

프로젝트 완료보고서

(프로젝트명 : 서울시 공공도서관 도서정보 제공 및 만남 공유 서비스)

2019.11.18

학과	컴퓨터공학과	금융수학과
성명	한다현	김도희
학번	201636096	201732707

목 차

1. 모바일 웹 프로젝트	3
1.1 애플리케이션 개요	3
1.2 애플리케이션 구성	4
1.3 애플리케이션 실행화면	7
1.4 데이터베이스 설계	18
1.5 애플리케이션 구현	20
2. 후기	89

1. 모바일 웹 프로젝트

1.1 애플리케이션 개요

■ 애플리케이션 명

『 도서관 꽃이 피었습니다 』

■ 애플리케이션 요약

도서관 꽃이 피었습니다 에서는 서울시 공공도서관의 위치정보, 이용시간 및 휴관일, 가는 방법, 대출안내, 회원가입 요건, 전화번호, 도서관 소개 등의 도서관 전체 현황정보를 한눈에 찾아볼 수 있습니다.

원하시는 도서관의 소장도서정보를 검색할 수 있습니다.

매번 업데이트되는 추천도서를 확인하세요.

도서의 리뷰조회, 작성을 하며 사용자와 리뷰를 공유하세요.

도서관에서 혼자 밥 먹고, 혼자 공부하기 싫은 날, 이곳에서 친구를 찾아보세요.

1. 로그인: 회원가입, 로그인, 회원탈퇴 할 수 있습니다.
2. 지도 제공: 지도 상에서 핀으로 원하는 도서관을 선택할 수 있습니다.
3. 선택된 도서관의 전체 현황정보와 소장자료 리스트: 도서관을 선택하면 해당 도서관의 전체 현황정보와 키워드 검색 시 관련 소장도서를 리스트로 보여줍니다.
4. 혼공싫어요, 혼밥싫어요, 스터디해요 게시판: 친구구하기 기능을 통해 도서관 내 스터디 등 실제 만남 서비스를 제공합니다.
5. 추천도서 리스트: 추천도서를 3권씩 보여줍니다. 추천도서는 메인 페이지에 새로 들어올 때마다 매번 업데이트 됩니다.
6. 도서 리뷰 조회 및 작성: 도서의 리뷰를 조회하고 직접 작성하고 다른 사용자와 공유할 수 있습니다.
7. 쪽지 기능: 사용자들끼리 쪽지를 주고받을 수 있습니다.
8. 개인서재 기능: 개인서재 기능을 통하여 내가 적은 글을 관리할 수 있습니다.

1.2 애플리케이션 구성

■ 프로젝트 구성도

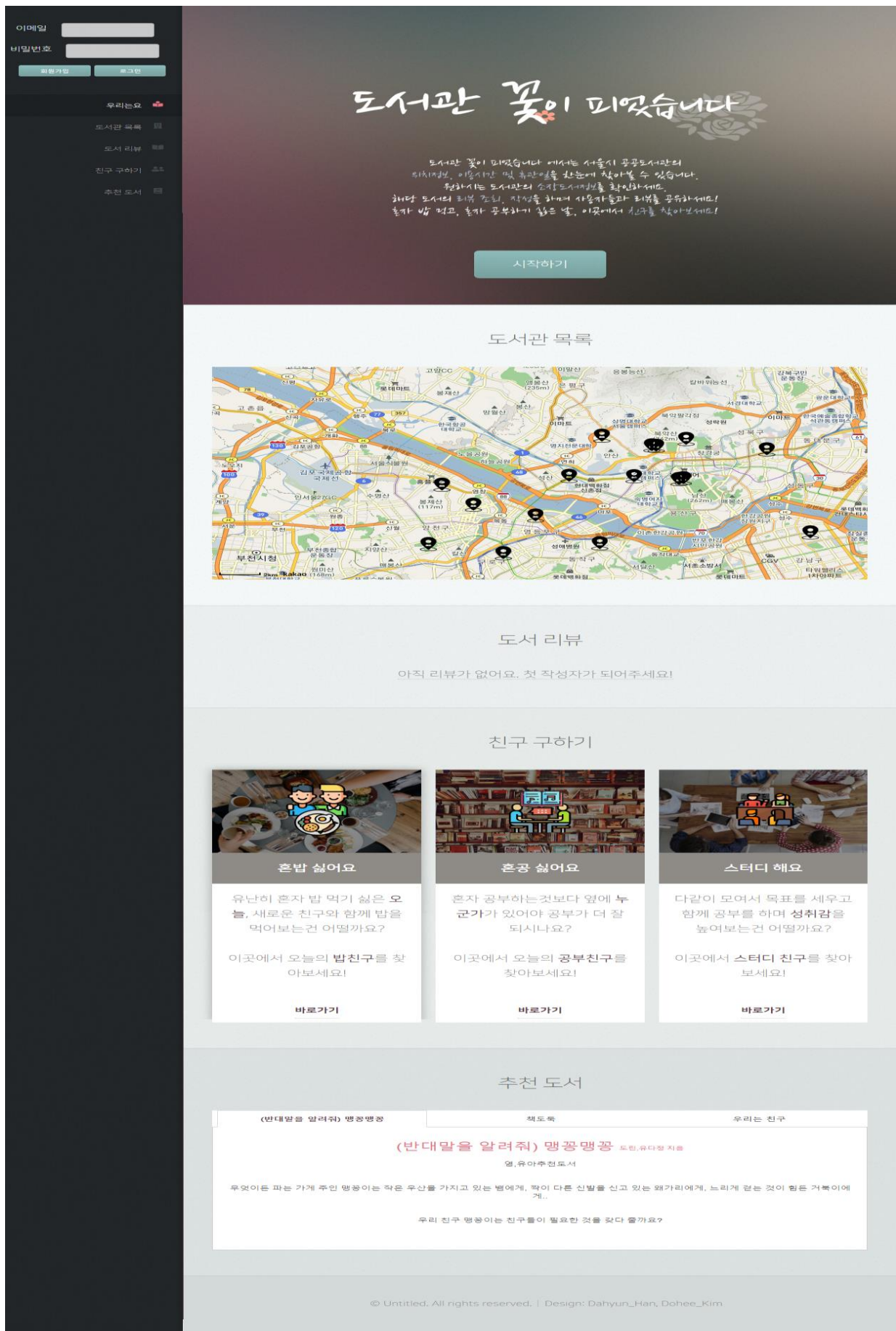


■ 실행환경

- 서버OS: 윈도우10
- 서버IP: localhost
- 서버접속URL: localhost:4500/
- 데이터베이스 : Mongo DB
- 개발언어 : HTML5, CSS3, jQuery, Bootstrap, Node.js, ……

■ 메인 화면 구성도

■ 메인화면 구성도를 그릴 것. (메인화면을 캡처해서 넣고 구성을 설명할 것)

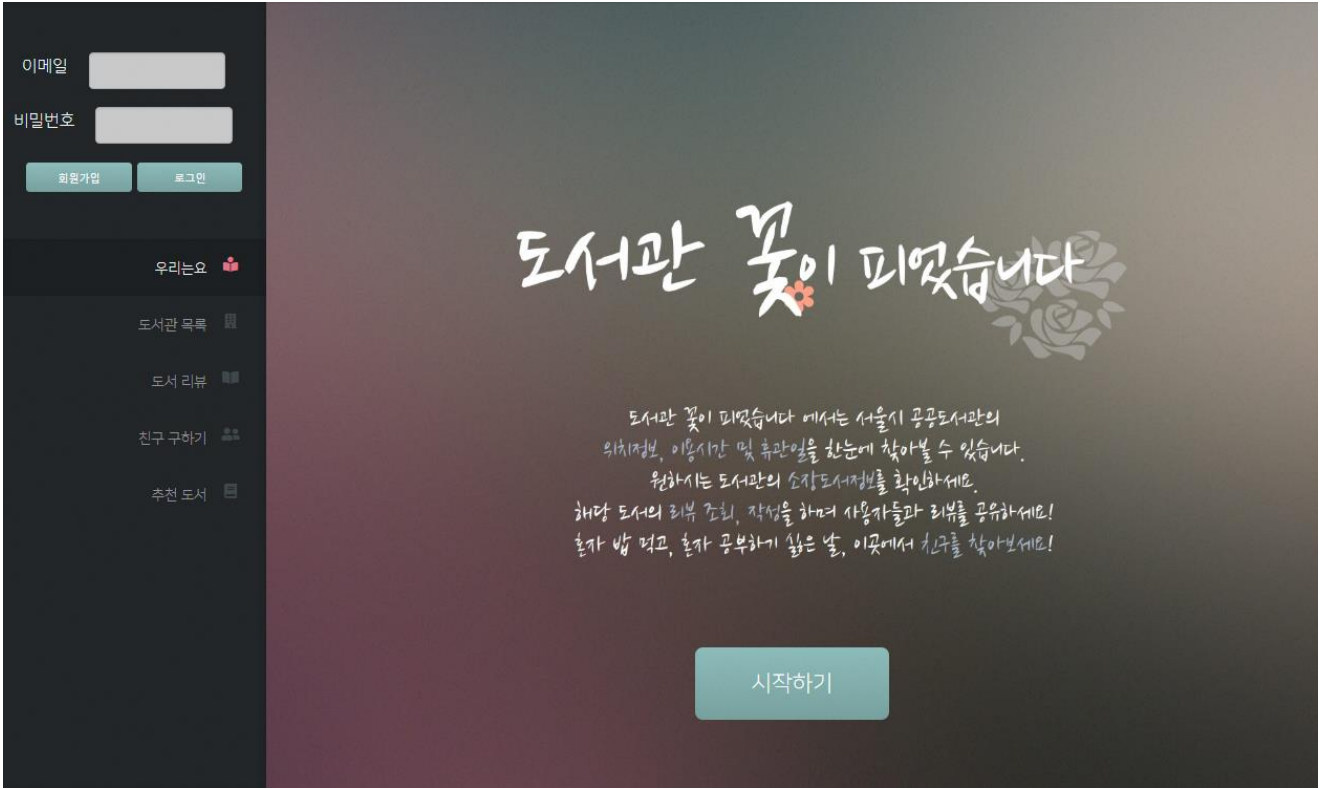
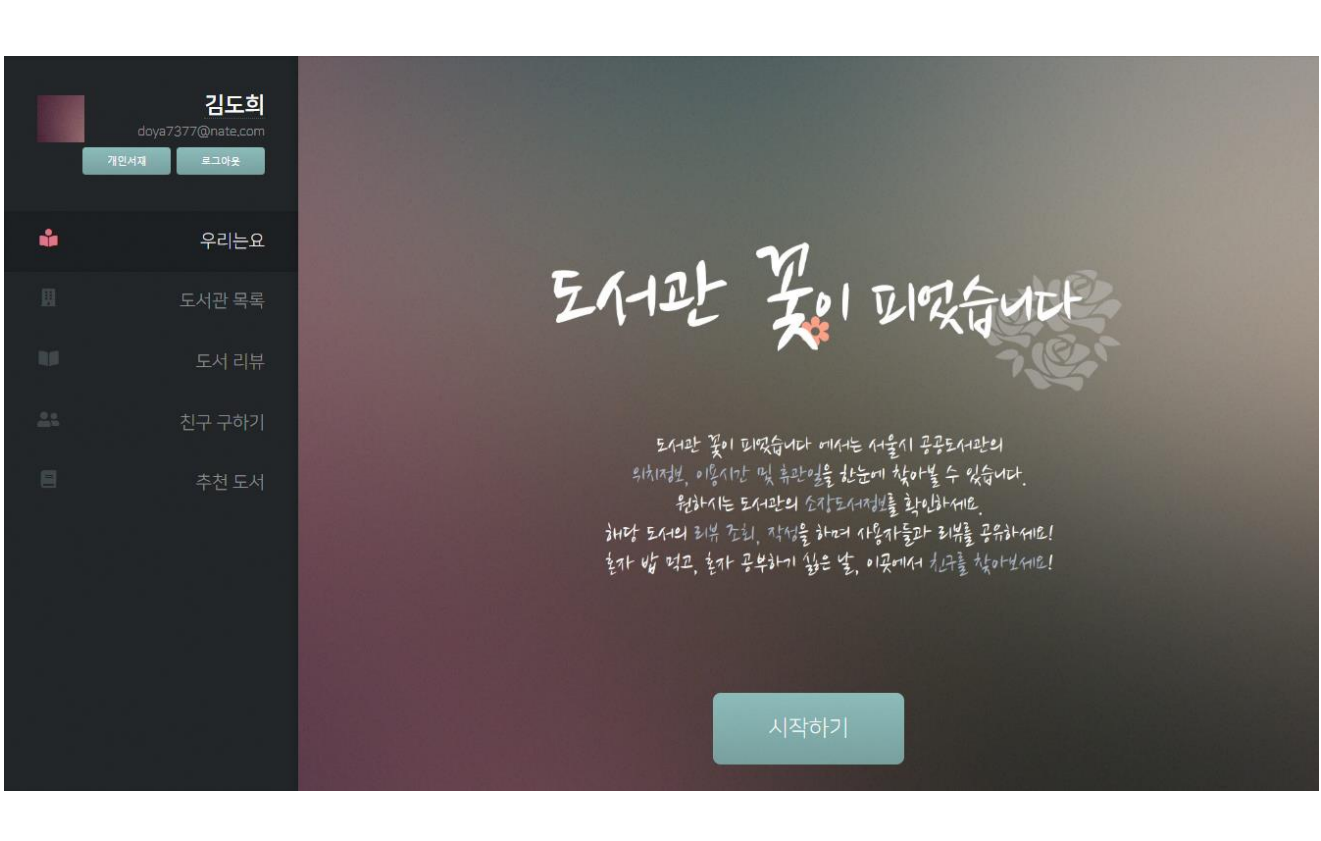


■ 실행 기능

주요기능	상세기능	비고
지도 제공 기능	도서관 위치를 지도상에서 확인할 수 있는 기능	메인화면
지도 선택 기능	지도 상에서 핀으로 원하는 도서관을 선택할 수 있는 기능	메인화면
소장자료 리스트기능	해당 도서관의 소장도서들을 테이블로 표시	도서관화면
도서관 현황정보 제공 기능	해당 도서관의 전체 현황정보를 확인할 수 있는 기능	도서관화면
추천도서 리스트 기능	매번 업데이트되는 추천도서들을 표시	메인화면
친구구하기 게시판 기능	친구구하기 게시판을 통해 도서관 내 스터디 등 실제 만남 서비스 기능	친구화면
도서리뷰 조회 기능	다른 사용자의 도서리뷰 조회 기능	리뷰화면
도서리뷰 작성 기능	원하는 도서의 도서리뷰 작성 기능	리뷰화면
로그인 기능	회원가입 및 로그인 기능	회원가입화면
쪽지 기능	사용자들끼리 쪽지를 주고받을 수 있는 기능	쪽지화면
개인서재 기능	개인서재(마이페이지)에서 나의 글을 관리하고, 받은 쪽지를 확인하고, 회원탈퇴를 할 수 있는 기능	마이페이지
메인화면 바로가기 기능	어느페이지에 있던 메인화면으로 바로 이동하는 기능	모든화면
관리자 기능	회원확인, 쪽지 관리 기능	관리자로그인시

1.3 애플리케이션 실행 화면


실행순서 대로 모든 화면을 캡처해서 넣어주세요..

화면1

메인화면
화면2

로그인화면(로그인시)

화면3

이메일
비밀번호

회원가입 로그인


회원가입

도서관 꽃이 피었습니다

회원가입

이름
이메일
비밀번호

회원가입


회원가입화면


화면4


회원가입성공


이메일
비밀번호


회원가입 로그인


우리는요


도서관 목록


도서 리뷰



친구 구하기



추천 도서


패스워드가 틀립니다.


이메일
비밀번호


회원가입 로그인


우리는요


도서관 목록


도서 리뷰



친구 구하기



추천 도서


매칭되는 아이디가 없습니다.


이메일
비밀번호


회원가입 로그인


우리는요


도서관 목록


도서 리뷰



친구 구하기


추천 도서

User validation failed: email: 이미 존재하는 이메일주소입니다.

