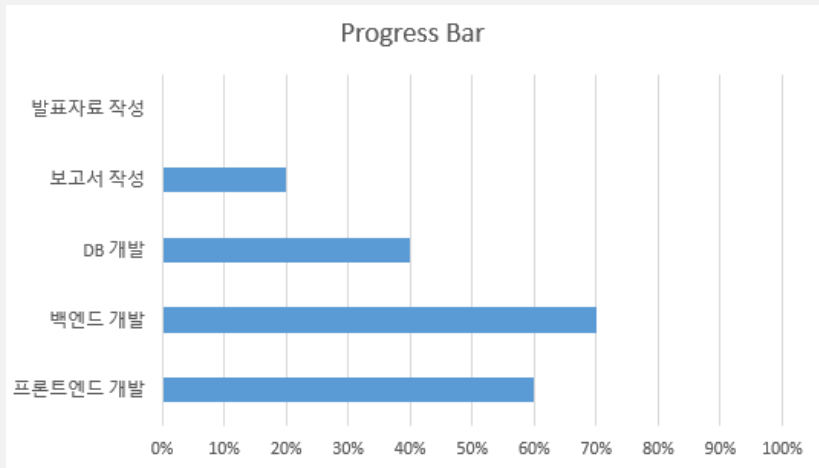
- 구현 완료된 메인페이지 기능
종합 및 보완점 확인(로그인,
회원가입 기능)
- 구현 못한 메인페이지 기능
해결안 회의(검색기능)
- 상세 페이지, 장바구니 페이지,
책 등록 페이지, 마이페이지
구현 계획



회의록 (backlog)

11월 23일

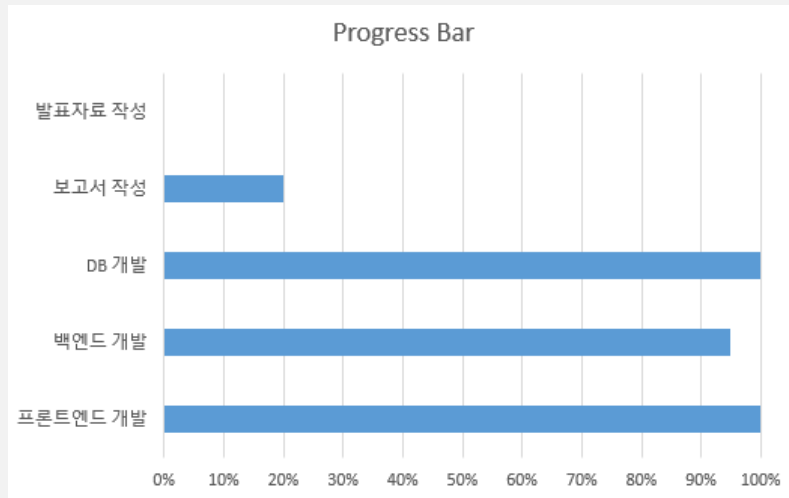
- 메인페이지, 책등록 페이지, 마이페이지 기능 구현 완료
구현 기능 테스트 및 보완점 확인
- 상세페이지와 장바구니
페이지 구현간 어려웠던 점
공유 및 해결안 회의
- 마이페이지 구현 계획



회의록 (backlog)

11월 27일

- 상세페이지, 장바구니 페이지 구현 완료
- 모든 페이지 기능 점검 및 확인
- 발표자 선정, 발표자료 각각 역할 분담하여 발표자료 작성



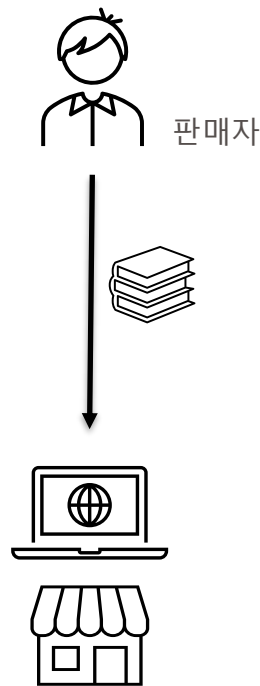
03

시나리오



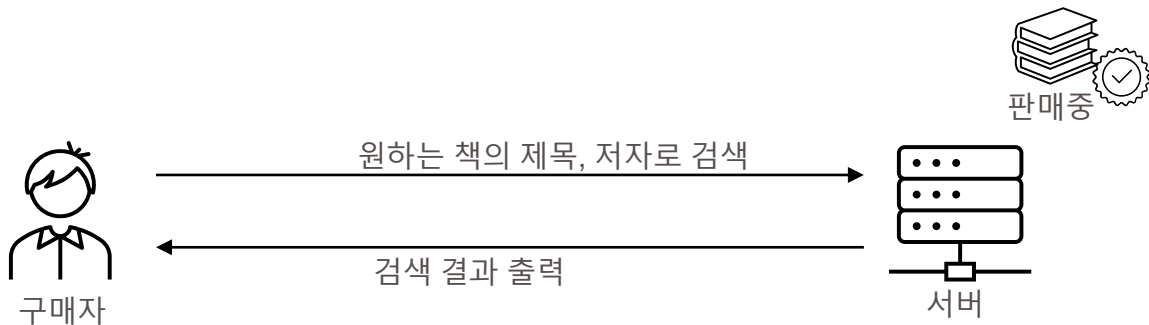
시나리오

1. 빈티지 서점 사이트는 인터넷을 통해 중고 서적 등록 및 판매가 가능한 시스템으로 구축되어 있다.
2. 사이트를 이용하기 위해 회원 가입을 하고 로그인을 한다.
3. 중고 서적 판매 시스템에는 중고 서적의 등록과 서적의 검색 기능이 있다.
4. 중고 서적 등록의 경우, 판매자는 서적의 이름, 분류, 품질, 희망 가격과 입금 될 계좌를 입력하고 고객은 등록할 서적을 중고 서점에 발송한다.



시나리오

5. 발송된 서적을 확인하여 중고 서점 관리자가 서적의 사진을 등록한다. 판매자가 등록을 취소한 경우 판매자에게 책을 다시 발송해준다.
6. 중고 서적 구매 기능은 현재 구매 가능한(DB에 있는) 중고 서적을 살펴볼 수 있는 웹페이지로 이동한다. 검색 페이지에서는 책 이름, 저자로 검색할 수 있다. 검색 시 같은 책이더라도 등록된 책이 모두 나열된다.



시나리오

7. 검색 페이지에서 책을 검색하고 장바구니에 담을 수 있고, 장바구니 페이지에서 현재 담겨진 책 목록과 재고 상태를 확인할 수 있다.

8. 구매 버튼을 누르면 나오는 팝업창에 구매자의 주소를 입력한다. 입력 확인 버튼을 누르면 회원의 주소가 서버로 전송되고 판매자 계좌가 마이페이지 구매 목록에 표시된다. 구매자는 판매자의 계좌로 송금한다.