이메일
비밀번호

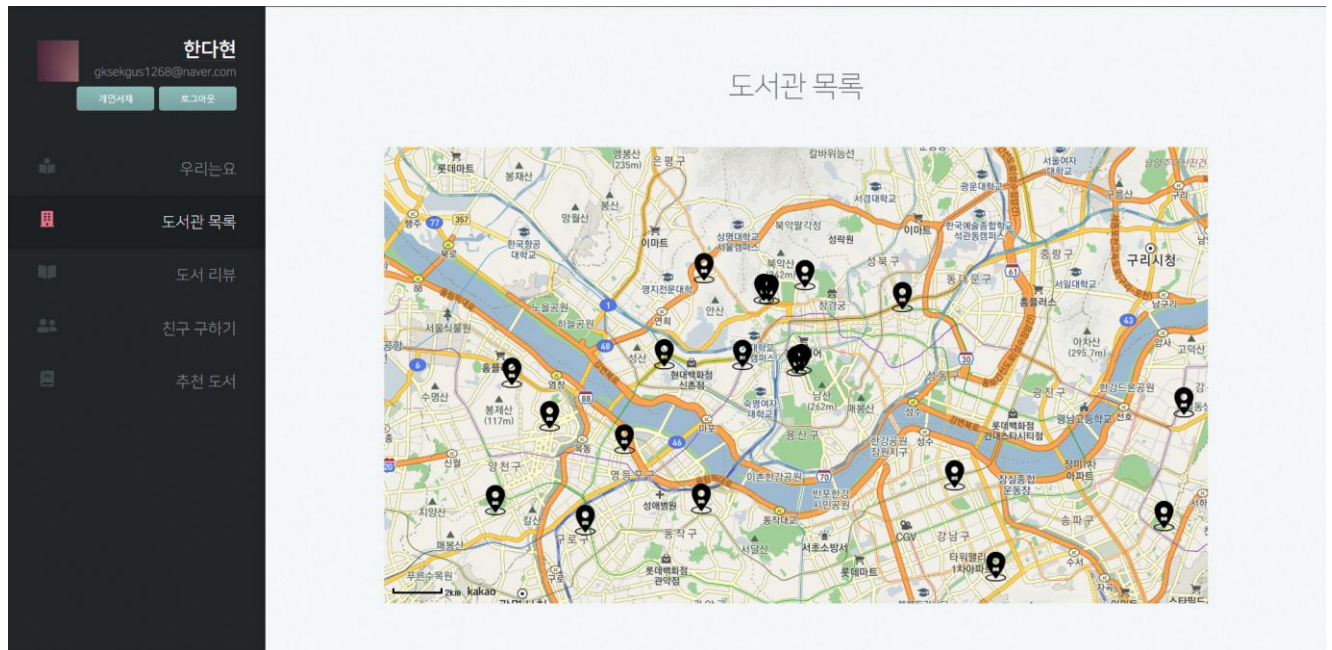
회원가입 로그인


회원가입

다양한 알림 화면- 내비게이션에 표시됨.

- 8 -

화면5



지도 제공 및 핀 선택 화면

화면6



도서관 전체 현황정보 화면

화면7

한다현

gksekus1268@naver.com

로그아웃

로그아웃

소장 도서

소장 도서 정보

제목	저자	출판사	위치
프로그래머가 알아야 할 1%의 핵심원리: 한 번 읽고 이해하는 컴퓨터의 핵심 개념	조현호,이광섭 지음	원시성만남	005-x614호
(우리 부모님을 위한) 컴퓨터 무작정 따라하기 원도 무?	강남구여성능력개발센터 지음	갈뎃	005.44-7239호
[책소개] 나게 줄거임! 컴퓨터 기초	데이스엠의 기획팀 기획:홍종기 집필	데이스엠	여 004-o334호-2
시사문드의 원리: 컴퓨터 음악의 모든 것	최종문 지음	한국학술정보:이담북스	676.7-x664호
(초등학교) 컴퓨터와 생활, 6학년	초등ICT교육연구회 편	책문서	여 004-x286호-6
영어 몰래 PC방: 몰랐사용? 컴퓨터	차영준 글:한태준 그림	타임주니어	여 031-7158호-2
1억에 빠른 양자 컴퓨터가 온다! 인공지능의 미래를 결정할 양자 컴퓨터 이야기	나시모의 허태호시,오오제키 마사유키 지음:신정태 옮김	로도북	004.1-486호
그래픽스 오퍼: 컴퓨터 코딩의 여행	글쓴이: 로리 윌리엄스그림이: 케이트 루:옮긴이: 장종원	두레메이들	여 998.5-o554호
세미다: 영동현 상상이 컴퓨터 프로그램을 만들었어	픽토나 로빈슨 지음:권지현 옮김	씨드북	여 998.5-a536호
LINE 과학: 첨단과학,002, 슈퍼컴퓨터	이준범 글:강기수 그림	한태교육	여 608-a216호-2

<

1

2

3

4

5

>

다른 책 조회하기

소장자료 리스트 화면

화면8

이메일

비밀번호

회원가입

로그인

우리는요

도서 리뷰

아직 리뷰가 없어요. 첫 작성자가 되어주세요!



한다현

gksekus1268@naver.com

로그아웃

로그아웃

우리는요

도서관 목록

도서 리뷰

친구 구하기

추천 도서

도서 리뷰

Node.js 교과서


조현영 | 갈뎃

노드의 기본부터 실무까지, 전 과정을 한 권에 담았다!

더 보기

도서리뷰 화면(메인)


화면9




한혜선
hsszz1120@naver.com

회원회원

로그아웃



도서 리뷰



내가 쓴 리뷰

도서 리뷰

작성하기

	한줄평	제목	저자
보기	쉽게 배우는 JSP 웹 프로그래밍	송미영	한빛아카데미
보기	파이썬 웹 프로그래밍	김석준	한빛미디어
보기	Node.js 교과서	조현영	길벗
보기	인사이드 자바스크립트	송형주, 고현준	한빛미디어

<

1

>

내가 쓴 리뷰

작성하기

	한줄평	제목	저자
보기	쉽게 배우는 JSP 웹 프로그래밍	송미영	한빛아카데미


<

1

>

도서리뷰 화면


화면10



한다현
gksekgus1268@naver.com

회원회원

로그아웃



리뷰 작성

리뷰 작성

리뷰 작성

한줄평

책 제목

작가

출판사

내용

사진 첨부

파일 선택

선택된 파일 없음

등록

리뷰 작성 화면

화면11

한다현

gksekgus1268@naver.com

로그아웃

리뷰 보기

리뷰 작성

책 제목

인사이드 자바스크립트

작가

송형주, 고현준

출판사

한빛미디어

한줄평

자바스크립트를 제대로 공부하고자 한다면 이 책을 추천할게요.

내용

자바스크립트를 제대로 공부하고자 하는 개발자에게 여러 자바스크립트 응용 기술들을 소화할 수 있는 기초 체력을 기를 수 있게 도와주는 좋은 가이드가 될 것이다.

수정

목록으로

삭제

(내가 작성한 리뷰)

한다현

gksekgus1268@naver.com

로그아웃

리뷰 보기

책 제목

쉽게 배우는 JSP 웹 프로그래밍

작가

송미영

출판사

한빛아카데미

한줄평

단계별로 쇼핑몰을 구현하며 배우는 JSP 웹 프로그래밍.

내용

JSP의 이론적 개념 → 기본 실습 → 응용 실습 순의 단계별 학습이 가능합니다. 응용 실습이 합쳐져 최종적으로 쇼핑몰 하나를 완성하는 구성이라 배운 내용이 어디에 어떻게 적용되는지 알 수 있습니다.

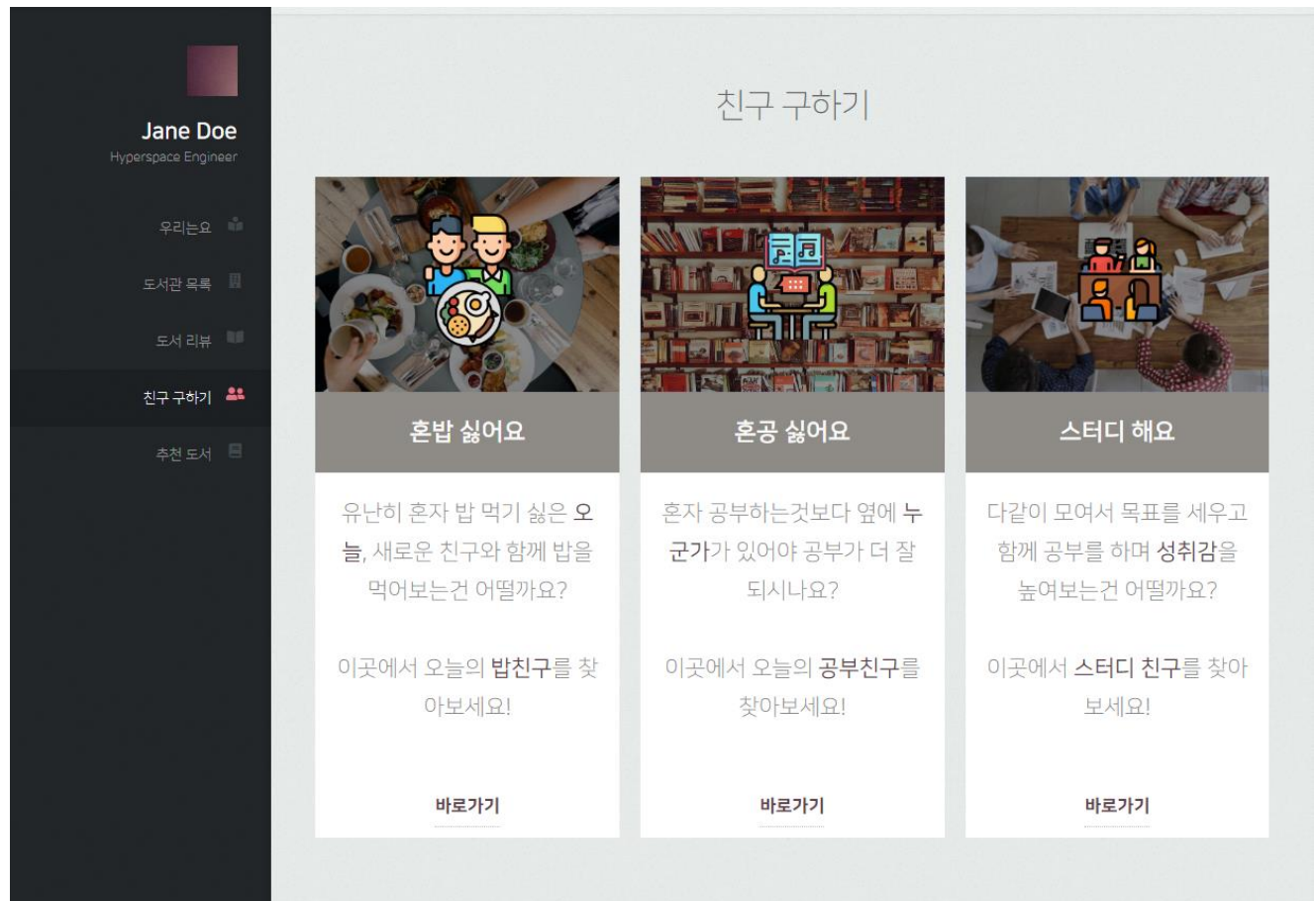
목록으로

(다른 사람이 작성한 리뷰)

리뷰 읽기 화면

- 12 -

화면12



화면 13

The image displays three sequential screenshots of a web application interface, likely a document management or study tool. Each screenshot shows a table with columns for document location, menu/topic, and content. The first screenshot is titled '혼밥 싫어요' (I don't like eating alone), the second '혼공 싫어요' (I don't like studying alone), and the third '스터디 해요' (Let's study). Each table has a '연락하기' (Contact) button on the left and a pagination bar at the bottom showing page 1 of 1.

First Screenshot: 혼밥 싫어요

	도서관 위치	메뉴	내용
연락하기	송파도서관	한식	오늘 점심에 한식 같이 드실 분 계신가요?

Navigation: < 1 >

Second Screenshot: 혼공 싫어요

	도서관 위치	대상	내용
연락하기	동대문도서관	상관없음	오늘 수학 공부할건데 모르는거 들어가봐면서 같이 하실 분?

Navigation: < 1 >

Third Screenshot: 스터디 해요

	도서관 위치	주제	내용
연락하기	고척도서관	토익	토익 스터디 모집합니다. 목표 점수 850점 이상

Navigation: < 1 >

친구 구하기 화면

화면14



한다현

gksekgus1268@naver.com

회원가입

로그아웃

 밥 메이트 구함


밥 메이트 구함

도서관

메뉴


내용

등록



한다현
gksekgus1268@naver.com

마이페이지 로그인

 공부 메이트 구함


공부 메이트 구함

도서관

대상


내용

등록



한다현
gksekgus1268@naver.com

마이페이지 로그인

 스터디 구함

스터디 구함

도서관

주제

내용

등록

친구 구하기 글 작성 화면

화면15



한다현
gksekgus1268@naver.com

마이페이지 로그인

 쪽지 보내기

쪽지 보내기

내용

보내기

쪽지 보내기 화면

화면16



김도희
doys7377@nate.com

가입일자 로그인

우리는요

도서관 목록

도서 리뷰

친구 구하기

추천 도서

추천 도서

언론이 말해주지 않는 불편한 진실

손정의

괴물재국 중국

언론이 말해주지 않는 불편한 진실 박종성 지음



언론이 말해주지 않는 불편한 진실
박종성 / 북스코프 / 2012 / 334 쪽 75원

문화포그림 담당 김대원

인터넷 창을 열어 포털사이트에서 세계인구시계를 검색해보자 - (U.S.)로 표시된 미국의 인구와 (World)로 표시된 세계의 인구를 볼 수 있는 사이트다. 자 이번에는 새로운 창(F5)를 실행시켜 눌러보자. 약 1초도 지나지 않은 시간동안 수 십명의 생명이 태어나는 모습을 숫자를 통해 보게 될 것이다. 자 이제 70억명이 같은 공간에 모여 동시에 태어난다는 상상을 해보자. 너무 시끄러워 우리는 미쳐버릴지도 모른다. 수 십만(70억명) 사람들이 살아가는 지구에서 모든 것을 모든 상황을 정확히 안다는 것은 얼마나 어려운 일인가? 더구나 세상의 소식을 알려주는 언론(미디어)이 알려주지 않는 일을 안다는 것은 너무나 힘든 일이다.

역사는 두 가지 모습을 가지고 있다. 하나의 모습은 실제 일어난 일이나 행위 그 자체이며, 또 다른 모습은 일어난 일이나 행위에 대한 기록을 말한다. 전자는 객관적인 사실로서의 역사, 후자는 역사가에 의해 주관적으로 서술된 역사를 말한다. 그래서 역사를 한 가지 모습 안에서만 이해하고 파악하는 것은 매우 위험한 일이 된다.

[언론이 말해주지 않는 불편한 진실]은 이런 역사의 불편한 진실을 정확히 알고 있는 책이다. 시간이 부족한 현대인을 위해 6가지 키워드 <양국화, 분장, 종교, 민족, 환경, 질병>로 나누어 사건을 요목조목 설명한다. 복잡한 세계에서 벌어지는 수많은 사건을 6개의 키워드와 24가지의 이야기를 통해 담았고, 전전히 꽃아가지는 것만으로도 한자를 살아가기 위한 필요한 상식을 제공하고 균형 잡힌 시각을 통해 세계를 바라볼 수 있도록 각 장을 구성 하였다.

저자 박종성은 현재 기자답게 여러 매체의 기사와 참고 문헌 등 많은 자료를 적절히 활용해 사건을 설명한다. 또한 머리말에서 다음과 같이 밝히며 하나의 역사적 사실을 인식하는 중요한 관점을 제시한다. "이 책은 우리가 사는 세상에서 일어나는 사건의 배경을 이해하는 데 조금이나마 도움을 주기 위해 쓰였다. 어떻게 해서 사건이 발생했고, 어떠한 과정을 거쳤으며, 앞으로는 어떻게 흘러갈지에 대해 다루었다. 일부가 아닌 전체로서 사건을 조망한 것이다."

정보의 홍수라고 표현되는 정보화시대 현재를 살아가기 위해 가장 중요한 책목으로 떠오르는 것이 올바른 정보를 수집하고 접근하며 정보를 올바르게 이해하는 것이다. 따라서 이 책처럼 다양한 기사와 참고 문헌을 적절히 활용한 책은 소중한 정보원이 된다. 마지막으로 저자의 말을 끝으로 서평을 마친다.