판매자



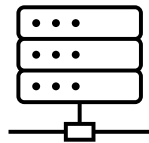
구매자



송금

판매자 계좌

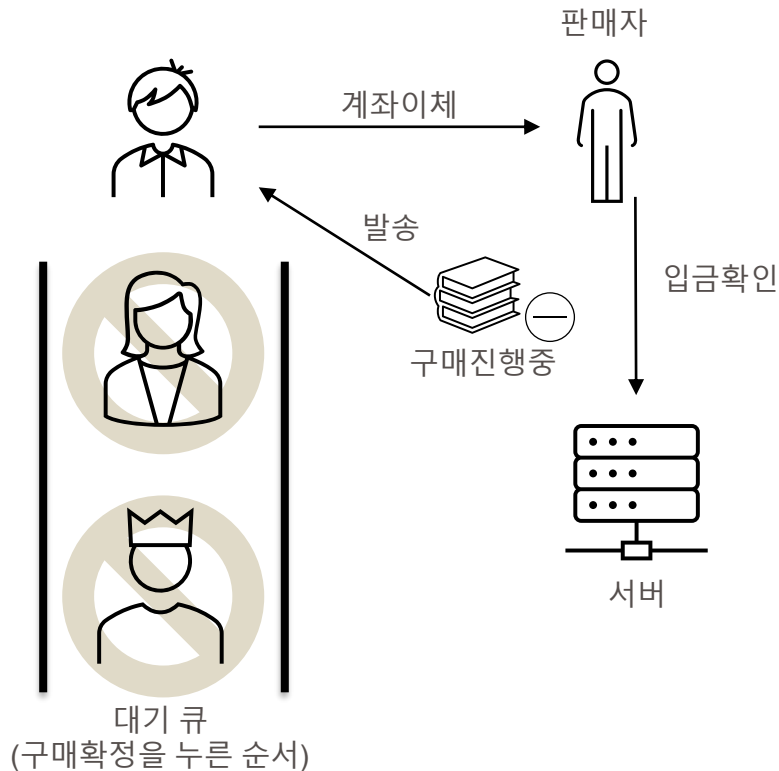
구매자 주소



서버

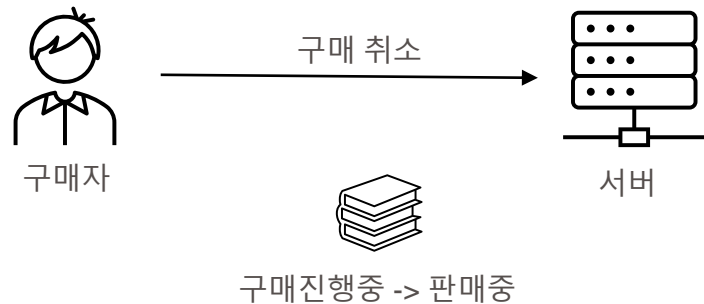
시나리오

9. 판매자 측에서 입금확인 버튼을 누르면 등록 된 책이 구매자에게 발송된다. 발송이 완료되면 재고 상태를 판매완료로 바꾼다.
10. 구매는 구매확정을 먼저 누른 사람이 우선적으로 선택된다. 책의 재고상태를 구매진행중으로 바꾸어 다른 사람이 구매하지 못하게 일시적으로 막는다



시나리오

11. 만약 구매자의 마음이 바뀌어 구매취소 버튼을 누르면, 도서는 다시 판매 중 상태로 바뀐다. 판매 중 상태일 때는 누구든지 책을 구매할 수 있는 상태이다.
12. 고객은 마이페이지에서 자신이 구매하거나 등록한 서적의 현황을 확인할 수 있다.