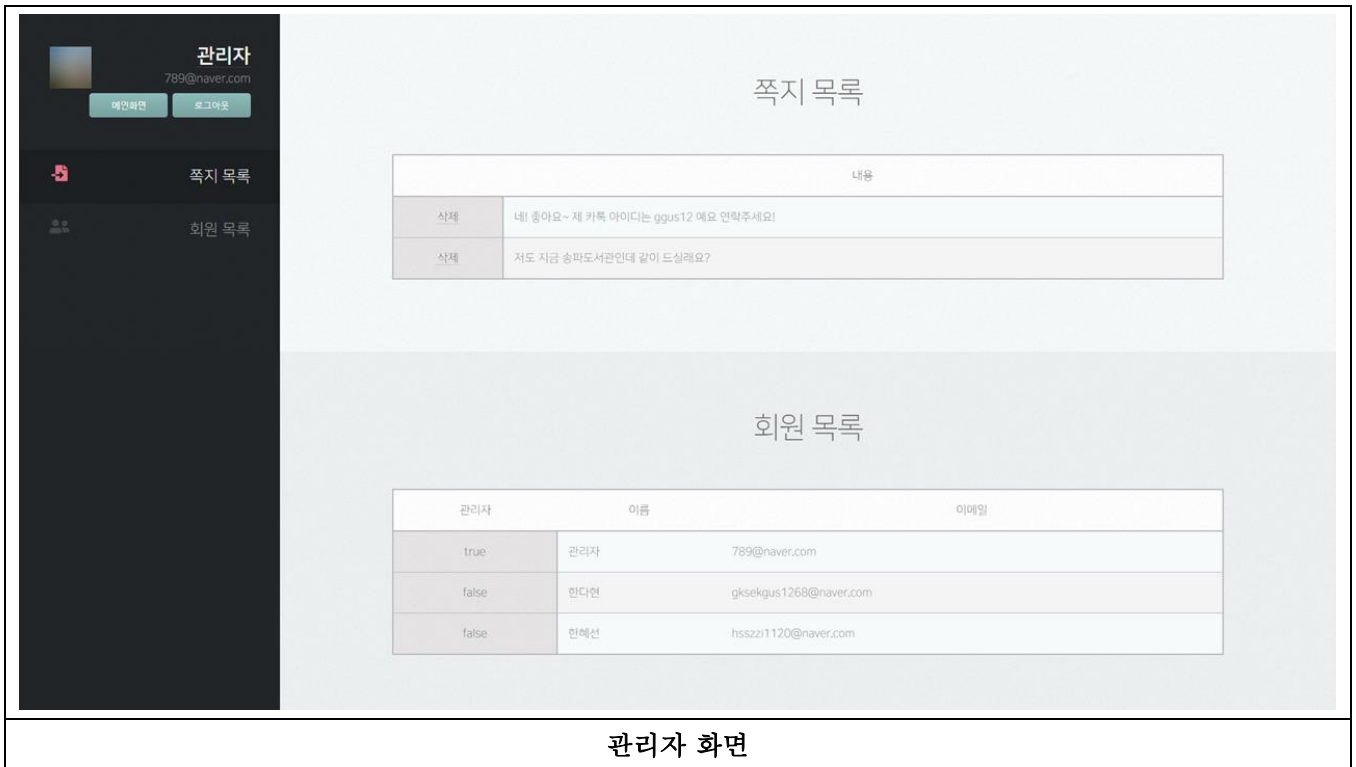
"논란의 일부만이 아니라 전체를 볼 수 있을 때, 비로소 사건의 전체를 볼 수 있고, 숨어 있는 '불편한 진실'과 마주할 수 있을 때만이 더 나은 내일로 나아갈 수 있다."

* 이 책은 동대문구정보화도서관 종합자료1관(2층)에서 보실 수 있습니다.

추천도서 화면

화면17

- 17 -



1.4 데이터베이스 설계

■ 데이터베이스 명 : LibraryDB

□ 테이블명 : Review

Name	Type	Etc
_id	ObjectId	NOT NULL
title	String	NOT NULL
book	String	NOT NULL
author	String	NOT NULL
publisher	String	NOT NULL
content	String	NOT NULL
user	String	
createdAt	Date	
photo	ObjectId	

□ 테이블명 : Food

Name	Type	Etc
_id	ObjectId	NOT NULL
food	String	NOT NULL
content	String	NOT NULL

location	String	NOT NULL
user	String	
createdAt	Date	
ID_HANDWRITING	INTEGER	
CREATE_DATE	TIMESTAMP	

□ 테이블명 : Study

Name	Type	Etc
_id	ObjectId	NOT NULL
target	String	NOT NULL
content	String	NOT NULL
location	String	NOT NULL
user	String	
createdAt	Date	

□ 테이블명 : Groupstudy

Name	Type	Etc
_id	ObjectId	NOT NULL
subject	String	NOT NULL
content	String	NOT NULL
location	String	NOT NULL
user	String	
createdAt	Date	

□ 테이블명 : Message

Name	Type	Etc
_id	ObjectId	NOT NULL
content	String	NOT NULL
receiver	String	NOT NULL
sender	String	NOT NULL
createdAt	Date	

□ 테이블명 : User

Name	Type	Etc
_id	ObjectID	NOT NULL

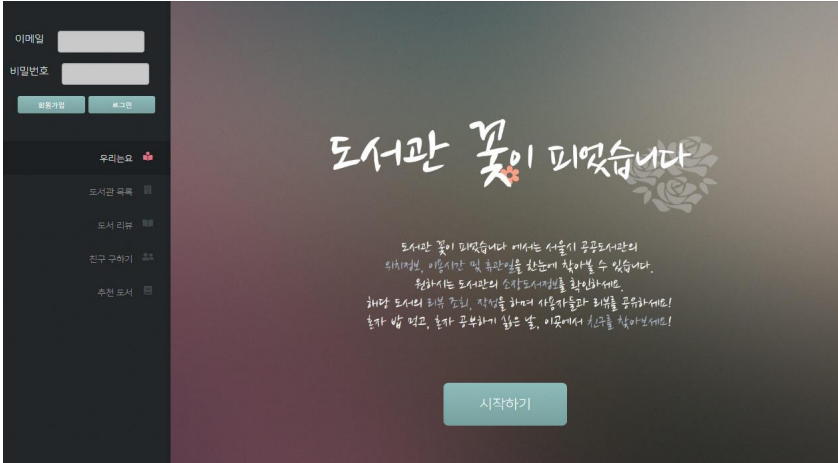
name	String	NOT NULL
email	String	NOT NULL
hashed_password	String	NOT NULL
admin	Boolean	default: false
image	String	default: '/images/avatar.jpg'
salt	String	NOT NULL

□ 테이블명 : Upload

Name	Type	Key	Etc
_id	ObjectID		NOT NULL
relatedID	ObjectID		
type	String		
filename	String		
originalname	String		
size	INT32		
createdAt	Date		

1.5 애플리케이션 구현

■ 화면1, 화면2

화면1, 화면2	설명
	<p>- 메인화면은 타이틀과 시작하기 버튼으로 구성</p> <p>- 왼쪽상단에 로그인창과 회원가입 버튼, 하단의 메뉴선택바가 있음</p> <p>- 사용자의 편리성을 높이기 위해 웹페이지의 기능을 소개하는 간단한 설명을 덧붙임</p> <p>- 로그인시 화면</p>

	<p>-사용자 이름과 프로필사진, 아이디가 표시됨</p> <p>-개인서재 버튼과 로그아웃 버튼이 생성됨</p>
--	---

소스코드

default.hbs

```
<!DOCTYPE html>
<html>
<head>
<title>도서관 꽃이 피었습니다.</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
  <link rel="stylesheet" href="/assets/css/main.css" />
  <link href="/lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
  <link href='https://cdnjs.cloudflare.com/ajax/libs/limonte-sweetalert2/6.11.2/sweetalert2.min.css' rel='stylesheet' type='text/css'>
</head>
<body class="is-preload">
<!-- Header -->
<div id="header">
<div class="top">
<!-- Logo -->
{{#if isUserLoggedIn}}
<div id="logo">
<span class="image avatar48"></span>
<h1 id="title"><a href="/mypage">{{user.name}}</a></h1>
<p>{{user.email}}</p>
<button type="button" onclick="{{status.link}}" class="btnlogout">{{status.print}}</button>
<button type="button" onclick="location.href ='/logout'" class="btnlogout">로그아웃</button>
</div>{{else}}<div><p></p></div><div class="col-md-12">
{{#each alert.message}}
<div class="alert alert-success">{{this}}</div>
{{/each}}
```

[illegible]

```

<script src="/assets/js/browser.min.js"></script>
<script src="/assets/js/breakpoints.min.js"></script>
<script src="/assets/js/util.js"></script>
<script src="/assets/js/main.js"></script>
<script
src="//dapi.kakao.com/v2/maps/sdk.js?appkey=27120206d14510c7ad8b8418b99243a2"></script>
<script src="/lib/owlcarousel/owl.carousel.min.js"></script>
<script src="/lib/wow/wow.min.js"></script>
<script src="/js/main.js"></script>
<script src="/assets/js/index.js"></script>
<script src="/lib/jquery/jquery.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/limonte-sweetalert2/6.11.2/sweetalert2.all.min.js"></script>
<script src="/lib/bootstrap/js/bootstrap.bundle.min.js"></script>
<script type="text/javascript" src="/js/recommendapi.js"></script>
</body>
</html>

```

index.hbs

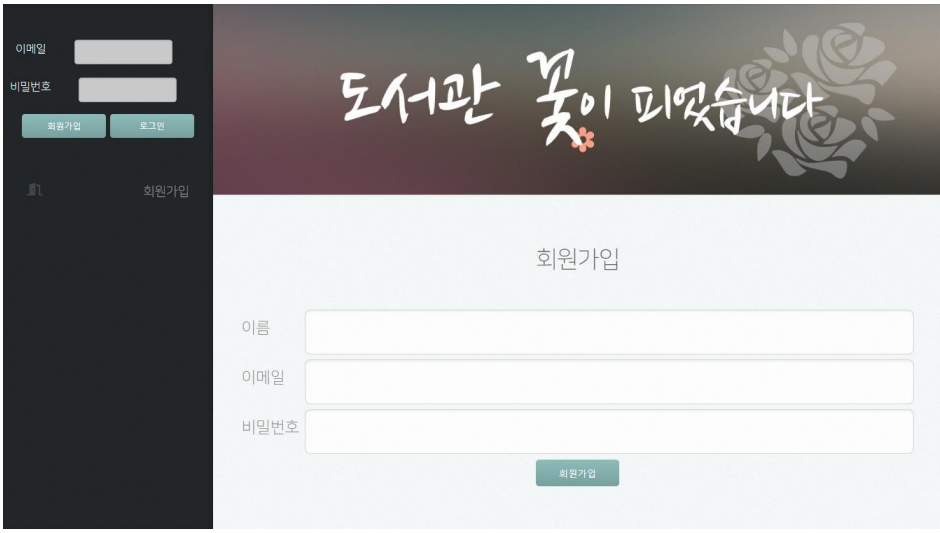
```

<!-- Nav -->
<nav id="nav">
<ul>
<li><a href="#top" id="top-link"><span class="icon solid fa-book-reader">우리는요</span></a></li>
<li><a href="#portfolio" id="portfolio-link"><span class="icon solid fa-building">도서관 목록</span></a></li>
<li><a href="#about" id="about-link"><span class="icon solid fa-book-open">도서 리뷰</span></a></li>
<li><a href="#contact" id="contact-link"><span class="icon solid fa-user-friends">친구 구하기</span></a></li>
<li><a href="#recommend"><span class="icon solid fa-book">추천 도서</span></a></li>
</ul></nav></div></div>
<!-- Main -->
<div id="main">
<!-- Intro -->
<section id="top" class="one dark cover">
<div class="container">
<p><br><br><br><br><br><br><br><br><br><br><br><br><br></p>
<footer>
<a href="#portfolio" class="button scrolly">시작하기</a>
</footer>

```

</div>
</section>

■ 화면3

화면3	설명
	<p>- 이름과 이메일주소, 비밀번호를 입력하면 회원가입이 가능하도록 함</p>

소스코드

signup.hbs

```
<!-- Nav -->
<nav id="nav"><ul>
<li><a href="#portfolio" id="portfolio-link"><span class="icon solid fa-door-open">회원가입
</span></a></li></ul></nav></div></div>

<!-- Main -->
<div id="main">
<!-- Intro -->
<section id="top" class="five dark cover">
<div class="container">
</div></section>

<!-- About Me -->
<section id="about" class="two">
<div class="container">
<header>
<h2>회원가입</h2>
</header>
<div class="col-md-12">
{{#each alert}}
<div class="alert alert-danger">
{{this}}
</div>
{{/each}}
```

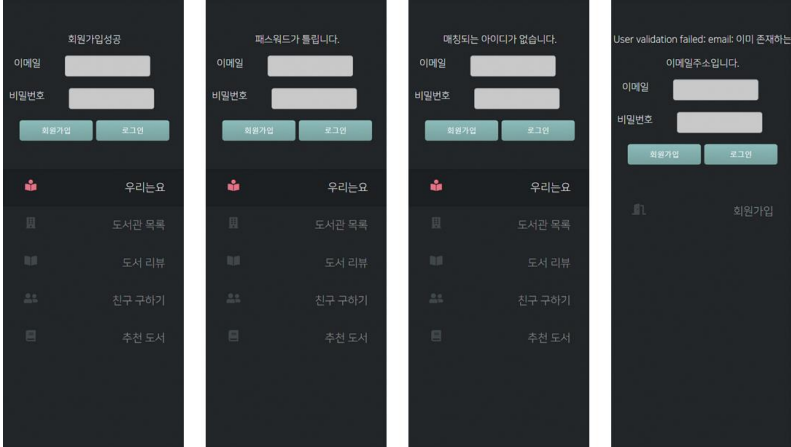


```

<form method="post" action="/create">
  <div class="form-group">
    <label for="inputEmail3" class="control">이름</label>
    <input type="text" class="textcontrol" name="name">
  </div>
  <div class="form-group">
    <label for="inputEmail3" class="control">이메일</label>
    <input type="email" class="textcontrol" name="user_email">
  </div>
  <div class="form-group">
    <label for="inputEmail3" class="control">비밀번호</label>
    <input type="password" class="textcontrol form-pass" name="user_password">
  </div>
  <button type="button" class="btnsingup">회원가입</button>
</form></div></div></section>
</div>

```

■ 화면4

화면4	설명
	<p>- 회원가입에 성공했을 때, 회원가입 할 때 아이디가 이미 중복되어 있을 때 로그인 할 때 비밀번호가 틀렸을 때, 아이디를 잘못 입력했을 때 상황별로 출력하는 에러 메시지가 있음.</p>

소스코드

local.js

```

/* 로컬 인증 방식으로 스트래티지 설정 모듈 */
console.log('call : /config/passport/local.js');
const mongoose = require('mongoose');
// 사용자의 이메일과 비밀번호를 전달받아 db에 저장된 정보와 비교하는 로컬 인증 방식 기능 제공
const LocalStrategy = require('passport-local').Strategy;
// User 모델 참조
const User = mongoose.model('User');
// 로컬 인증 방식으로 스트래티지 설정
module.exports = new LocalStrategy({ //첫번째 파라미터