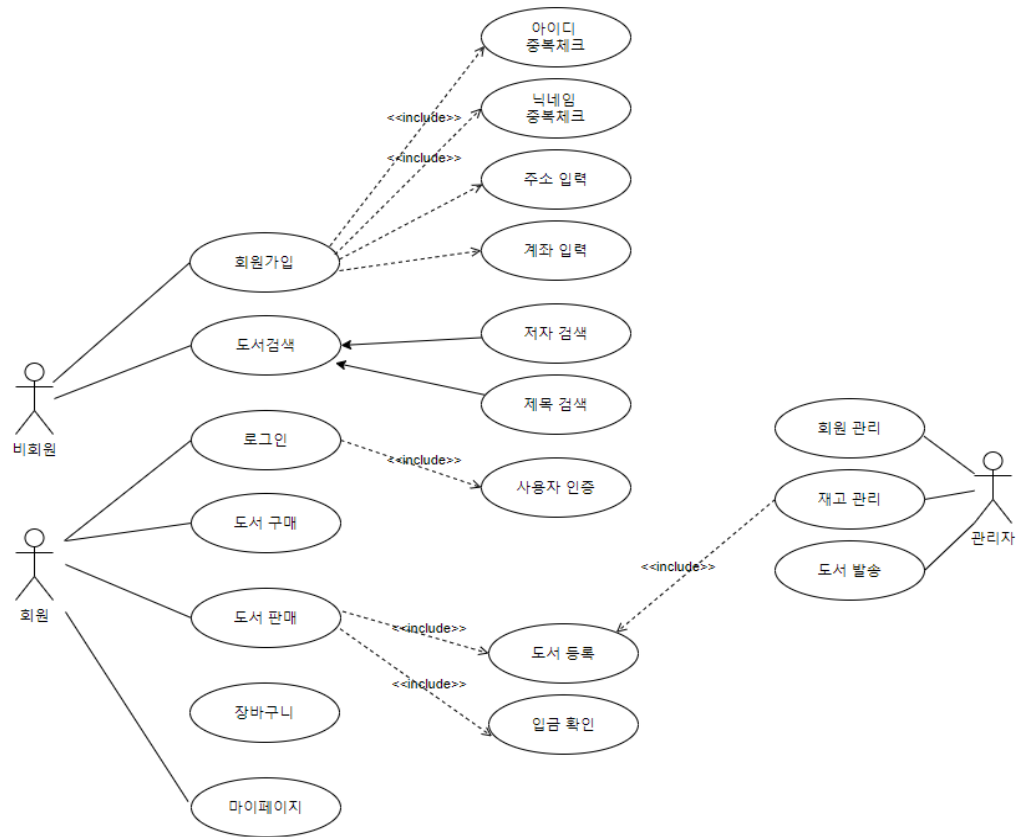
04

UML 다이어그램

1. 유스케이스 다이어그램
2. 액티비티 다이어그램
3. 시퀀스 다이어그램



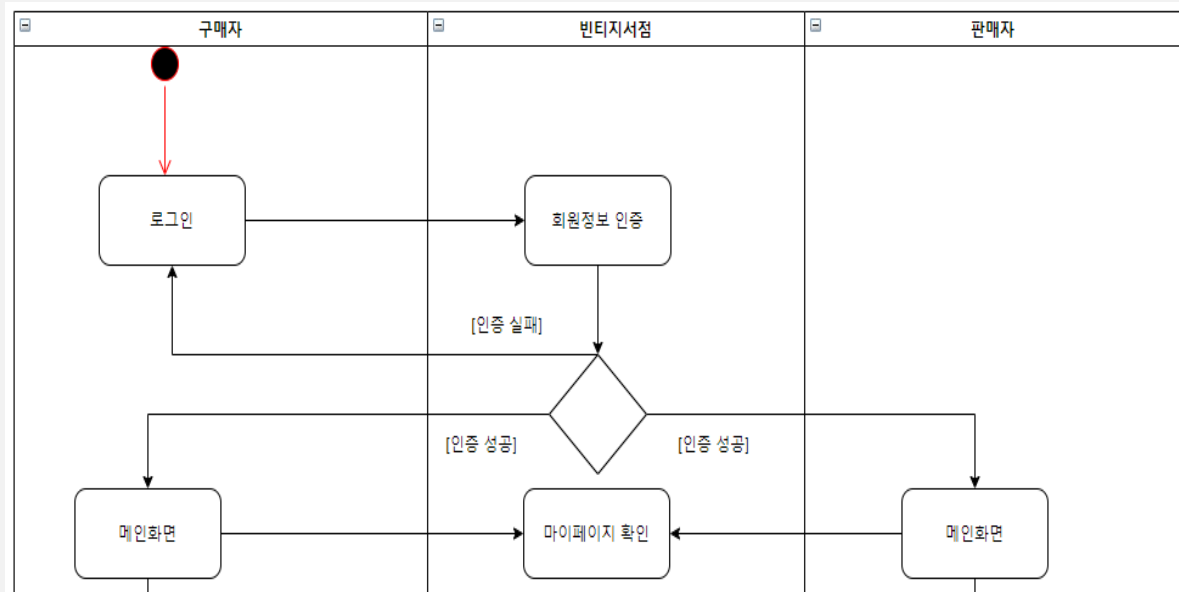
유스케이스 다이어그램



액티비티 다이어그램

- 로그인

- 마이페이지 확인

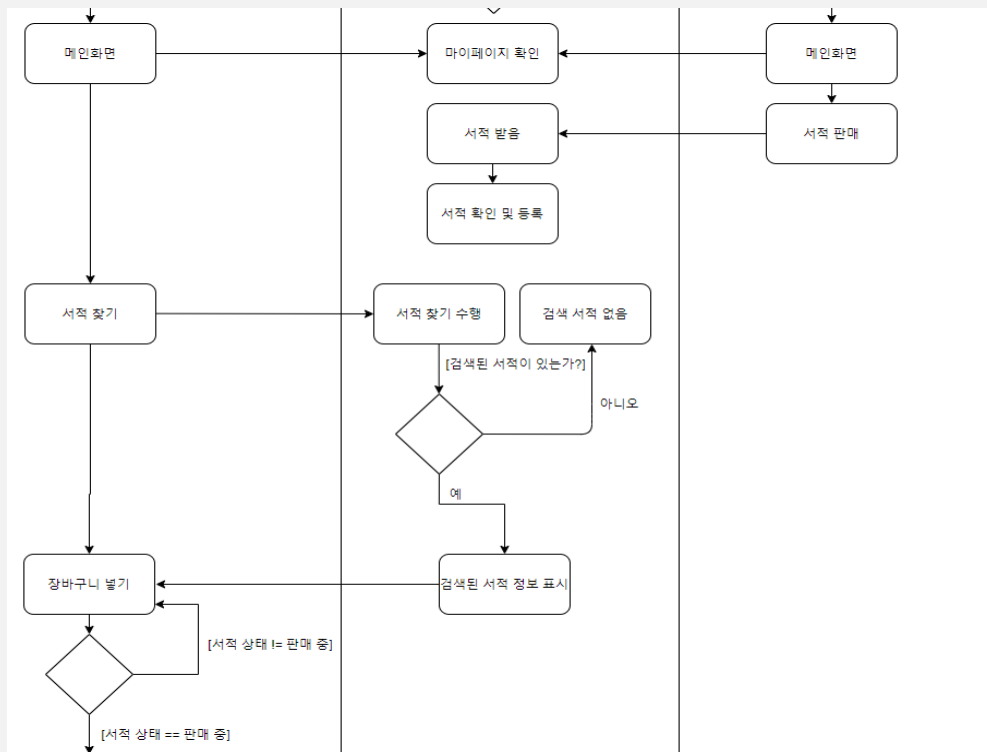


액티비티 다이어그램

- 서적 판매

- 서적 검색

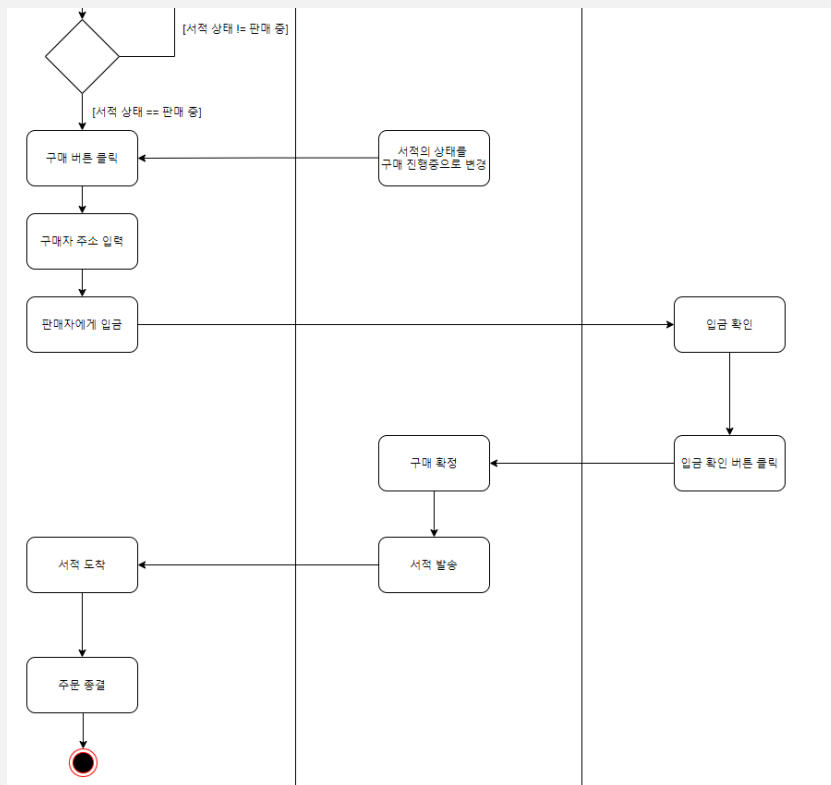
- 장바구니



액티비티 다이어그램

- 서적 구매

- 구매 확정 및 발송



시퀀스 다이어그램

도서 판매자 입장

1. 시스템 로그인

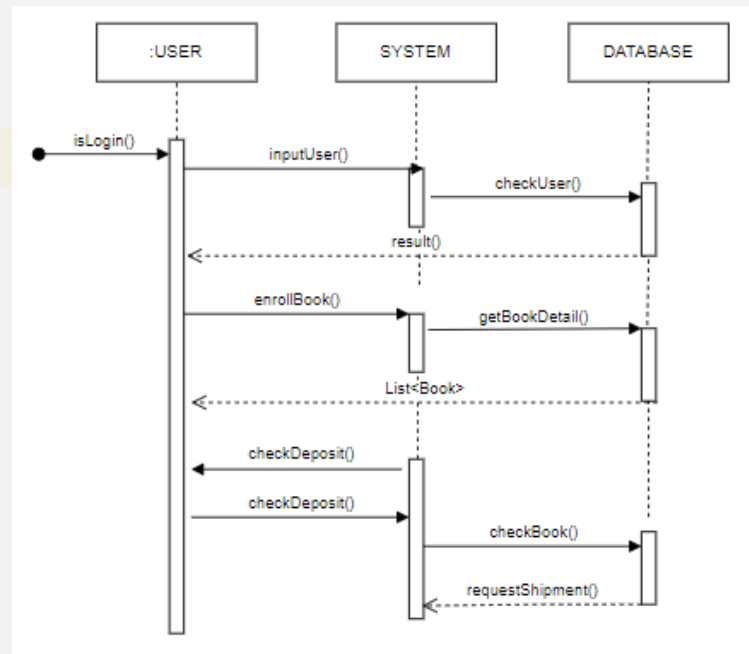
유저가 ID/PW입력 시 시스템이 입력정보를 받아 DB로 넘김. DB는 입력정보를 토대로 결과 출력

2. 도서 등록

도서 등록시 이름, 분류, 품질, 희망가격을 입력하면 시스템에서 정보를 받아 DB에 입력 후 DB는 등록 결과 출력

3. 입금 확인 요청

입금 확인 요청이 뜰 시 판매자는 입금 확인 버튼을 누른 후 시스템은 발송할 책을 DB에서 확인한 후 시스템 상에서 발송 요청이 뜨게 함.



시퀀스 다이어그램

도서 구매자 입장

1. 시스템 로그인

2. 도서 검색

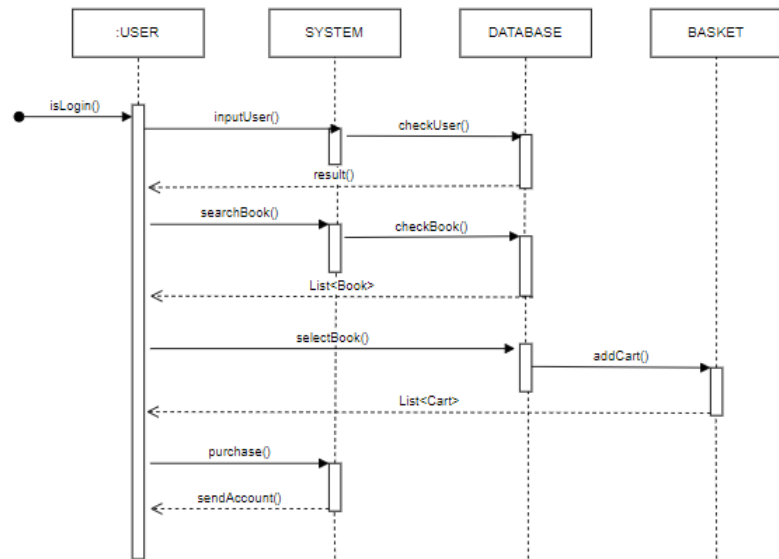
구매자는 도서를 검색할 시 시스템에서 검색 정보를 가져와 DB에서 해당 도서를 조회함. DB는 조회한 도서들의 리스트를 제공

3. 도서 선택

구매하려는 도서를 선택 시 DB에서 해당 도서 정보를 장바구니로 보내서 추가함. 장바구니는 지금까지 등록된 도서의 내역을 구매자에게 제공

4. 구매 확정