```

```

    usernameField: 'user_email',
    passwordField: 'user_password',
  },
  //클라이언트의 요청파라미터를 받아서 처리
  async function (email, password, done) { //두번째 파라미터:검증 콜백 함수(local 인증처리)
    console.log('passport local 인증 처리');
    /* User모델(users 컬렉션)에서 이메일을 기준으로 검색합니다. */
    const user = await User.findOne({
      "email":email, //email로 사용자 정보가 있는지 확인() "email":속성에 email 파라미터를 전달
    }).exec();
    /* 인증결과를 done()메소드를 이용하여 authenticate 쪽으로 알려 주어야
    라우팅 함수안에서 authenticate를 호출했을 때 각각의 상황에 따라 분기가 됨 */
    if (user) {
      /* 이메일로 검색된 유저가 있다면 패스워드가 맞는지 확인합니다.*/
      /** authenticate 메소드는 UserSchema에서 선언한 메소드로서,
      유저로부터 비밀번호를 받아 대조하는 기능을 수행하는 메소드 */
      if (user.authenticate(password)) { //인증된 경우
        console.log('done 콜백을 통해서 user객체 정보를 serializeUser() 메서드로 전달');
        /* 유저가 있다면 */
        return done(null, user)
      } else {
        /* 패스워드가 틀리다면 */
        return done(null, false, "패스워드가 틀립니다."); //설정된값은 req.flash()에서 받은 값에서 error
        object로 받습니다.
      }
    } else {
      /*이메일로도 없는경우 */
      return done(null, false, "매칭되는 아이디가 없습니다."); //설정된값은 req.flash()에서 받은 값에서
      error object로 받습니다.
    }
  },
);

passport.js

/* 사용할 인증 방식 등록
- 인증 후 사용자 정보를 세션에 저장하거나 사용자 정보를 복원하는 모듈 */

/* 인증 방식 설정 파일 참조
- 인증 방식별로 설정 파일을 만들어 스트래티지를 선언해야함
*/

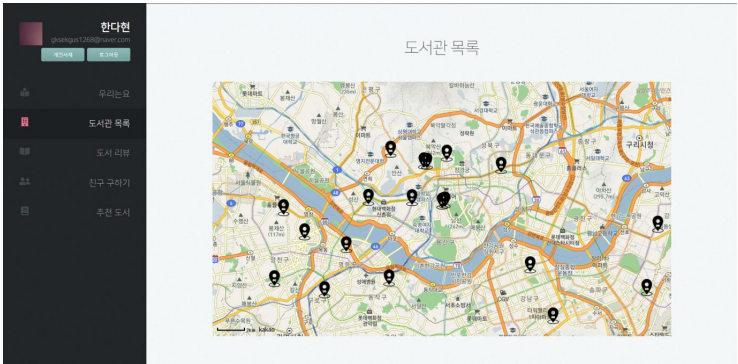
```

```

const local = require('./local');//로컬 스트래티지
const LocalStrategy = require('passport-local').Strategy;
console.log('call : /config/passport/passport.js');
module.exports = function (passport) {
  /* 사용자 인증에 성공했을 때 호출되는 serializeUser() 콜백 함수는
  전달받은 user.email 객체의 정보를 콘솔에 출력하고 done() 메소드를 호출함
  - done() 메소드의 두번째 파라미터로 user.email 객체를 전달하면,
  이 객체는 그 다음에 처리할 함수쪽으로 전달됨
  - 즉, 인증을 통해 로그인에 성공하였다면 이쪽에서 어떤 사용자 정보를 세션에 저장할지 설정해줍니다.*/
  passport.serializeUser(function (user, done) {
    console.log('passport.serializeUser() 호출 : ' + user.email);
    done(null, user.email)//user.email을 세션에 저장하는 done()
  });
  /* 사용자 인증 이후 사용자 요청이 있을 때마다 호출됨 - 로그인 상태인 경우 */
  /* 사용자로부터 받은 세션정보와 실제 DB의 데이터와 비교하여줍니다.*/
  /* 세션값과 나의 값을 체크해줌으로써 보안을 위한 기능입니다. */
  passport.deserializeUser(function (email, done) {
    console.log('passport.deserializeUser() 호출 : ' + email);
    const profile = {email: email};
    done(null, profile);//세션 상태인지를 계속 체크해서 필요한 처리
  });
  // 사용하기 위해 참조한 인증방식을 미들웨어로 등록
  passport.use(local);
};

```

■ 화면5

화면5	설명
	<p>- 지도 제공 및 핀 선택 화면에서는 서울시 공공도서관의 위치정보를 지도상에서 확인할 수 있음</p> <p>-도서관의 위치를 핀으로 표시하여 클릭 시 해당도서관의 상세정보페이지로 이동하게 됨</p>
소스코드	
<pre> index.hbs <section id="portfolio" class="two"> <div class="container"> </pre>	

```

<header><h2>도서관 목록</h2></header>
<div class="mappart" id="map"></div>
</div></section>
<section id="about" class="three">
<div class="container">
<header><h2>도서 리뷰</h2></header><section id="testimonials"><div>
    <div class="justify-content-center">
        <div>
            <div class="owl-carousel testimonials-carousel">
                {{#each reviews}}
<div class="testimonial-item">
    {{#if this.photo}}

    {{else}}
    
    {{/if}}
    <h3>{{this.book}}</h3>
    <h4>{{this.author}} | {{this.publisher}}</h4>
    <p>{{this.title}}</p>
    <a href="/review" class="more"> 더 보기</a>
    </div>
    {{else}}
    <a href="/writereview" class="more">아직 리뷰가 없어요. 첫 작성자가 되어주세요!</a>
    </div>{{/each}}</div></div>
</div></section></div></section>

```

showmap.js

```

$(document).ready(function() {
    var selectlibrary = "";
    console.log("js파일 시작");
    var mapContainer = document.getElementById('map'), // 지도를 표시할 div
    mapOption = {
        center: new kakao.maps.LatLng(37.55200494, 126.9801399), // 지도의 중심좌표
        level: 8 // 지도의 확대 레벨
    };

    var map = new kakao.maps.Map(mapContainer, mapOption); // 지도를 생성합니다
    // 마커를 표시할 위치와 title 객체 배열입니다

```

```

var positions = [
    {
        title: '강남도서관',
        latlng: new kakao.maps.LatLng(37.51368143, 127.0468374)
    }, {
        title: '강동도서관',
        latlng: new kakao.maps.LatLng(37.53815107, 127.1434874)
    }, {
        title: '강서도서관',
        latlng: new kakao.maps.LatLng(37.5479003, 126.8599262)
    }, {
        title: '개포도서관',
        latlng: new kakao.maps.LatLng(37.48308019, 127.063867)
    }, {
        title: '고덕평생학습관',
        latlng: new kakao.maps.LatLng(37.55930391, 127.1579066)
    }, {
        title: '고척도서관',
        latlng: new kakao.maps.LatLng(37.50534626, 126.8530515)
    }, {
        title: '구로도서관',
        latlng: new kakao.maps.LatLng(37.49867236, 126.8911162)
    }, {
        title: '남산도서관',
        latlng: new kakao.maps.LatLng(37.55255227, 126.9821555)
    }, {
        title: '노원평생학습관',
        latlng: new kakao.maps.LatLng(37.63953995, 127.067837)
    }, {
        title: '도봉도서관',
        latlng: new kakao.maps.LatLng(37.65253218, 127.0127628)
    }, {
        title: '동대문도서관',
        latlng: new kakao.maps.LatLng(37.57348736, 127.0247655)
    }, {
        title: '동작도서관',
        latlng: new kakao.maps.LatLng(37.50596898, 126.9401028)
    }, {

```

```

        title: '마포평생학습관아현분관',
        latlng: new kakao.maps.LatLng(37.55416693, 126.9573466)
    }, {
        title: '마포평생학습관',
        latlng: new kakao.maps.LatLng(37.5543222, 126.9242046)
    }, {
        title: '서대문도서관',
        latlng: new kakao.maps.LatLng(37.58331954, 126.9408583)
    }, {
        title: '송파도서관',
        latlng: new kakao.maps.LatLng(37.49999503, 127.1348515)
    }, {
        title: '양천도서관',
        latlng: new kakao.maps.LatLng(37.53349939, 126.8757691)
    }, {
        title: '어린이도서관',
        latlng: new kakao.maps.LatLng(37.57607257, 126.9683689)
    }, {
        title: '영등포평생학습관',
        latlng: new kakao.maps.LatLng(37.52580934, 126.9071819)
    }, {
        title: '용산도서관',
        latlng: new kakao.maps.LatLng(37.55200494, 126.9801399)
    }, {
        title: '정독도서관',
        latlng: new kakao.maps.LatLng(37.58103147, 126.983694)
    }, {
        title: '종로도서관',
        latlng: new kakao.maps.LatLng(37.5764504, 126.9664672)
    }
];

// 마커 이미지의 이미지 주소입니다
var imageSrc = './images/location.png', // 마커이미지의 주소입니다
    imageSize = new kakao.maps.Size(30, 44), // 마커이미지의 크기입니다
    imageOption = {offset: new kakao.maps.Point(15, 40)}; // 마커이미지의 옵션입니다. 마커의 좌표와 일치시킬 이미지 안에서의 좌표를 설정합니다.

// 마커 이미지의 이미지 크기 입니다
var imageSize = new kakao.maps.Size(24, 35);

```

```
// 마커 이미지를 생성합니다
var markerImage = new kakao.maps.MarkerImage(imageSrc, imageSize);
var libraryMarkers = [];
createLibraryMarkers();
function createMarker(position, title) {
    var marker = new kakao.maps.Marker({
        position:position,
        title:title,
        image:markerImage
    });
    return marker;
}
function createLibraryMarkers() {
    positions.forEach(function(i) {
        var marker = createMarker(i.latlng, i.title);
        libraryMarkers.push(marker);
        marker.setMap(map);

        daum.maps.event.addListener(marker, 'click', function(mouseEvent) {
            location.href = '/librarys/'+ i.title;
        });
    })
}
});
```

■ 화면6

화면6	설명
	<p>- 도서관 전체 현황정보 화면에서는 오픈API를 이용해 도서관의 위치, 가는 방법, 휴관일, 대출안내, 회원가입 요건, 전화번호, 도서관 소개 정보를 제공함</p> <p>-원하는 메뉴 클릭 시 해당 정보 표시</p>
소스코드	
librarys.hbs <nav id="nav">	

```

<ul><li>
<a href="#info" id="portfolio-link"><span class="icon solid fa-building">도서관 정보</span></a></li>
<li><a href="#book" id="about-link"><span class="icon solid fa-book-open">소장 도서 검색</span></a></li></ul><nav></div></div>

    <!-- Main -->
    <div id="main">
<section id="top" class="five dark cover">
    <div class="container"></div></section>
    <section id="info" class="two">
    <div class="container">
        <header>
            <h2 id="libraryname">{{title}}</h2>
        </header><section id="faq">
<div class="row justify-content-center">
    <div class="divsize">
        <ul id="faq-list">
            <li>
                <a data-toggle="collapse" class="collapsed" href="#faq1">도서관 위치<i class="fa fa-minus-circle"></i></a>
                <div id="faq1" class="collapse" data-parent="#faq-list">
                    <p id="llocation"></p></div></li><li>
                <a data-toggle="collapse" href="#faq2" class="collapsed" id>가는 방법<i class="fa fa-minus-circle"></i></a>
                <div id="faq2" class="collapse" data-parent="#faq-list">
                    <p id="lway"></p></div></li><li>
                <a data-toggle="collapse" href="#faq3" class="collapsed">휴관일<i class="fa fa-minus-circle"></i></a>
                <div id="faq3" class="collapse" data-parent="#faq-list">
                    <p id="lholiday"></p>
                </div></li><li>
                <a data-toggle="collapse" href="#faq4" class="collapsed">대출안내<i class="fa fa-minus-circle"></i></a>
                <div id="faq4" class="collapse" data-parent="#faq-list">
                    <p id="lborrow"></p>
                </div></li> <li>
                <a data-toggle="collapse" href="#faq5" class="collapsed">회원가입 요건<i class="fa fa-minus-circle"></i></a>
                <div id="faq5" class="collapse" data-parent="#faq-list">
                    <p id="lsignin"></p>
                </div></li> <li>

```



```

    <a data-toggle="collapse" href="#faq6" class="collapsed">전화번호<i class="fa fa-minus-circle"></i></a>
    <div id="faq6" class="collapse" data-parent="#faq-list">
      <p id="ltel"></p>
    </div></li><li>
      <a data-toggle="collapse" href="#faq7" class="collapsed">도서관 소개<i class="fa fa-minus-circle"></i></a>
      <div id="faq7" class="collapse" data-parent="#faq-list">
        <p id="linfo"></p><div></li></ul></div></div></section></div></section>

```

librarys.js

```

/* 라우팅 함수 선언 [librarys] */

/* mongoose 모듈 참조 */
const mongoose = require('mongoose');
//mongoose.Promise = Promise;

/* Model 불러오기 */
const User = mongoose.model('User');
const Food = mongoose.model('Food');
const only = require('only');//object중 원하는 데이터만 sorting하여 리턴하는 helper 모듈

const {getCodeByName} = require('../..helper/utility');

var status = new Object();
status={"print":"메인화면", "link":"location.href = '/'"};

console.log('call : /controllers/librarys.js');

//도서관 정보 화면
exports.librarys = function (req, res) {
  const isLogin = req.isAuthenticated();
  var title = req.params.title;
  var code = getCodeByName(req.params.title);
  if(isLogin) {
    User.load(req.user.email, function (user) {
      res.render('librarys/librarys', { //뷰 템플릿 렌더링(템플릿:index.hbs)
        title: title, code: code, isUserLoggedIn: isLogin, user: user, status:status //students 객체를 템플릿에 바인딩

```

```

});

});

} else {

    res.render('librarys/librarys', { //뷰 템플릿 렌더링(템플릿:index.hbs)

        title: title, code: code, isUserLoggedIn: isLogin //students 객체를 템플릿에 바인딩


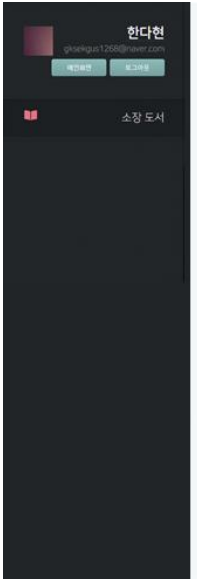
    });

}

});

```

■ 화면7

화면7	설명
	<ul style="list-style-type: none"> - 도서관을 선택하면 소장도서를 검색할 수 있음
	<ul style="list-style-type: none"> - 소장자료 리스트 화면에서는 검색단어가 도서명에 포함되는 모든 도서들을 리스트로 보여줌 - 페이지이동버튼을 통해 다음페이지로 이동 가능 - 하단의 다른책조회하기 버튼을 누르면 다시 소장도서 검색 화면으로 이동

소스코드

librarys.hbs

```

<section id="book" class="three">

    <div class="container">

<header>

    <h2>소장 도서 검색</h2>

    </header>

    <div class="col-md-12">

        <form method="post" action="/bookresult">

            <input type="hidden" name="manageCd" value="{{code}}">

            <input type="hidden" name="numOfRows" value="10">

```

```

        <input type="hidden" name="pageNo" value="1">
        <input type="hidden" name="library" value="{{title}}">
    <div class="form-group">
        <label for="inputEmail3" class="col-sm-2 control">책 제목</label>
        <input type="text" class="textcontrol" name="title"><div>
    <button type="submit" class="btn btn-book-submit">조회하기</button>
    </form>
</div></div></section> </div>
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script type="text/javascript" src="/js/libraryapi.js"></script>

bookresult.hbs
<nav id="nav"><ul>
<li><a href="#book" id="portfolio-link"><span class="icon solid fa-book-open">소장 도서
</span></a></li>
    </ul></nav></div><div><div id="main">
<section id="top" class="five dark cover">
    <div class="container">
    </div></section>
    <section id="book" class="two">
    <div class="container">
    <header>
    <h2>소장 도서 정보</h2>
    </header>
    <table class="table table-hover">
<thead>
    <tr>
    <th scope="col">제목</th>
    <th scope="col">저자</th>
    <th scope="col">출판사</th>
    <th scope="col">위치</th>
    </tr>
</thead>
<tbody>
    {{#each libraryResult.items.item}}
    <tr class="table-odd">
    <th class="titlesize">{{this.title}}</th>
    <td>{{this.author}}</td>
    <td>{{this.publisher}}</td>

```

```

        <td>{{this.callNo}}</td>

    </tr>

    {{/each}}
</tbody>
</table>

<form method="post" action="/bookresult">
    <input type="hidden" name="manageCd" value="{{manageCd}}">
        <input type="hidden" name="numOfRows" value="10">
            <input type="hidden" name="pageNo" value="{{pageNo}}">
            <input type="hidden" class="textcontrol" value="{{title}}" name="title">
            <input type="hidden" class="textcontrol" value="{{library}}" name="library">
        <div>
            <ul id="page" class="pagination">
                <li class="page-item {{first}}">
                    <a class="page-link pageclickpre" href="#page">&laquo;</a>
                </li>
                {{#each print}}
                <li class="page-item {{this.Status}}">
                    <a class="page-link pageclick" href="#page">{{this.page}}</a>
                </li>
                {{/each}}
                <li class="page-item {{second}}">
                    <a class="page-link pageclickne" href="#page">&raquo;</a>
                </li></ul></div></form>
<button type="button" onclick="location.href='/librarys/{{library}}#book'">다른 책 조회하기</button>
</div></section></div>

```

ReqInfoController.js

```

const libraryList = require('../config/libraryData');
const {getCodeByName} = require('../helper/utility');
const {API_URL, SERVICEKEY} = require('../config/environment');
const request = require('request-promise');
const only = require('only');
const mongoose = require('mongoose');
const User = mongoose.model('User');
console.log('call : /controllers/ReqInfoController.js');

```

```
//도서 검색
```

```

module.exports.bookresult = async function (req, res) {
  const libraryResult = await request.get({
    url: API_URL,
    timeout: 10000,
    json: true,
    qs: {
      'serviceKey': SERVICEKEY,
      'title': req.body.title,
      'manageCd': req.body.manageCd,
      'numOfRows': req.body.numOfRows,
      'pageNo': req.body.pageNo,
      '_type': "json",
    },
  }).then((result) => {
    return result.response.body //api 통신에 성공하면 정보를 libraryResult에 저장
  }).catch(e => {
    console.error("request Error : " + e)
  });

  //검색된 전체 도서 갯수를 통해 페이지징 구현
  var last5 = (parseInt(libraryResult.totalCount/50)*5);
  var last = ((parseInt((parseInt(libraryResult.totalCount))-1)/10)+ 1);

  var page = parseInt(req.body.pageNo);
  if(page <= last5) {
    var defaultpage = (parseInt((page-1)/5)*5+ 1);
    var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2, defaultpage+ 3, defaultpage+ 4 ];
  }
  else {
    var etc = last-last5;
    var defaultpage = (parseInt((page-1)/5)*5+ 1);
    switch(etc) {
      case 1:
        var pageArray = [ defaultpage ];
        break;
      case 2:
        var pageArray = [ defaultpage, defaultpage+ 1 ];
        break;
    }
  }
}

```

```

        case 3:
            var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2];
            break;
        case 4:
            var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2, defaultpage+ 3];
            break;
    }
}

var print = new Object();
for(var i=0; i<pageArray.length; i+ ) {
    if(parseInt(req.body.pageNo) == pageArray[i]) {
        print[i] = { "page": pageArray[i], "Status": "active" };
    } else {
        print[i] = { "page": pageArray[i], "Status": "" };
    }
}

if(parseInt(req.body.pageNo) <= 5) {
    var first = "disabled";
    var second = "";
} else if(parseInt(req.body.pageNo) > last5) {
    var first = "";
    var second = "disabled";
} else {
    var first = "";
    var second = "";
}

const isLogin = req.isAuthenticated();
if(isLogin) {
    User.load(req.user.email, function (user) {
        var status = new Object();
        status={"print":"메인 화면", "link":"location.href ='/'};
        res.render('librarys/bookresult', {
            libraryResult: libraryResult, title: req.body.title, manageCd: req.body.manageCd, print:
            print, pageNo: req.body.pageNo, first: first, second: second, library: req.body.library, isUserLoggedIn:
            isLogin, user: user, status:status
        });
    });
}

```

```

    } else {
        res.render('librarys/bookresult', {
            libraryResult: libraryResult, title: req.body.title, manageCd: req.body.manageCd, print: print,
            pageNo: req.body.pageNo, first: first, second: second, library: req.body.library, isUserLoggedIn:
            isLogin
        });
    }
};

```

libraryData.json

```

[ { "title": "강남도서관", "code" : "MA"},
  { "title": "강동도서관", "code" : "MB" },
  { "title": "강서도서관", "code" : "MC" },
  { "title": "개포도서관", "code" : "MD" },
  { "title": "고덕평생학습관", "code" : "ME" },
  { "title": "고척도서관", "code" : "MF" },
  { "title": "구로도서관", "code" : "MG" },
  { "title": "남산도서관", "code" : "MH" },
  { "title": "노원평생학습관", "code" : "MV" },
  { "title": "도봉도서관", "code" : "MJ" },
  { "title": "동대문도서관", "code" : "MK" },
  { "title": "동작도서관", "code" : "ML" },
  { "title": "마포평생학습관아현분관", "code" : "MX" },
  { "title": "마포평생학습관", "code" : "MM" },
  { "title": "서대문도서관", "code" : "MP" },
  { "title": "송파도서관", "code" : "MW" },
  { "title": "양천도서관", "code" : "MN" },
  { "title": "어린이도서관", "code" : "MQ" },
  { "title": "영등포평생학습관", "code" : "MR" },
  { "title": "용산도서관", "code" : "MS" },
  { "title": "정독도서관", "code" : "MT" },
  { "title": "종로도서관", "code" : "MU" }]

```

utility.js

```

const libraryList = require('../config/libraryData');
console.log('call : /helper/utlity.js');
//도서관 코드로 도서관 이름 가져오기
module.exports.getCodeByName = function(code){
    return libraryList.filter(function(data){

```

```

    return data.title == code;
  }][0].code;
};

```

environment.js

```

console.log('call : /config/environment.js');

module.exports = {
  PORT:4500,//포트 번호
  DATABASE:"mongodb://localhost:27017/libraryDB",
  SERVICEKEY:"spIogpzyHT653luGRUeM7ATfWrT7y25rWagxWFC/GX5i4Br2D1gfv+ +
O8RAeQILa61XvGKz1WhM9sqK+ H9qyw==",
  MONGO_SESSION_COLLECTION_NAME:"sessions",
  SESSION_SECRET:"your_secret", //세션 암호화에 사용할 값 -> 랜덤 숫자를 넣으면 됨.
  API_URL:"http://openapi-lib.sen.go.kr/openapi/service/lib/openApi",
  // PAGINATION:{
  //   PAGE_SIZE:10
  // }
};

```

index.js

```

//페이징 form의 페이지 클릭
$(".pageclick").click(function () {
  var pageno = $(this).text();
  console.log(pageno);
  $("input[name='pageNo']").val(pageno);
  $(this).closest("form").submit();
});

$(".pageclickpre").click(function () {
  var pageno = $("input[name='pageNo']").val();
  var setpage = (parseInt((pageno-1)/5-1)*5+ 1);
  $("input[name='pageNo']").val(setpage);
  $(this).closest("form").submit();
});

$(".pageclickne").click(function () {

```



```
var pageno = $("input[name='pageNo']").val();
var setpage = (parseInt((pageno-1)/5+ 1)*5+ 1);
$("input[name='pageNo']").val(setpage);
$(this).closest("form").submit();
});

$(".pageclick2").click(function () {
    var pageno = $(this).text();
    console.log(pageno);
    $("input[name='pageNo2']").val(pageno);
    $(this).closest("form").submit();
});

$(".pageclickpre2").click(function () {
    var pageno = $("input[name='pageNo2']").val();
    var setpage = (parseInt((pageno-1)/5-1)*5+ 1);
    $("input[name='pageNo2']").val(setpage);
    $(this).closest("form").submit();
});

$(".pageclickne2").click(function () {
    var pageno = $("input[name='pageNo2']").val();
    var setpage = (parseInt((pageno-1)/5+ 1)*5+ 1);
    $("input[name='pageNo2']").val(setpage);
    $(this).closest("form").submit();
});

$(".pageclick3").click(function () {
    var pageno = $(this).text();
    console.log(pageno);
    $("input[name='pageNo3']").val(pageno);
    $(this).closest("form").submit();
});

$(".pageclickpre3").click(function () {
    var pageno = $("input[name='pageNo3']").val();
```

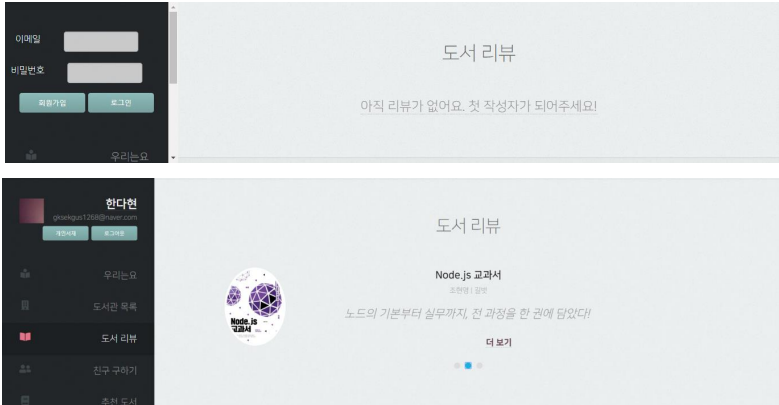
```

var setpage = (parseInt((pageno-1)/5-1)*5+ 1);
$("#input[name='pageNo3']").val(setpage);
$(this).closest("form").submit();
});

$(".pageclickne3").click(function () {
    var pageno = $("#input[name='pageNo3']").val();
    var setpage = (parseInt((pageno-1)/5+ 1)*5+ 1);
    $("#input[name='pageNo3']").val(setpage);
    $(this).closest("form").submit();
});

```

■ 화면8

화면8	설명
	<ul style="list-style-type: none"> -리뷰생성 전과 리뷰생성 후 화면이 메인 화면에서 달라짐 - 다른 사용자들의 도서 리뷰를 볼 수 있음 -슬라이드 형식으로 표시 -더보기 버튼을 클릭하면 해당도서의 상세 페이지로 이동

소스코드

index.hbs

```

<section id="about" class="three">
  <div class="container">
    <header>
      <h2>도서 리뷰</h2>
    </header>
    <section id="testimonials">
      <div>
        <div class="justify-content-center">
          <div>
            <div class="owl-carousel testimonials-carousel">
              {{#each reviews}}
                <div class="testimonial-item">
                  {{#if this.photo}}
                    
                  {{else}}

```

```

        
      {{/if}}
    <h3>{{this.book}}</h3>
    <h4>{{this.author}} | {{this.publisher}}</h4>
    <p>{{this.title}}</p>
    <a href="/review" class="more"> 더 보기</a>
  </div>
  {{else}}
    <a href="/writereview" class="more">아직 리뷰가 없어요. 첫 작성자가 되어주세요!</a>
  </div>    {{/each}}    </div>    </div>    </div>
</section><!-- #testimonials --></div>    </section>

```

main.js

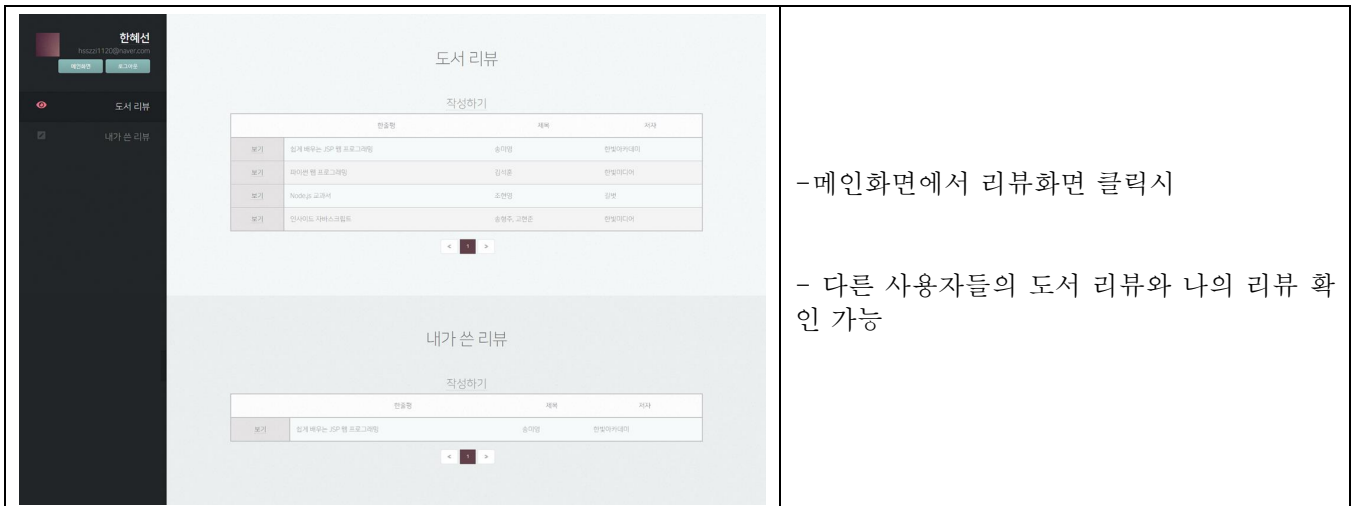
```

exports.index = function (req, res) {
  const isLogin = req.isAuthenticated(); //로그인 여부 확인.
  var statusi = new Object(); //index 화면에서는 로그인시 마이페이지로 갈 수 있게 함.
  Review.select(function (reviews) { //작성된 리뷰들 중 3개만 추출하여 화면에 표시
    if(isLogin) {
      User.load(req.user.email, function (user) { //로그인 한 사용자 정보
        if(user.admin){//관리자가 로그인 했다면 관리자 모드로 연결할 수 있게 함.
          statusi={"print":"관리자모드", "link":"location.href ='/adminpage'"};
        } else {//일반 사용자가 로그인 했다면 마이페이지로 연결할 수 있게 함.
          statusi={"print":"개인서재", "link":"location.href ='/mypage'"};
        }
        res.render('main/index', { //index.hbs와 랜더링
          reviews: reviews, isUserLoggedIn: isLogin, user: user, status:statusi
        });
      });
    } else {
      res.render('main/index', {
        reviews: reviews, isUserLoggedIn: isLogin
      });
    }
  });
};

```

■ 화면9

화면9	설명
-----	----



-메인화면에서 리뷰화면 클릭시

- 다른 사용자들의 도서 리뷰와 나의 리뷰 확인 가능

소스코드

review.hbs

```
<!-- Nav -->
<nav id="nav">
<ul>
<li><a href="#review" id="portfolio-link"><span class="icon solid fa-eye">도서 리뷰</span></a></li>
<li><a href="#ireview" id="about-link"><span class="icon solid fa-pen-square">내가 쓴 리뷰</span></a></li>
</ul>
</nav>
</div>

<!-- Main -->
<div id="main">
<section id="top" class="five dark cover">
<div class="container">
</div>
</section>

<section id="review" class="two">
<div class="container">
<header>
<h2>도서 리뷰</h2>
</header>
<a href="/writereview">작성하기</a>
<table class="table table-hover">
<thead>
<tr>
<th scope="col"></th>
<th scope="col">한줄평</th>
<th scope="col">제목</th>
<th scope="col">저자</th>
</tr>
</thead>
```

```

</thead>
<tbody>
  {{#each reviews}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -->
    <tr class="table-odd">
      <th scope="row"><a href="/readreview/{{this._id}}">보기</a></th>
      <td>{{this.book}}</td>
      <td>{{this.author}}</td>
      <td>{{this.publisher}}</td>
    </tr>
  {{/each}}
</tbody>
</table>
<form method="post" action="/review">
  <input type="hidden" name="pageNo2" value="{{pageNo2}}">
  <input type="hidden" name="pageNo" value="{{pageNo}}">
  <div>
    <ul id="page" class="pagination">
      <li class="page-item {{first1}}">
        <a class="page-link pageclickpre" href="#review">&laquo;</a>
      </li>
      {{#each print}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -->
      <li class="page-item {{this.Status}}">
        <a class="page-link pageclick" href="#review">{{this.page}}</a>
      </li>
      {{/each}}
      <li class="page-item {{second}}">
        <a class="page-link pageclickne" href="#review">&raquo;</a>
      </li>
    </ul>
  </div>
</form>
</div>
</section>
<section id="ireview" class="three">
  <div class="container">
    <header>
      <h2>내가 쓴 리뷰</h2>
    </header>
  </div>
</section>

```

```

        <a href="/writereview">작성하기</a>

        <table class="table table-hover">

<thead>
    <tr>
        <th scope="col"></th>
        <th scope="col">한줄평</th>
        <th scope="col">제목</th>
        <th scope="col">저자</th>
    </tr>
</thead>
<tbody>
    {{#each myreviews}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -->
    <tr class="table-odd">
        <th scope="row"><a href="/readreview/{{this._id}}">보기</a></th>
        <td>{{this.book}}</td>
        <td>{{this.author}}</td>
        <td>{{this.publisher}}</td>
    </tr>
    {{/each}}
</tbody>
</table>
<form method="post" action="/review">
    <input type="hidden" name="pageNo" value="{{pageNo}}">
    <input type="hidden" name="pageNo2" value="{{pageNo2}}">
    <div>
<ul id="page" class="pagination">
<li class="page-item {{first2}}">
    <a class="page-link pageclickpre2" href="#ireview">&laquo;</a>
</li>
{{#each print2}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -->
    <li class="page-item {{this.Status}}">
        <a class="page-link pageclick2" href="#ireview">{{this.page}}</a>
    </li>
    {{/each}}
    <li class="page-item {{second2}}">
        <a class="page-link pageclickne2" href="#ireview">&raquo;</a>
    </li>
</ul>

```

```

</div>
</form>      </div>      </section>      </div>

```

reviews.js

```

/* 라우팅 함수 선언[reviews] */

/* mongoose 모듈 참조 */
const mongoose = require('mongoose');
//mongoose.Promise = Promise;

/* Model 불러오기*/
const User = mongoose.model('User');
const Review = mongoose.model('Review');
const Upload = mongoose.model('Upload');
const only = require('only');//object중 원하는 데이터만 sorting하여 리턴하는 helper 모듈

var status = new Object();
status={"print":"메인화면", "link":"location.href = '/'"};

console.log('call : /controllers/reviews.js');

//도서 리뷰 화면
exports.review = function (req, res) {
  const isLogin = req.isAuthenticated();
  Review.count(function(totalCount) {//저장된 전체 리뷰 개수
    var page = parseInt(req.body.pageNo);//보고싶은 리뷰 페이지 번호
    if(!page) {
      page = 1;//페이지 번호가 지정되지 않았다면 1페이지를 보여줌.
    }
    var pageArray = setpage(totalCount, page);//총 페이지 개수
    var print = setprint(pageArray, page);//현재 페이지 정보=active
    var last5 = (parseInt(totalCount/50)*5);//전체 페이지를 5 단위로 나누었을때 나오는 마지막 페이지
    첫 시작 페이지

    if(page <= 5) {//만약 현재 페이지가 5페이지 이하라면 이전페이지로 넘어가는 버튼 비활성화.
      var first1 = "disabled";
      var second = "";

      if(last5 < 5){//만약 현재 페이지가 5페이지 이하이고, 마지막 페이지도 5페이지를 넘지 않는다면 다음페이지로 넘어가는 버튼 역시 비활성화.