장바구니에 있는 도서를 구매하고자 할 시 구매 확정 버튼을 누르면 시스템에서 판매자의 계좌 정보를 전송

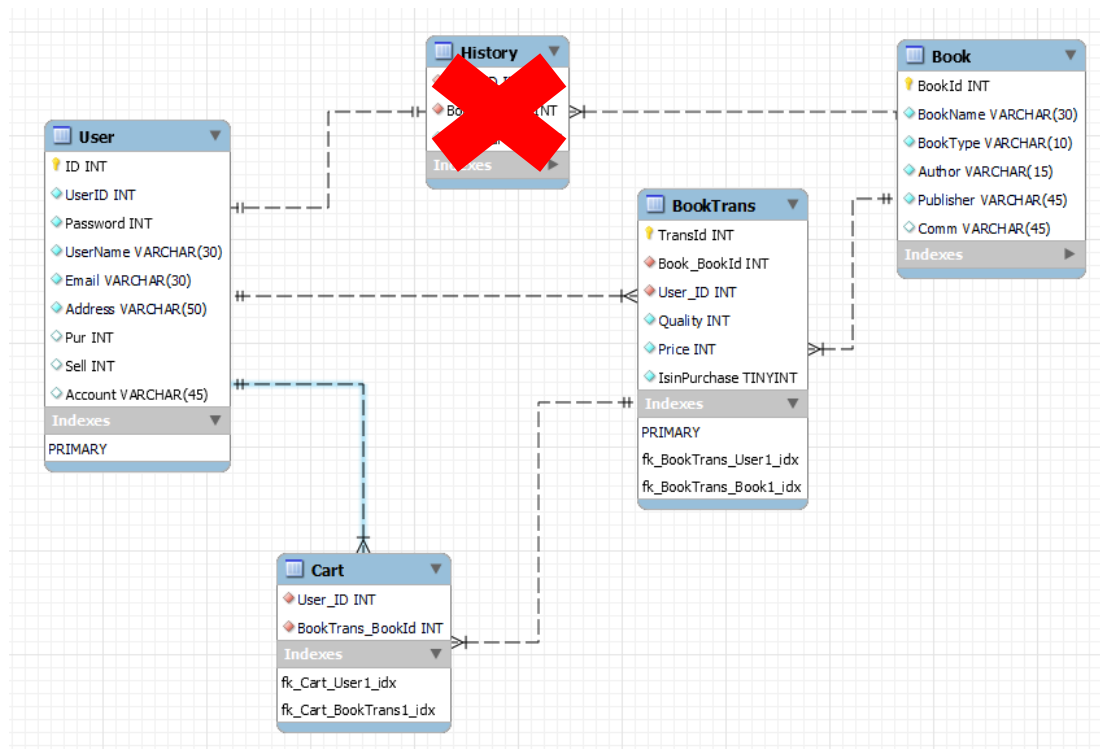


05

DB 설계



DB 스키마 구조



주요 테이블 코드 내용

```
1 package com.vintagelibrary.backend.domain.entity
2
3 import lombok.Getter
4 import lombok.NoArgsConstructor
5 import lombok.Setter
6 import javax.persistence.*
7
8 @Entity
9 @Table(name = "book")
10 @Getter
11 @Setter
12 @NoArgsConstructor
13 class Book(bookName : String, author : String,
14           publisher : String, quality : String,
15           bookType : String, price : String,
16           comm : String, imageName : String)
17 /* price : String,*/
18 @Id
19 @GeneratedValue(strategy= GenerationType.IDENTITY)
20 var bookId:Long?=null // 각 객체마다 고유한 id
21 var bookName = bookName
22 var bookType = bookType
23 var author = author
24 var publisher = publisher
25 var comm = comm
26 var quality = quality
27 var price = price
28 var imageName = imageName
29
```

<Book>

```
1 package com.vintagelibrary.backend.domain.entity
2
3 import lombok.*
4 import javax.persistence.*
5
6 @Entity
7 @Table(name = "user")
8 @Getter
9 @Setter
10 @NoArgsConstructor
11 class User(name : String, userId : String, password: String, email : String,
12           address: String, account: String)
13 @Id
14 @GeneratedValue(strategy=GenerationType.IDENTITY)
15 var id:Long?=null // 각 객체마다 고유한 id
16 var name:String = name // 유저 실명
17 var userId:String = userId // 유저가 지정한 id(==nickname)
18 var password:String = password
19 var email:String = email
20 var address:String = address
21 var account:String = account // 계좌 입력
22
```

<User>

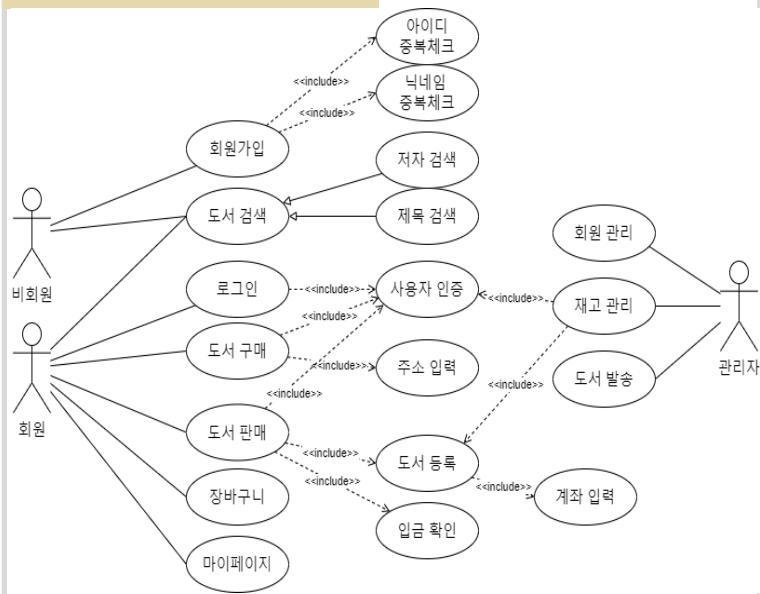
06

내용 비교 & 소스 코드



유스케이스 다이어그램

2차 발표

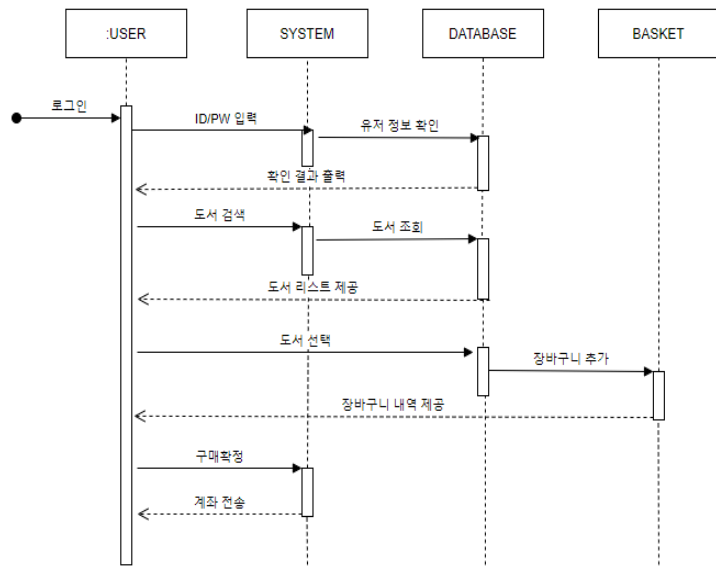


최종안

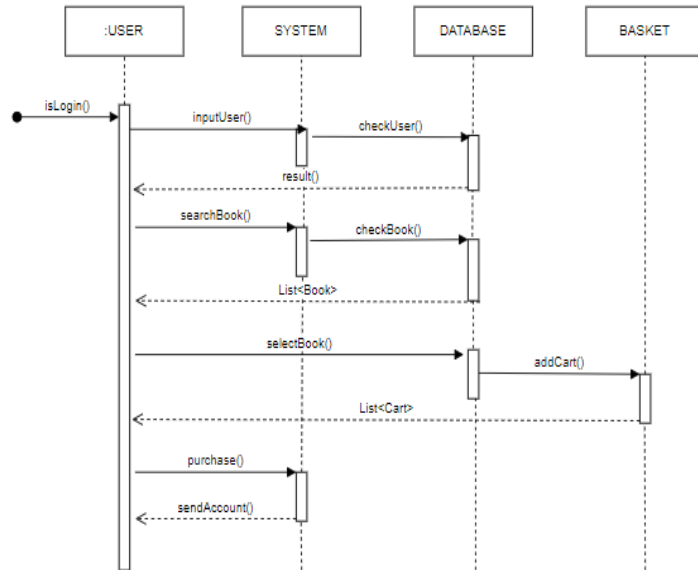


시퀀스 다이어그램(구매자)