```

```

        var first1 = "disabled";
        var second = "disabled";
    }
} else if(page > last5) { //현재 페이지가 마지막 첫 페이지 이상이라면 다음 페이지 버튼 비활성화.
    var first1 = "";
    var second = "disabled";
} else { //모두 아니라면 이전페이지, 다음 페이지 버튼 모두 활성화.
    var first1 = "";
    var second = "";
}
}

Review.list10(page, function (reviews) { //전체 리뷰를 현재 페이지에 들어갈 내용들로만 10개 가져오기.

    if(isLogin) {
        User.load(req.user.email, function (user) {
            Review.mycount(req.user.email, function(mytotalCount) { //저장된 내가 작성한 리뷰 개수
                var last52 = (parseInt(mytotalCount/50)*5);
                var page2 = parseInt(req.body.pageNo2);
                if(!page2) {
                    page2 = 1;
                }
                var pageArray2 = setpage(mytotalCount, page2);
                var print2 = setprint(pageArray2, page2);
                if(page2 <= 5) {
                    var first2 = "disabled";
                    var second2 = "";
                    if(last52 < 5){
                        var first2 = "disabled";
                        var second2 = "disabled";
                    }
                }
                else if(page2 > last52) {
                    var first2 = "";
                    var second2 = "disabled";
                }
                else {
                    var first2 = "";
                    var second2 = "";
                }
            }
        }
    }

    //내가 작성한 리뷰를 현재 페이지에 들어갈 내용들로만 10개 가져오기.

```



```

        Review.mylist10(req.user.email, page2, function (myreviews) {
            res.render('reviews/review', {
                reviews: reviews, isUserLoggedIn: isLogin, user: user, status:status,
myreviews:myreviews, print: print, pageNo: page, second: second, first1: first1, print2: print2,
pageNo2: page2, first2: first2, second2: second2
            });
        });
    });
});
} else {
    res.render('reviews/review', {
        reviews: reviews, isUserLoggedIn: isLogin, print: print, pageNo: page, first: first, second:
second
    });
}
});
});
};

function setpage(totalCount, page) {
    var last5 = (parseInt(totalCount/50)*5);
    var last = (parseInt((totalCount-1)/10)+ 1);

    if(page <= last5) {
        var defaultpage = (parseInt((page-1)/5)*5+ 1);
        var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2, defaultpage+ 3, defaultpage+ 4 ];
    } else {
        var etc = last-last5;
        var defaultpage = (parseInt((page-1)/5)*5+ 1);
        switch(etc) {
            case 1:
                var pageArray = [ defaultpage];
                break;
            case 2:
                var pageArray = [ defaultpage, defaultpage+ 1];
                break;
            case 3:
                var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2];
                break;

```

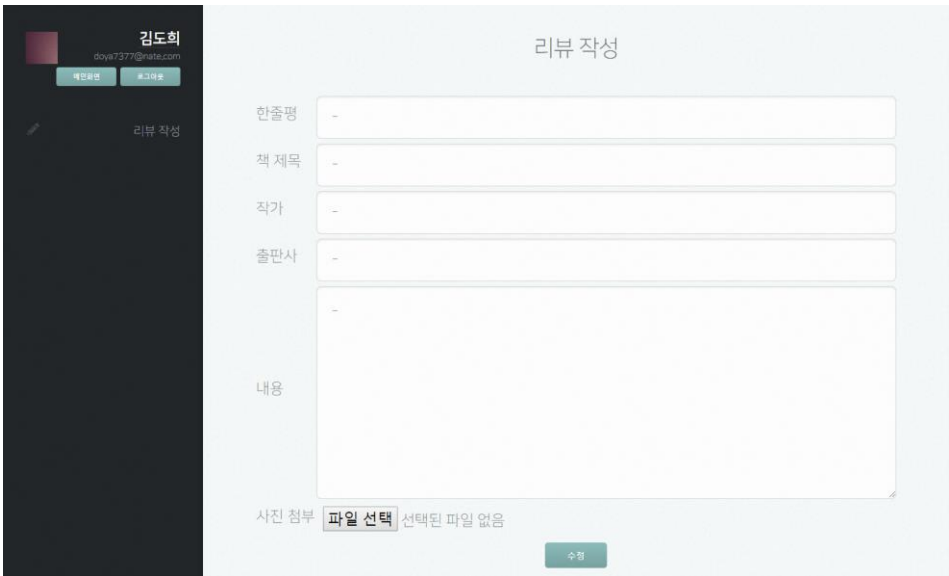
```

    case 4:
        var pageArray = [ defaultpage, defaultpage+ 1, defaultpage+ 2, defaultpage+ 3];
        break;
    }
}
return pageArray;
}

function setprint(pageArray, page){
    var print = new Object();
    for(var i=0; i<pageArray.length; i+ ) {
        if(page == pageArray[i]) {
            print[i] = { "page": pageArray[i], "Status": "active" };
        } else {
            print[i] = { "page": pageArray[i], "Status": "" };
        }
    }
    return print;
}

```

■ 화면10

화면10	설명
	<p>-리뷰 작성. 이미지 첨부 가능.</p> <p>-이미지 첨부는 필수사항은 아님.</p>
소스코드	
<pre> writerreview.hbs <nav id="nav"> 리뷰 작성</nav> </div> </div>div id="main"> <section id="top" class="five dark cover"> <div class="container"> </pre>	

```

<section id="review" class="two">
    <div class="container">
<header>
        <h2>리뷰 작성</h2>
    </header>
    <div class="formsize">
        <form style="" method="post" action="/storereview" class="form-horizontal"
enctype="multipart/form-data">
<div class="form-group">
<label for="inputEmail3" class="col-sm-2 control">한줄평</label>
        <input type="text" class="textcontrol" name="title">
    </div>
<div class="form-group">
        <label for="inputEmail3" class="control">책 제목</label>
        <input type="text" class="textcontrol" name="book">
    </div>
<div class="form-group">
        <label for="inputEmail3" class="control">작가</label>
        <input type="text" class="textcontrol" name="author">
    </div>
<div class="form-group">
        <label for="inputEmail3" class="control">출판사</label>
        <input type="text" class="textcontrol" name="publisher">
    </div>
        <div class="form-group">
            <label for="inputEmail3" class="control">내용</label>
<textarea class="textcontrol" name="content" placeholder=""></textarea>
            </div>
            <input type="hidden" class="textcontrol" name="user" value="{{user.email}}">
            <div class="form-group">
                <label for="inputEmail3" class="control">사진 첨부</label>
<input type="file" name="file" class="form-control-file" id="exampleInputFile" aria-
describedby="fileHelp">
            </div>
<div><button type="button" class="btn btn-submit">등록</button>
    </div></form></div></section></div>

```

reviews.js

//리뷰 작성 화면

```

exports.writereview = function (req, res) {
  const isLogin = req.isAuthenticated();
  if(isLogin) {
    User.load(req.user.email, function (user) {
      res.render('reviews/writereview', {
        isUserLoggedIn: isLogin, user: user, status:status
      });
    });
  } else {
    res.render('reviews/writereview', {
      isUserLoggedIn: isLogin
    });
  }
};

//리뷰 저장
exports.storereview = function (req, res) {
  //form 으로부터 한줄평, 책 제목, 저자, 출판사, 내용, 작성자 정보를 가져옴.
  const review = new Review(only(req.body, 'title book author publisher content user'));
  //review DB에 저장
  review.save(function (err, result) {
    if (err) {
      /* Client Validation도 무시한 후 데이터가 들어온 경우 400코드 전송*/
      res.sendStatus(400)
    }
    /* 업로드한 파일이 있으면 */
    if (req.files.length > 0) {
      /* 업로드한 파일을 확인하여 데이터베이스에 저장 */
      req.files.forEach(function (file) {
        //업로드 모델(upload) 생성
        const upload = new Upload({
          relatedId: result,
          type: "review_photo",//업로드 파일 구분 카테고리
          filename: file.filename,
          originalname: file.originalname,
          type: file.mimetype,
          size: file.size,
        });

```

```

        //데이터베이스에 사진 저장(uploads 컬렉션)
        upload.save(function (err, result) {
            review.photo = result;
            review.save();
        });
    });
}
res.redirect('/review'); //저장이 완료되면 review화면을 보여줌.
});
};

```

index.js

//리뷰 저장

```

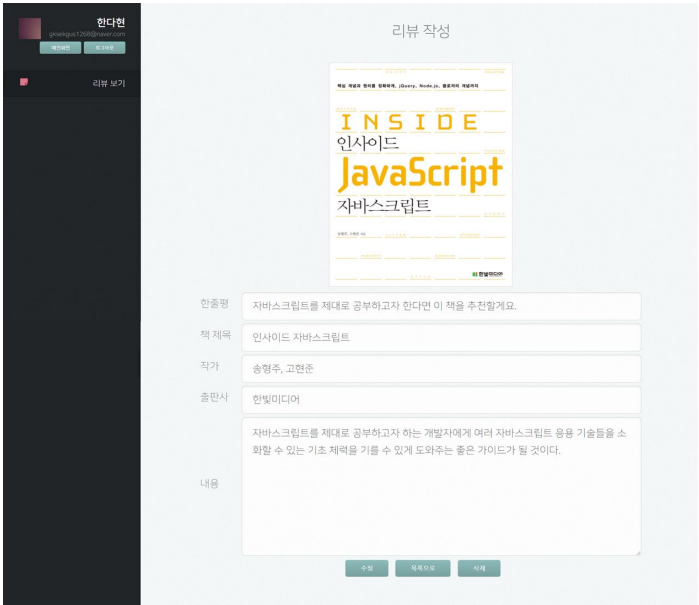

$(".btn-submit").click(function () {
    console.log("submit 들어옴");
    var title = $("input[name='title']").val();
    var book = $("input[name='book']").val();
    var author = $("input[name='author']").val();
    var publisher = $("input[name='publisher']").val();
    var content = $("textarea[name='content']").val();

    if(!title){
        swal('경고', '글 제목을 확인해주세요.', 'error')
    }else if(!book){
        swal('경고', '책 제목을 확인해주세요.', 'error')
    }else if(!author){
        swal('경고', '작가 이름을 확인해주세요.', 'error')
    }else if(!publisher){
        swal('경고', '출판사명을 확인해주세요.', 'error')
    }else if(!content){
        swal('경고', '책의 내용을 확인해주세요.', 'error')
    }else{
        $(this).closest("form").submit();
    }
});

```

■ 화면11

화면11	설명
------	----

 	<p>-작성된 리뷰를 클릭하면 리뷰의 상세 내용을 확인 할 수 있음.</p> <p>-내가 작성한 리뷰의 경우 수정과 삭제가 가능한 버튼이 함께 보임.</p> <p>-관리자 역시 삭제와 수정가능.</p> <p>-다른 사람이 작성한 리뷰의 경우 삭제 및 수정 불가능.(버튼이 보이지 않는다.)</p>
--	---

소스코드

readreview.hbs

```

<!-- Nav --><nav id="nav"><ul><li><a href="#read" id="portfolio-link"><span class="icon solid fa-sticky-note">리뷰 보기</span></a></li></ul></nav>
</div></div><div id="main">
<section id="top" class="five dark cover">
    <div class="container"></div>
</section>
<section id="read" class="two">
    <div class="container">
<header><h2>리뷰 작성</h2></header>
{{#if review.photo}}

    {{else}}
    
    {{/if}}

<div class="formsize">
    <form>
        <div class="form-group">

```

```

        <label for="inputEmail3" class="col-sm-2 control">한줄평</label>
<input type="text" class="textcontrol" name="title" placeholder="{{review.title}}" readonly>
    </div>
<div class="form-group">
<label for="inputEmail3" class="control">책 제목</label>
        <input type="text" class="textcontrol" name="book" placeholder="{{review.book}}"
readonly>
    </div>
<div class="form-group">
        <label for="inputEmail3" class="control">작가</label>
        <input type="text" class="textcontrol" name="author" placeholder="{{review.author}}"
readonly></div>
<div class="form-group">
<label for="inputEmail3" class="control">출판사</label>
<input type="text" class="textcontrol" name="publisher" placeholder="{{review.publisher}}"
readonly></div>
<div class="form-group">
<label for="inputEmail3" class="control">내용</label>
<textarea class="textcontrol" name="content" placeholder="{{review.content}}"
readonly></textarea></div>
<div class="action">
<button onclick="location.href ='/editreview/{{review._id}}'" class="buttonmargin {{showstatus}}"
type="button">수정</button>
<button onclick="location.href ='/review'" class="buttonmargin" type="button">목록으로</button>
<button data-id="{{review._id}}" class="buttonmargin btn-delete {{showstatus}}" type="button">삭제
</button></div></form></div></div></section></div>

```

reviews.js

// 리뷰 수정 화면

```

exports.editreview = function (req, res) {
    const isLogin = req.isAuthenticated();
    Review.load(req.params.id, function (review) {
        if(isLogin) {
            User.load(req.user.email, function (user) {
                res.render('reviews/editreview', {
                    review: review, isUserLoggedIn: isLogin, user: user, status:status
                });
            });
        } else {
            res.render('reviews/editreview', {
                review: review, isUserLoggedIn: isLogin
            });
        }
    });
}

```

```

    });
  }
});
};

//리뷰가 수정이 되면 업데이트
exports.updatereview = function (req, res) {
  Review.load(req.body.id, function (review) {
    //필요한 정보를 가져와서 저장
    review.title = req.body.title;
    review.book = req.body.book;
    review.author = req.body.author;
    review.publisher = req.body.publisher;
    review.content = req.body.content;

    //리뷰 저장
    review.save(function (err, result) {
      if (err) {
        /* Client Validation도 무시한 후 데이터가 들어온 경우 400코드 전송*/
        res.sendStatus(400)
      }
      /* 업로드한 파일이 있으면 */
      if (req.files.length > 0) {
        /* 업로드한 파일을 확인하여 데이터베이스에 저장 */
        req.files.forEach(function (file) {
          //업로드 모델(upload) 생성
          const upload = new Upload({
            relatedId: result,
            type: "rreview_photo",
            filename: file.filename,
            originalname: file.originalname,
            type: file.mimetype,
            size: file.size,
          });
          //데이터베이스에 사진 저장(uploads 컬렉션)
          upload.save(function (err, result) {
            review.photo = result;
            review.save();
          });
        });
      }
    });
  });
};

```



```

        });
    });
}
res.redirect('/review');
})
});
};

// 리뷰 삭제
exports.deleteReview = function (req, res) {
    //Review DB에서 리뷰 삭제
    Review.remove({
        _id: req.body.id
    }, function (err, result) {
        if (err) return res.send(err);
        res.sendStatus(200)
    });
};

```

index.js

```

//리뷰 삭제
$("#btn-delete").click(function () {
    var _id = $(this).data("id");
    swal({
        title: '경고',
        text: "정말 지우시겠습니까?",
        type: 'warning',
        showCancelButton: true,
        confirmButtonColor: '#3085d6',
        cancelButtonColor: '#d33',
        confirmButtonText: '네 삭제하겠습니다',
        cancelButtonText: '아니요'
    }).then(function () {
        $.ajax({
            url: '/deletereview',
            type: 'post',
            data: {id: _id},
            success: function (response) {

```

```

        location.href = '/review';

    }

});

})

}

```

■ 화면12

화면12	설명
	<p>- 친구구하기 화면에서는 자유로운 게시판 글 게시를 통해 사용자들끼리의 만남, 소통을 가능하게 함</p> <p>-혼밥싫어요: 도서관에서 밥을 함께 먹을 친구를 구할 수 있음</p> <p>-혼공싫어요: 도서관에서 공부를 함께 할 친구를 구할 수 있음</p> <p>-스터디해요: 도서관에서 스터디를 함께 할 친구를 구할 수 있음</p> <p>-바로가기 버튼을 클릭하면 해당 게시판으로 이동</p>

소스코드

index.hbs

```

<section id="contact" class="four">
  <div class="container">
    <header>
      <h2>친구 구하기</h2>
    </header>
    <div class="row">
      <div class="span3">
        <div class="home-post">
          <div class="post-image">
            
          </div><div class="post-meta">
            <span class="date">혼밥 싫어요</span>
          </div>
          <div class="entry-content">
            <p>혼자 밥 먹기 싫은 <b>오늘</b>, 새로운 친구와 함께 밥을 먹어보는건 어떨까요?<br><br>이곳에서
            오늘의 <b>밥친구</b>를 찾아보세요! p>
          </div>
          <a href="/friends#food" class="more">바로가기</a></div></div>
      <div class="span3">

```

```

        <div class="home-post">
            <div class="post-image">

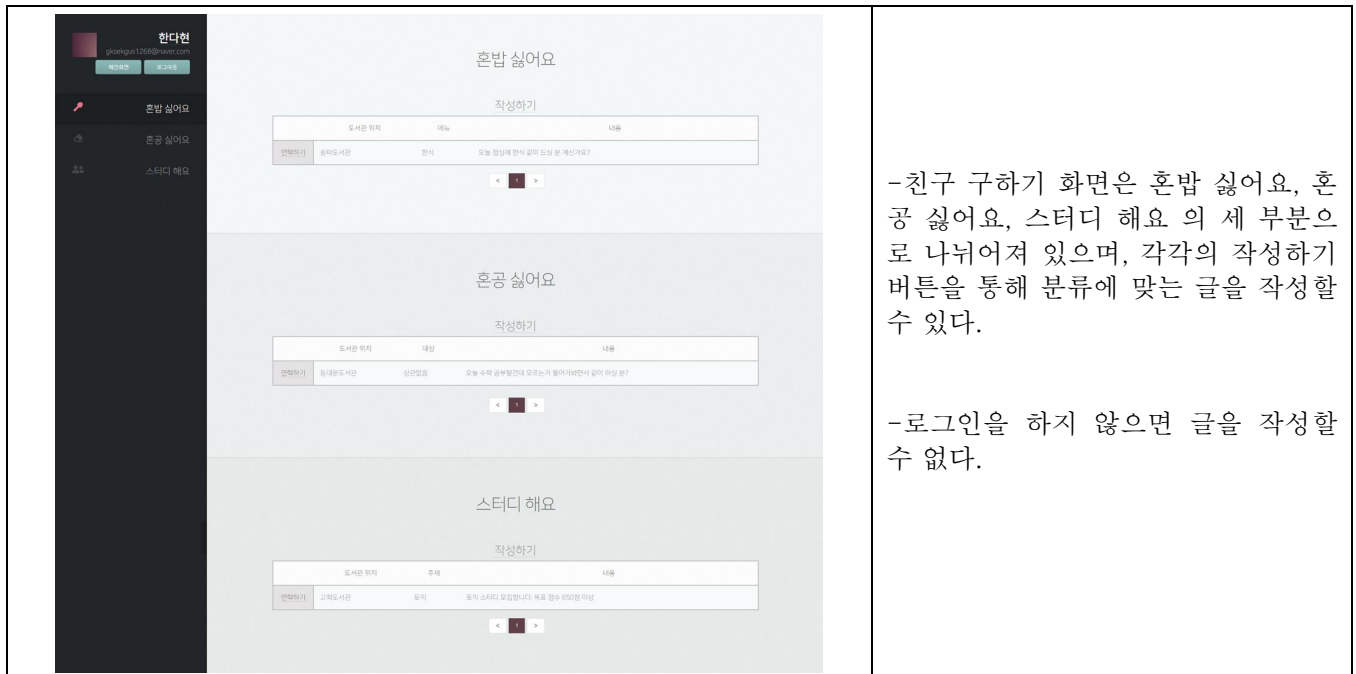
            </div>
            <div class="post-meta">
                <span class="date">혼공 싫어요</span>
            </div>
            <div class="entry-content">
<p>혼자 공부하는것보다 옆에 <b>누군가</b>가 있어야 공부가 더 잘되시나요?<br><br>이곳에서 오
날의 <b>공부친구</b>를 찾아보세요! </p>
            </div>
            <a href="/friends#study" class="more">바로가기</a>
        </div>
</div>
<div class="span3">
    <div class="home-post" >
        <div class="post-image">

        </div>
        <div class="post-meta">
            <span class="date">스터디 해요</span>
        </div>
        <div class="entry-content">
<p>다같이 모여서 목표를 세우고 함께 공부를 하며 <b>성취감</b>을 높여보는건 어떨까요?<br><br>
이곳에서 <b>스터디 친구</b>를 찾아보세요! </p>
        </div><a href="/friends#groupstudy" class="more">바로가기</a></div>
    </div>
</div>
</section>

```

■ 화면13

화면13	설명
------	----



-친구 구하기 화면은 혼밥 싫어요, 혼공 싫어요, 스터디 해요 의 세 부분으로 나뉘어져 있으며, 각각의 작성하기 버튼을 통해 분류에 맞는 글을 작성할 수 있다.

-로그인을 하지 않으면 글을 작성할 수 없다.

소스코드

friends.hbs

```
<nav id="nav">
<ul>
<li><a href="#food" id="portfolio-link"><span class="icon solid fa-utensil-spoon">혼밥 싫어요</span></a></li>
<li><a href="#study" id="about-link"><span class="icon solid fa-eraser">혼공 싫어요</span></a></li>
<li><a href="#groupstudy" id="about-link"><span class="icon solid fa-user-friends">스터디 해요</span></a></li></ul></nav></div></div>

<!-- Main -->
<div id="main">
<!-- Intro -->
<section id="top" class="five dark cover">
<div class="container">
</div>
</section>

<section id="food" class="two">
<div class="container">
<header>
<h2>혼밥 싫어요</h2>
</header>
<a href="/writefood">작성하기</a>
<table class="table table-hover">
<thead>
<tr>
<th scope="col"></th>
```

```

    <th scope="col">도서관 위치</th>
    <th scope="col">메뉴</th>
    <th scope="col">내용</th>
  </tr>
</thead>
<tbody>
  {{#each foods}}
<th class="showsize" scope="row"><a href="/foodmessage/{{this.user}}">연락하기</a></th>
    <td>{{this.location}}</td>
    <td>{{this.food}}</td>
    <td>{{this.content}}</td>
  </tr>
  {{/each}}
</tbody>
</table>

<form method="post" action="/friends">
  <input type="hidden" name="pageNo2" value="{{pageNo2}}">
  <input type="hidden" name="pageNo" value="{{pageNo}}">
  <input type="hidden" name="pageNo2" value="{{pageNo3}}">
  <div>
    <ul id="page" class="pagination">
      <li class="page-item {{first1}}">
        <a class="page-link pageclickpre" href="#food">&laquo;</a>
      </li>
      {{#each print}}
        <li class="page-item {{this.Status}}">
          <a class="page-link pageclick" href="#food">{{this.page}}</a>
        </li>
      {{/each}}
      <li class="page-item {{second}}">
        <a class="page-link pageclickne" href="#food">&raquo;</a>
      </li>
    </ul>
  </div>
</form></div></section>

<section id="study" class="three">    <div class="container">

```

```

        <header>
            <h2>혼공 싫어요</h2>
        </header>
        <a href="/writestudy">작성하기</a>
        <table class="table table-hover">
<thead>
    <tr>
        <th scope="col"></th>
        <th scope="col">도서관 위치</th>
        <th scope="col">대상</th>
        <th scope="col">내용</th>
    </tr>
</thead>
<tbody>
    {{#each studys}}
<tr class="table-odd">
<th class="showsize" scope="row"><a href="/studymessage/{{this.user}}">연락하기</a></th>

        <td>{{this.location}}</td>
        <td>{{this.target}}</td>
        <td>{{this.content}}</td>

    </tr>
    {{/each}}
</tbody>
</table>

<form method="post" action="/friends">
    <input type="hidden" name="pageNo" value="{{pageNo}}">
    <input type="hidden" name="pageNo2" value="{{pageNo2}}">
    <input type="hidden" name="pageNo2" value="{{pageNo3}}">
    <div>
<ul id="page" class="pagination">
<li class="page-item {{first2}}">
    <a class="page-link pageclickpre2" href="#ireview">&laquo;</a>
</li>
    {{#each print2}}
    <li class="page-item {{this.Status}}">
        <a class="page-link pageclick2" href="#ireview">{{this.page}}</a>

```

```

</li>
{{/each}}

<li class="page-item {{second2}}">
  <a class="page-link pageclickne2" href="#ireview">&raquo;</a>
</li>
</ul>
</div>
</form>
</div>

</section>

<section id="groupstudy" class="four">
  <div class="container">

    <header>
      <h2>스터디 해요</h2>
    </header>

    <a href="/writegroupstudy">작성하기</a>

    <table class="table table-hover">
<thead>
  <tr>
    <th scope="col"></th>
    <th scope="col">도서관 위치</th>
    <th scope="col">주제</th>
    <th scope="col">내용</th>
  </tr>
</thead>
<tbody>
  {{#each groupstudys}}
    <tr class="table-odd">
      <th class="showsize" scope="row"><a href="/groupstudymessage/{{this.user}}">연락하
기</a></th>

      <td>{{this.location}}</td>
      <td>{{this.subject}}</td>
      <td>{{this.content}}</td>
    </tr>
  {{/each}}
</tbody>
</table>

```

```

<form method="post" action="/friends">
    <input type="hidden" name="pageNo" value="{{pageNo}}">
    <input type="hidden" name="pageNo2" value="{{pageNo2}}">
    <input type="hidden" name="pageNo3" value="{{pageNo3}}">
    <div>
        <ul id="page" class="pagination">
            <li class="page-item {{first3}}">
                <a class="page-link pageclickpre3" href="#ireview">&laquo;</a>
            </li>
            {{#each print3}}
                <li class="page-item {{this.Status}}">
                    <a class="page-link pageclick3" href="#ireview">{{this.page}}</a>
                </li>
            {{/each}}

            <li class="page-item {{second3}}">
                <a class="page-link pageclickne3" href="#ireview">&raquo;</a>
            </li>
        </ul></div> </form></div></section></div>

```

friends.js

```

/* 라우팅 함수 선언[friends] */
/* mongoose 모듈 참조 */
const mongoose = require('mongoose');

/* Model 불러오기*/
const User = mongoose.model('User');
const Food = mongoose.model('Food');
const Groupstudy = mongoose.model('Groupstudy');
const Study = mongoose.model('Study');
const Messages = mongoose.model('Messages');
const only = require('only');

var status = new Object();
status={"print":"메인 화면", "link":"location.href = '/'"};

console.log('call : /controllers/friends.js');

```


//friends 요청 시 Food, Study, GroupStudy 모델에서 정보를 가져와서 화면에 보여줌.

```
exports.friends = function (req, res) {
  const isLogin = req.isAuthenticated();
  Food.count(function(totalCount) {
    var page = parseInt(req.body.pageNo);

    //body에 pageNo 정보가 없는 경우는 정보의 첫번째 페이지 이므로 1로 세팅.
    if(!page) {
      page = 1;
    }
    var pageArray = setpage(totalCount, page);
    var print = setprint(pageArray, page);
    var last5 = (parseInt(totalCount/50)*5);

    //페이지 넘버를 5개씩 스킵하는 화살표 비활성화 여부 판단
    if(page <= 5) {
      var first1 = "disabled";
      var second = "";
      if(last5 < 5){
        var first1 = "disabled";
        var second = "disabled";
      }
    } else if(page > last5) {
      var first1 = "";
      var second = "disabled";
    } else {
      var first1 = "";
      var second = "";
    }

    Food.list10(page, function (foods) {
      Study.count(function(totalCount) {
        var page2 = parseInt(req.body.pageNo2);
        if(!page2) {
          page2 = 1;
        }

        var pageArray2 = setpage(totalCount, page2);
```

```
var print2 = setprint(pageArray2, page2);
var last52 = (parseInt(totalCount/50)*5);

if(page2 <= 5) {
    var first2 = "disabled";
    var second2 = "";
    if(last52 < 5){
        var first2 = "disabled";
        var second2 = "disabled";
    }
} else if(page2 > last52) {
    var first2 = "";
    var second2 = "disabled";
} else {
    var first2 = "";
    var second2 = "";
}

Study.list10(page2, function (studys) {
    Groupstudy.count(function(totalCount) {
        var page3 = parseInt(req.body.pageNo3);
        if(!page3) {
            page3 = 1;
        }
        var pageArray3 = setpage(totalCount, page3);
        var print3 = setprint(pageArray3, page3);
        var last53 = (parseInt(totalCount/50)*5);

        if(page3 <= 5) {
            var first3 = "disabled";
            var second3 = "";
            if(last53 < 5){
                var first3 = "disabled";
                var second3 = "disabled";
            }
        }

        else if(page3 > last52) {
```

```




        var first3 = "";
        var second3 = "disabled";
    }
    else {
        var first3 = "";
        var second3 = "";
    }

    Groupstudy.list10(page3, function (groupstudys) {
        if(isLogin) {
            User.load(req.user.email, function (user) {
                res.render('friends/friends', {
                    foods: foods, studys: studys, groupstudys: groupstudys, isUserLoggedIn:
isLogin, user: user, status:status, print: print, pageNo: page, second: second, first1: first1, print2:
print2, pageNo2: page2, first2: first2, second2: second2, pageNo3: page3, first3: first3, second3:
second3, print3: print3, pageNo3: page3
                });
            });
        } else {
            res.render('friends/friends', {
                foods: foods, studys: studys, groupstudys: groupstudys, isUserLoggedIn:
isLogin, status:status, print: print, pageNo: page, second: second, first1: first1, print2: print2,
pageNo2: page2, first2: first2, second2: second2, pageNo3: page3, first3: first3, second3: second3,
print3: print3, pageNo3: page3
            });
        }
    });
});
});
});
});
});
});
};

```

■ 화면14

화면14	설명
------	----

	<p>-밥 메이트, 공부 메이트, 스터디 글은 기본적으로 도서관의 위치와 내용을 기본적으로 가지고 있다.</p>
	<p>-밥 메이트는 음식 종류를 입력하는 칸이 존재한다.</p> <p>-공부 메이트는 같이 공부하고 싶은 사람의 기준을 작성하는 칸이 존재한다.</p>
	<p>-스터디는 스터디의 주제를 작성하는 칸이 존재한다.</p>

소스코드

writefood.hbs

```

<nav id="nav">
  <ul>
    <li><a href="#food" id="portfolio-link"><span class="icon solid fa-utensil-spoon">밥 메이트 구함
  </span></a></li></ul></nav></div></div>

  <div id="main">
    <section id="top" class="five dark cover">
      <div class="container">
        </div>
      </section>

    <section id="food" class="two">
      <div class="container">
        <header>
          <h2>밥 메이트 구함</h2>
        </header>
        <div class="formsize">
          <form style="" method="post" action="/storefood" class="form-horizontal">
            <div class="form-group">
              <label for="inputEmail3" class="col-sm-2 control">도서관</label>

```

```

<input type="text" class="textcontrol" id="location" name="location">
    </div>
    <div class="form-group">
        <label for="inputEmail3" class="col-sm-2 control">메뉴</label>
        <input type="text" class="textcontrol" id="food" name="food">
    </div>
    <div class="form-group">
        <label for="inputEmail3" class="control">내용</label>
        <input type="text" class="textcontrol" id="content" name="content">
    </div>
    <input type="hidden" class="textcontrol" name="user" value="{{user.email}}">
    <div>
        <button type="button" class="btn-food-submit">등록</button>
    </div>form></div></div></section></div>

```

writestudy.hbs

```

    <nav id="nav">
        <ul>
            <li><a href="#food" id="portfolio-link"><span class="icon solid fa-eraser">공부 메이트 구함
            </span></a></li></ul></nav></div></div>
    <div id="main">
        <section id="top" class="five dark cover">
            <div class="container">
                </div>
            </section>
            <section id="food" class="two">
                <div class="container">
                    <header>
                        <h2>공부 메이트 구함</h2>
                    </header>
                    <div class="formsize">
                        <form style="" method="post" action="/storestudy" class="form-horizontal">
                            <div class="form-group">
                                <label for="inputEmail3" class="col-sm-2 control">도서관</label>
                                <input type="text" class="textcontrol" name="location">
                            </div>
                            <div class="form-group">
                                <label for="inputEmail3" class="col-sm-2 control">대상</label>
                                <input type="text" class="textcontrol" name="target">

```

```

</div>
    <div class="form-group">
        <label for="inputEmail3" class="control">내용</label>
        <input type="text" class="textcontrol" name="content">
    </div>

    <input type="hidden" class="textcontrol" name="user" value="{{user.email}}"><div>
        <button type="button" class="btn btn-study-submit">등록</button>
</div></form></div></div></section></div>