2차 발표

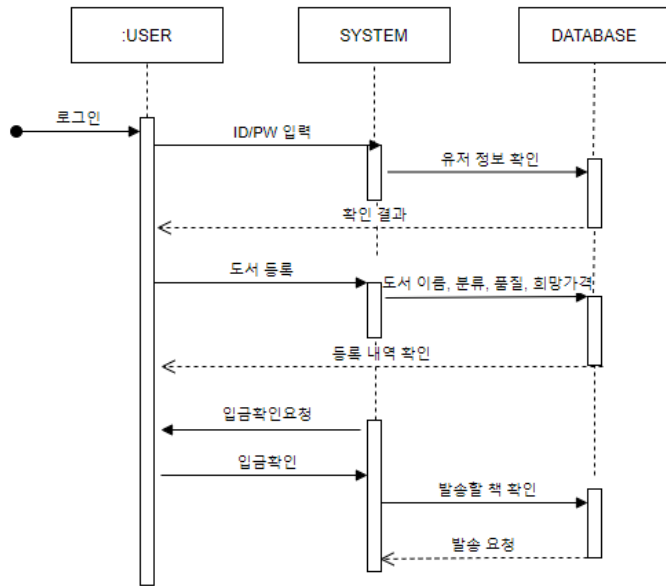


최종안

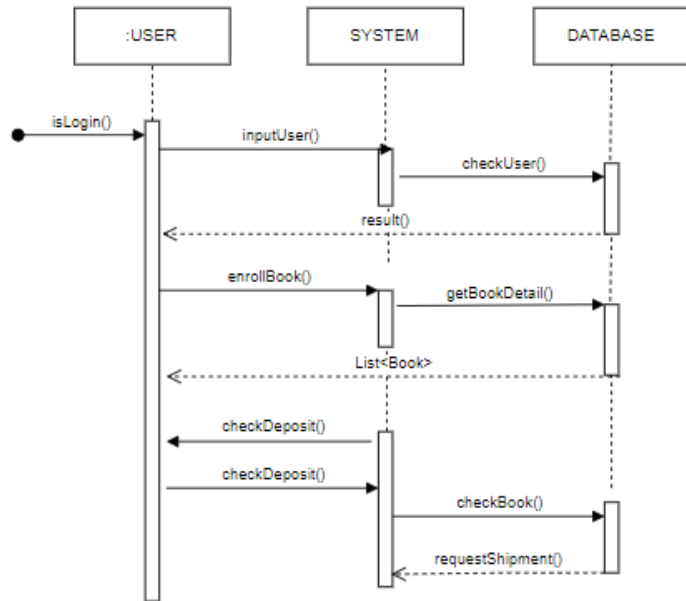


시퀀스 다이어그램(판매자)

2차 발표



최종안



소스 코드

1. 회원가입

회원가입에 필요한 정보를 입력하고 DB에 저장된다.
주소, 계좌번호도 입력하며 기존회원,아이디 중복체크
기능도 수행한다.

```
fun signup(req: HttpServletRequest) : String{
    try{
        // name id pw email address
        val name = req.getParameter( name: "name")
        val id = req.getParameter( name: "id")
        val pw = req.getParameter( name: "pw")
        val email = req.getParameter( name: "email")
        val address = req.getParameter( name: "address")
        val account = req.getParameter( name: "account")

        println(id);

        val overlap = userService.findByUserId(id) // 이미 해당 id 가진 사람이 존재할 시

        if(overlap != null){ // 재시도 요청
            return "<script>" + "alert(\"이미 있는 아이디입니다.\");" + "location.href=\"signup\";" + "</script>";
        }

        // 없을 시 입력받은 정보로 db에 새로운 User 저장
        userService.save(User(name,id,pw,email,address,account))
    }
}
```


소스 코드

2. 시스템 로그인

유저가 ID/PW입력 시 시스템이 입력정보를 받아 DB로 넘김. DB는 입력정보를 토대로 결과 출력

```
fun login(model : Model,
    @RequestParam("id") id : String,
    @RequestParam("pw") pw : String, session: HttpSession, req : HttpServletRequest) : String{
    val found = userService.findById(id) // userId로 db 검색
    var loggedin = false
    if(found == null){
    }
    else{
        if(pw == found.password){
            //println("찾았다")
            session.setAttribute( name: "user", found) // session 에 유저 정보 등록
            loggedin = true
        }
        else{
        }
    }
}

if(loggedin == false) { // 로그인 실패
    return "redirect:/login"
}

var redirectURI = req.session.getAttribute( name: "redirectURI") // 로그인 성공 후 접근했던 이전 URI로 연결
if(redirectURI == null) // 이전 URI 없을 시 메인으로 보냄
    redirectURI = "/"
else{
    redirectURI = req.session.getAttribute( name: "redirectURI") //
    if(redirectURI == "http://localhost:8080/signup") // 회원가입 후 로그인 장 이동했을시 경우만 예외처리
        redirectURI = "http://localhost:8080/"
}

println("redirectURI = $redirectURI");
return "redirect:" + redirectURI
}
```

소스 코드

3. 도서 등록

도서 등록 시 이름, 분류, 품질, 희망가격을 입력하면
시스템에서 정보를 받아 DB에 입력하고 등록 결과를
출력

```
class UploadController(val postService: PostService, val booktransService: BooktransService) {
    @PostMapping("/upload")
    // bookname author publisher
    fun upload(session: HttpSession, req: HttpServletRequest, @RequestParam("image") multipartFile: MultipartFile) : String{
        val bookName = req.getParameter( name: "bookname")
        val author = req.getParameter( name: "author")
        val publisher = req.getParameter( name: "publisher")
        val quality = req.getParameter( name: "quality")
        val booktype = req.getParameter( name: "booktype")
        val price = req.getParameter( name: "price")
        val comment = req.getParameter( name: "comment")
        var imageName: String = StringUtils.cleanPath(multipartFile.originalFilename.toString())

        val uploadDir = "src/main/resources/static/image-upload/" // 업로드된 이미지 폴더
        imageName = (1+postService.count()).toString() + "_" + imageName
        //imageName = bookId_원래이름
        postService.imageUpload(uploadDir, imageName, multipartFile)
        val currentBook = postService.save(Book(bookName, author, publisher, quality, booktype, price, comment, imageName))
        val currentUser: User = session.getAttribute( name: "user") as User

        booktransService.save(Booktrans(currentBook.bookId, buyerId: -1, currentUser.id, isinPurchase: true, state: 0L))

        return "<script>" + "location.href='/';" + "</script>";
    }
}
```

소스 코드

4. 도서 검색

검색창에 키워드를 입력하면 DB에 포함된 도서들의 정보를 출력

```
@Controller
class ResearchController (val bookService: BookService, val booktransService: BooktransService){

    @GetMapping("/research")
    fun research(model : Model, @RequestParam qs : String) : String{

        var bookList = bookService.searchByBookname(qs)
        if(bookList != null)
            bookList = bookList.reversed() // 최신순으로 뒤집어줌

        val bookStatus = mutableListOf<Long>()
        if (bookList != null) {
            for(item in bookList) {
                val tmp = booktransService.findByBookId(item.bookid!!)
                bookStatus.add(tmp.state)
            }
        }

        model.addAttribute( attributeName: "bookList", bookList)
        model.addAttribute( attributeName: "bookStatus", bookStatus)
        return "research_page"
    }
}
```

소스 코드

5. 도서 상세 정보 페이지

검색 목록에 있는 특정 도서의 상세정보를 출력

```
fun test(req: HttpServletRequest, model : Model, @RequestParam bookid : String) : String {
    try {
        //println("bookId " + bookid)
        val book = bookService.findByBookId(bookid.toLong())
        if(book != null) {
            model.addAttribute(attributeName: "bookname", book.bookName)
            model.addAttribute(attributeName: "btype", book.bookType)
            model.addAttribute(attributeName: "bauth", book.author)
            model.addAttribute(attributeName: "bpub", book.publisher)
            model.addAttribute(attributeName: "bcomm", book.comm)
            model.addAttribute(attributeName: "bqual", book.quality)
            model.addAttribute(attributeName: "bpri", book.price)
            model.addAttribute(attributeName: "bimage", book.imageName)
        }
        else {
            return "null"
        }
    } catch (e:Exception){
        e.printStackTrace()
    }
    return "detailtest";
}
```

소스 코드

```
// 이전에 해당 유저가 담아놓은 장바구니 목록
val prevCartList = cartService.findAllByUserId(user.id!!)
val bookInCartList = ArrayList<Book>() // 해당 유저 장바구니에 담긴 책들 목록

// 기존에 담겨있을 책들 완성
if(prevCartList != null) {
    for (prevBook in prevCartList) {
        val book = bookService.findById(prevBook.bookId)
        if(book != null)
            bookInCartList.add(book)
    }
}
```

6. 장바구니(장바구니에 담기)
장바구니 DB에 담겨진 도서들의 목록을 출력해주며
구매하기 버튼을 누르면 장바구니 DB는 초기화

```
// 새로운 장바구니에 담지는 책Id
var bookIds = request.getParameterValues( name: "books");
if(bookIds != null) {
    for (bookId in bookIds) {
        val book = bookService.findById(bookId.toLong())
        var overlap = false
        var cartInUser = cartService.findAllByUserId(user.id!!)
        if (cartInUser != null) { // 기존 장바구니에 있는지, 중복되는지 탐색
            for (bookInCart in cartInUser) {
                if (bookInCart.bookId == book!!.bookId) {
                    overlap = true
                    break;
                }
            }
        }
        if (!overlap) { // 장바구니에 없을 시 장바구니 목록에 추가
            cartService.save(Cart(bookId.toLong(), user.id!!))
            bookInCartList.add(book!!)
        }
    }
}

// view단에 books로 장바구니 안의 책들을 넘겨준다
model.addAttribute( attributeName: "books", bookInCartList)
```

소스 코드

6. 장바구니(구매하기)

장바구니에 담겨 있는 책을 구매 할 때 이미 누구한테 팔렸으면 구매할 때 자동으로 장바구니 구매목록에서 삭제가 됨

```
fun purchase(session: HttpSession): String {
    val currentUser: User = session.getAttribute( name: "user") as User
    val currentCart = cartService.findAllByUserId(currentUser.id!!)
    var scripts = "<script>"
    var flag = false
    if (currentCart != null) {
        for(item in currentCart) {
            val bookInfo = bookService.findById(item.bookId)
            val booktransInfo = booktransService.findById(item.bookId)
            if(booktransInfo.state == 0L) {
                booktransInfo.buyerId = currentUser.id
                booktransInfo.state = 1L
                booktransService.save(booktransInfo)
            }
            else{ // 장바구니에 담아놓았던 책이 누군가에게 팔렸음
                // 리다이렉트 전에 플래그로 alert 띄우기?
                if(!flag) {
                    scripts += "alert('";
                    flag = true
                }

                val book = bookService.findById(item.bookId)
                scripts += book!!.bookName
                scripts += "\n"
            }
            cartService.deleteByCartId(item.cartId!!)
        }
    }

    if(flag)
        scripts += "책은 현재 구매 진행중이거나 판매 완료되었습니다. \n자동으로 구매 목록에서 제외됩니다.'');"
    scripts += "location.href='mypageinfo';" + "</script>"

    println("scripts " + scripts)

    return scripts
}
```

소스 코드

```
if (buyingList != null) {
    for(item in buyingList) {
        val b = bookService.findByBookId(item.bookId!!)
        val seller = userService.findById(item.sellerId!!)
        var stat = ""
        if (item.state == 1L) {
            inPur++
            stat = "구매진행중"
        }
        else if(item.state == 2L) {
            pur++
            stat = "구매완료"
        }

        val obj = BookTransInfo(b!!.imageName, b.bookName, b.author, stat,
            seller.name, user.name, item.transId, seller.account)
        buyingBookList.add(obj)
    }
}
```

7. 마이페이지

유저의 책 구매/판매 현황과 각각의 목록을 출력

```
// 현재 사용자가 판매중인 리스트
if (sellingList != null) {
    for(item in sellingList) {
        val b = bookService.findByBookId(item.bookId!!)
        var buyer = "미정"
        if (item.buyerId != -1L) {
            val tmp = userService.findById(item.buyerId!!)
            buyer = tmp.name
        }
        var stat = ""
        if (item.state == 1L || item.state == 0L) {
            inSell++
            if (buyer.equals("미정")) stat = "판매중"
            else stat = "구매진행중"
        } else if(item.state == 2L){
            sell++
            stat = "판매완료"
        }

        val obj = BookTransInfo(b!!.imageName, b.bookName, b.author, stat, user.name, buyer, item.transId)
        sellingBookList.add(obj)
    }
}

if (cartList != null) { // 현재 유저의 장바구니에 담긴 책 개수
    inCart = cartList.size
}
```

소스 코드

7. 마이페이지(입금확인, 구매취소)

구매자가 장바구니에서의 구매하기 버튼을 누르면
판매자의 경우 입금확인 버튼이 활성화 되며
구매자의 경우 구매취소 버튼이 활성화된다.

```
@PostMapping("/payconfirm")
fun paymentConfirm(req: HttpServletRequest): String {
    val transId = req.getParameter( name: "transId")
    val bookTransInfo = booktransService.findById(transId.toLong())
    bookTransInfo.isInPurchase = false
    bookTransInfo.state = 2L
    booktransService.save(bookTransInfo)

    return "redirect:/mypageinfo"
}

@PostMapping("/paycancel")
fun paymentCancel(req: HttpServletRequest): String {
    val transId = req.getParameter( name: "transId")
    val bookTransInfo = booktransService.findById(transId.toLong())
    bookTransInfo.isInPurchase = false
    bookTransInfo.state = 0L
    bookTransInfo.buyerId = -1L
    booktransService.save(bookTransInfo)

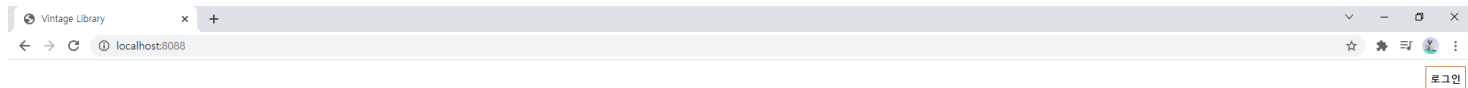
    return "redirect:/mypageinfo"
}
```


07

웹페이지
시연 화면



메인페이지



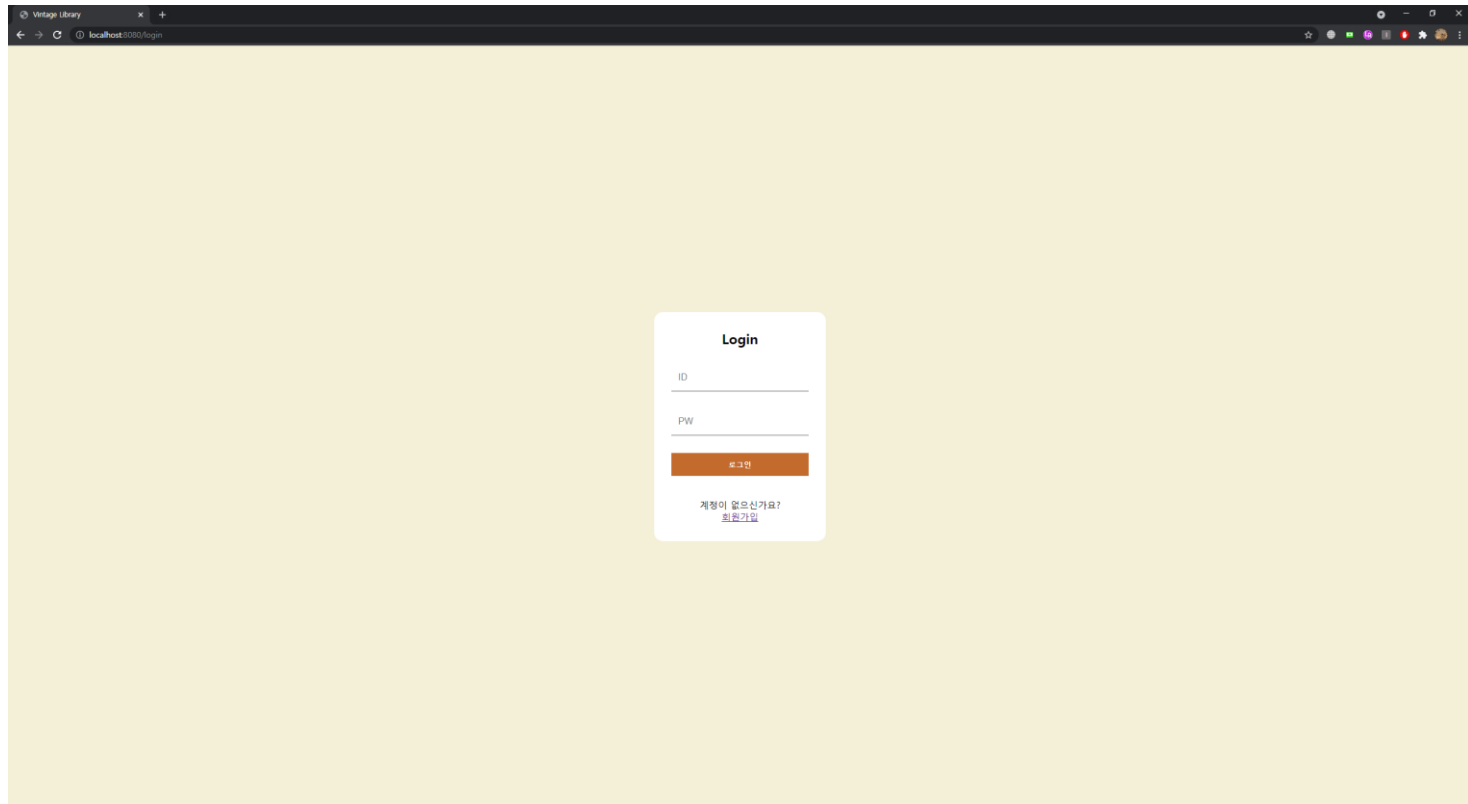
Vintage Library

책 이름 또는 저자를 입력하세요.



로그인

로그인 페이지



The screenshot shows a web browser window with the address bar displaying "localhost:3000/login". The page has a light yellow background. In the center, there is a white login form with the following elements:

- Login** (Title)
- (ID field)
- (PW field)
- (Login button)
- 계정이 없으신가요? [회원가입](#) (Link for users who don't have an account)

회원가입 페이지

SignUp!

홍길동

abcd1234

aaa@aaa.com

서울


111-11111111

회원가입

검색페이지

localhost:8080/research?q=


localhost:8080/research?q=



Vintage Library

[메인 페이지](#)[검색 페이지](#)[마이 페이지](#)[장바구니 페이지](#)[도서등록 페이지](#)[회원가입 페이지](#)

전체 상품



유튜브 레볼루션

로버트 킨슬

더퀘스트

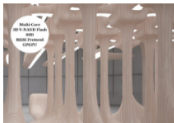
15000원

유튜브 관련된 도서입니다

평점 4.5/5

상태 : 상

☐



컴퓨터구조론(개정판 5판)(양장본 HardCover)

김종현

생능출판

상세페이지



마이페이지

MyPage

localhost:8080/mypageinfo

aaa111 님
bbb@bbb.com

구매/판매 현황

0
장바구니


2
구매진행중

0
구매완료

1
판매중

0
판매완료

판매목록



유튜브 레볼루션


로버트 킨슬

판매자: 홍길동

입금확인

구매진행중

구매목록



컴퓨터 왕기초 초보탈출비법(뚝뚝뚝 배우는)


IT교재연구팀

판매자: 홍길동

계좌번호: 111-11111111

구매취소

구매진행중



컴퓨터구조론(개정판 5판)(양장본 HardCover)

김종현

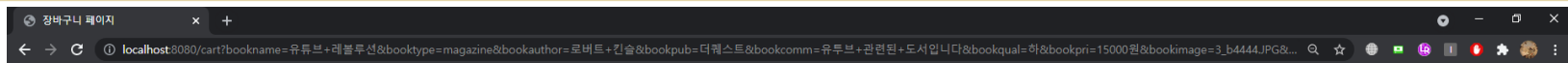
판매자: 홍길동

계좌번호: 111-11111111

구매취소

구매진행중

장바구니페이지




Vintage Library

책 이름 또는 저자를 입력하세요.

검색

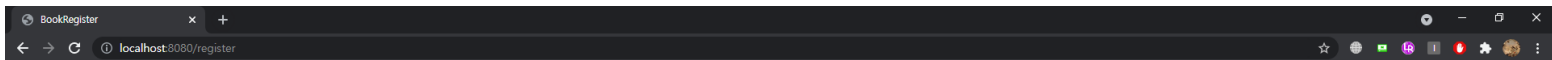
장바구니 상품

이미지	상품명	가격	품질	삭제
	컴퓨터구조론(개정판 5판)(양장본 HardCover)	30000원	중	<input type="checkbox"/> 삭제
	컴퓨터 왕기초 초보탈출비법(특별특약 배우는)	20000원	상	<input type="checkbox"/> 삭제

총 가격 : 50,000원

구매하기

등록페이지



도서 등록

도서 이름	컴퓨터구조론(개정판 5판)(양장본 HardCover)	검색
저자	김종현	
출판사	성남출판	
도서 품질	<input type="radio"/> 최상 <input type="radio"/> 상 <input checked="" type="radio"/> 중 <input type="radio"/> 하	
도서 분류	교재	
희망 가격	30000원	
도서 설명	컴퓨터구조론에 관련된 책입니다	
도서의 상태 이미지 등록	파일 선택 b2.jpg	

도서 등록

실제 시연

감사합니다