```

writegroupstudy.hbs

```

<nav id="nav">
<ul><li><a href="#food" id="portfolio-link"><span class="icon solid fa-user-friends">스터디 구함
</span></a></li></ul></nav></div></div>

    <div id="main">
<section id="top" class="five dark cover">
    <div class="container">
    </div></section>
<section id="food" class="two">
    <div class="container">
        <header>
            <h2>스터디 구함</h2>
        </header>
        <div class="formsize">
            <form style="" method="post" action="/storegroupstudy" class="form-horizontal">
                <div class="form-group">
                    <label for="inputEmail3" class="col-sm-2 control">도서관</label>
                    <input type="text" class="textcontrol" name="location">
                </div>
                <div class="form-group">
                    <label for="inputEmail3" class="col-sm-2 control">주제</label>
                    <input type="text" class="textcontrol" name="subject">
                </div>
                <div class="form-group">
                    <label for="inputEmail3" class="control">내용</label>
                    <input type="text" class="textcontrol" name="content">
                </div>
                <input type="hidden" class="textcontrol" name="user" value="{{user.email}}">
            <div>
                <button type="button" class="btn btn-groupstudy-submit">등록</button>
            </div>

```

```
</div></form></div></section></div>
```

friends.js

```
//혼밥 메이트 구하는 글 작성
```

```
exports.writefood = function (req, res) {  
  const isLogin = req.isAuthenticated();  
  if(isLogin) {  
    User.load(req.user.email, function (user) {  
      res.render('friends/writefood', {  
        isUserLoggedIn: isLogin, user: user, status:status  
      });  
    });  
  } else {  
    res.render('friends/writefood', {  
      isUserLoggedIn: isLogin  
    });  
  }  
};
```

```
//공부 메이트 구하는 글 작성
```

```
exports.writestudy = function (req, res) {  
  const isLogin = req.isAuthenticated();  
  if(isLogin) {  
    User.load(req.user.email, function (user) {  
      res.render('friends/writestudy', {  
        isUserLoggedIn: isLogin, user: user, status:status  
      });  
    });  
  } else {  
    res.render('friends/writestudy', {  
      isUserLoggedIn: isLogin  
    });  
  }  
};
```

```
//스터디 메이트 구하는 글 작성
```

```
exports.writegroupstudy = function (req, res) {  
  const isLogin = req.isAuthenticated();
```

```

if(isLogin) {
  User.load(req.user.email, function (user) {
    res.render('friends/writegroupstudy', {
      isUserLoggedIn: isLogin, user: user, status:status
    });
  });
} else {
  res.render('friends/writegroupstudy', {
    isUserLoggedIn: isLogin
  });
}
};

//혼밥 메이트 글 DB에 저장
exports.storefood = function (req, res) {
  //form 으로부터 도서관 위치, 음식 종류, 내용, 작성자 정보를 가져와 객체 생성
  const food = new Food(only(req.body, 'location food content user'));
  food.save(function (err, result) { //생성된 객체를 DB에 저장
    if (err) {
      res.sendStatus(400)
    }
    res.redirect('/friends#food');//저장이 완료 되면 혼밥 글 목록을 보여줌
  });
};

//공부 메이트 글 DB에 저장
exports.storestudy = function (req, res) {
  //form 으로부터 도서관 위치, 공부 대상, 내용, 작성자 정보를 가져와 객체 생성
  const study = new Study(only(req.body, 'location target content user'));
  study.save(function (err, result) { //생성된 객체를 DB에 저장
    if (err) {
      res.sendStatus(400)
    }
    res.redirect('/friends#study');//저장이 완료 되면 공부 글 목록을 보여줌
  });
};

//스터디 모집 글 DB에 저장

```



```

exports.storegroupstudy = function (req, res) {
  //form 으로부터 도서관 위치, 스터디 주제, 내용, 작성자 정보를 가져와 객체 생성
  const groupstudy = new Groupstudy(only(req.body, 'location subject content user'));
  groupstudy.save(function (err, result) { //생성된 객체를 DB에 저장
    if (err) {
      res.sendStatus(400)
    }
    res.redirect('/friends#groupstudy');//저장이 완료 되면 스터디 글 목록을 보여줌
  });
};

```

■ 화면15

화면15	설명
	<p>-연락하기 버튼을 통해서 글 작성자에게 쪽지를 보낼 수 있다.</p>

소스코드

message.hbs

```

<nav id="nav"><ul><li><a href="#review" id="portfolio-link"><span class="icon solid fa-pencil-alt">
쪽지 보내기</span></a></li></ul></nav>

</div>

</div>

<div id="main">
  <section id="top" class="five dark cover">
    <div class="container">
      </div>
    </section>

    <section id="review" class="two">
      <div class="container">
        <header>
          <h2>쪽지 보내기</h2>
        </header>
        <div class="formsize">
          <form style="" method="post" action="/sendmessage" class="form-horizontal">

```

```

<div>
  <label for="inputEmail3" class="control">내용</label>
  <textarea class="textcontrol" name="content" placeholder=""></textarea>
  </div>
  <input type="hidden" class="textcontrol" name="receiver" value="{{db.user}}">
  <input type="hidden" class="textcontrol" name="sender" value="{{user.email}}">
</div><p></p>
  <button type="button" class="btn btn-send">보내기</button>
</div> </form></div></div> </section></div>

```

friends.js

//혼밥 메이트 관련 쪽지

```

exports.foodmessage = function (req, res) {
  const isLogin = req.isAuthenticated();
  if(isLogin) {
    Food.load(req.params.email, function (food) { //쪽지를 받는 사람 정보가 필요함.
      User.load(req.user.email, function (user) { //쪽지를 보내는 사람 정보가 필요함.
        res.render('friends/message', {
          isUserLoggedIn: isLogin, user: user, status:status, db:food
        });
      });
    });
  } else {
    req.flash('message', "로그인이 필요합니다."); //쪽지 보내기는 로그인 되어 있는 경우에만 사용 가능.
    res.redirect('/login');
  }
};

```

//공부 메이트 관련 쪽지

```

exports.studymessage = function (req, res) {
  const isLogin = req.isAuthenticated();
  if(isLogin) {
    Study.load(req.params.email, function (study) {
      User.load(req.user.email, function (user) {
        res.render('friends/message', {
          isUserLoggedIn: isLogin, user: user, status:status, db:study
        });
      });
    });
  }
};

```

```

    } else {
        req.flash('message', "로그인이 필요합니다.");
        res.redirect('/login');
    }
};

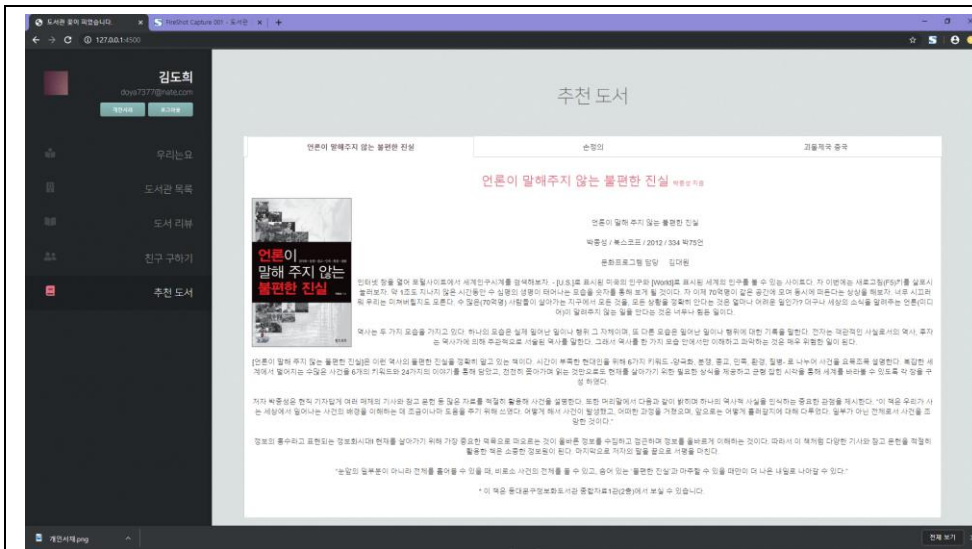
//스터디 관련 쪽지
exports.groupstudymessage = function (req, res) {
    const isLogin = req.isAuthenticated();
    if(isLogin) {
        Groupstudy.load(req.params.email, function (groupstudy) {
            User.load(req.user.email, function (user) {
                res.render('friends/message', {
                    isUserLoggedIn: isLogin, user: user, status:status, db:groupstudy
                });
            });
        });
    } else {
        req.flash('message', "로그인이 필요합니다.");
        res.redirect('/login');
    }
};

//쪽지 보내기를 실행하면 쪽지 내용, 수신자, 발신자를 DB에 저장한다.
exports.sendmessage = function (req, res) {
    const messages = new Messages(only(req.body, 'content receiver sender'));
    messages.save(function (err, result) {
        if (err) {
            res.sendStatus(400)
        }
        res.redirect('/friends');
    });
};

```

■ 화면16

화면16	설명
------	----



- 추천도서 화면에서는 오픈 API를 통해 추천도서의 제목, 작가, 내용요약을 보여줌

- 웹 방문시마다 매번 다른 추천도서를 표시함

소스코드

index.hbs

```
<section id="recommend" class="six">
  <div class="container">
    <header>
      <h2>추천 도서</h2>
    </header>
    <div class="sixstyle">
      <div class="tab-menu">
        <!-- Nav tabs -->
        <ul class="nav nav-tabs" role="tablist">
          <li class="active">
            <a href="#p-view-1" role="tab" data-toggle="tab" id="bookname0"></a>
          </li>
          <li>
            <a href="#p-view-2" role="tab" data-toggle="tab" id="bookname1"></a>
          </li>
          <li>
            <a href="#p-view-3" role="tab" data-toggle="tab" id="bookname2"></a>
          </li>
        </ul>
      </div>
      <div class="tab-content">
        <div class="tab-pane active" id="p-view-1">
          <div class="tab-inner">
            <div class="event-content head-team">
              <h4 id="author0"><span></span></h4>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

```

        <p id="Area0"></p>
    </div>
</div>
</div>
<div class="tab-pane" id="p-view-2">
    <div class="tab-inner">
        <div class="event-content head-team">
            <h4 id="author1"><span></span></h4>
            <p id="Area1"></p>
        </div>
    </div>
</div>
<div class="tab-pane" id="p-view-3">
    <div class="tab-inner">
        <div class="event-content head-team">
            <h4 id = "author2"><span></span></h4>
            <p id="Area2"></p>
        </div>
    </div>
</div>
</div></section></div>

```

recommendapi.js

```

$(document).ready(function() {

    var result = Math.floor(Math.random() * 1363) + 1;

    var newurl =
"http://openapi.seoul.go.kr:8088/534d6b4767676b73313239724152544e/json/SeoulLibRecommendInfo/"
+ result+ "/" + (result+ 2)+ "/"

    console.log(newurl);

    //도서 추천 자료 가져오기
    fetch(newurl).then(function(response) {
        response.json().then((data) => {
            var content0 = data.SeoulLibRecommendInfo.row[0].CONTENT
            var content1 = data.SeoulLibRecommendInfo.row[1].CONTENT
            var content2 = data.SeoulLibRecommendInfo.row[2].CONTENT

```

```

var author0 = data.SeoulLibRecommendInfo.row[0].AUTHOR
var author1 = data.SeoulLibRecommendInfo.row[1].AUTHOR
var author2 = data.SeoulLibRecommendInfo.row[2].AUTHOR

var book0 = data.SeoulLibRecommendInfo.row[0].BOOK_NM
var book1 = data.SeoulLibRecommendInfo.row[1].BOOK_NM
var book2 = data.SeoulLibRecommendInfo.row[2].BOOK_NM

check = /[ㄱ-ㅎ|ㅏ-ㅣ|가-힣]/;

var lastcontent0 = "";
var lastcontent1 = "";
var lastcontent2 = "";
var content0Array = content0.split('>||<');
var content1Array = content1.split('>||<');
var content2Array = content2.split('>||<');

for(var i = 0; i < content0Array.length; i++) {
    if(check.test(content0Array[i])) {
        lastcontent0 += content0Array[i] + "<br><br><br>";
    }
}
for(var i = 0; i < content1Array.length; i++) {
    if(check.test(content1Array[i])) {
        lastcontent1 += content1Array[i] + "<br><br><br>";
    }
}
for(var i = 0; i < content2Array.length; i++) {
    if(check.test(content2Array[i])) {
        lastcontent2 += content2Array[i] + "<br><br><br>";
    }
}

var sliceStr0 = lastcontent0.slice(0,-4);
var sliceStr1 = lastcontent1.slice(0,-4);
var sliceStr2 = lastcontent2.slice(0,-4);

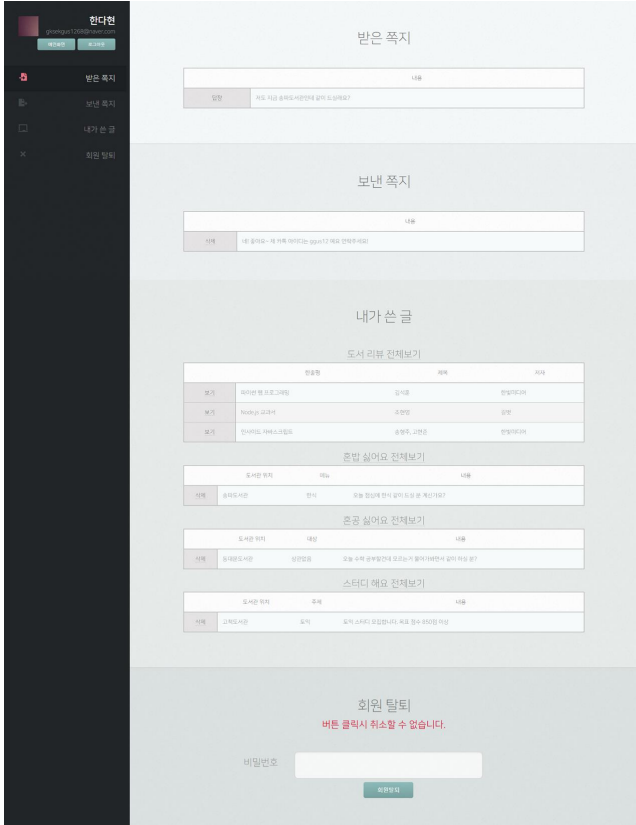
```

```
//가져온 자료 출력
$('#Area0').html(sliceStr0);
$('#Area1').html(sliceStr1);
$('#Area2').html(sliceStr2);

$('#bookname0').html(book0);
$('#bookname1').html(book1);
$('#bookname2').html(book2);

$('#author0').html(book0 + " <span class='spansaml'>" + author0 + " 지음</span>");
$('#author1').html(book1 + " <span class='spansaml'>" + author1 + " 지음</span>");
$('#author2').html(book2 + " <span class='spansaml'>" + author2 + " 지음</span>");
});
});
});
```

■ 화면17

화면17	설명
	<p>- 개인서재(마이페이지)화면에서는 받은 쪽지, 보낸 쪽지, 내가 쓴 글을 확인할 수 있음</p> <p>-삭제 가능</p> <p>-회원 탈퇴 가능</p>
소스코드	
<pre>mypage.hbs <nav id="nav"> 받은 쪽지</pre>	

```

</span></a></li>
<li><a href="#send" id="portfolio-link"><span class="icon solid fa-file-export">보낸 쪽지
</span></a></li>
<li><a href="#contact" id="about-link"><span class="icon solid fa-chalkboard">내가 쓴 글
</span></a></li>
<li><a href="#signout" id="contact-link"><span class="icon solid fa-times">회원 탈퇴</span></a></li>
</ul></nav></div></div>

<div id="main">
<section id="top" class="five dark cover">
<div class="container">
</div>
</section>

<section id="receive" class="two">
<div class="container">
<header>
<h2>받은 쪽지</h2>
</header>
<table class="table table-hover">
<thead>
<tr>
<th scope="col"></th>
<th scope="col">내용</th>
</tr>
</thead>
<tbody>
{{#each receivemessages}}
<tr class="table-odd">
<th scope="row"><a href="/reply/{{this.sender}}/{{this.receiver}}">답장</a></th>
<td>{{this.content}}</td>
</tr>
{{/each}}
</tbody></table></div></section>

<!-- About Me -->
<section id="send" class="three">
<div class="container">
<header>
<h2>보낸 쪽지</h2>
</header>
<section id="testimonials">

```



```

    <table class="table table-hover">
<thead>
    <tr>
        <th scope="col"></th>
        <th scope="col">내용</th>
    </tr>
</thead>
<tbody>
    {{#each sendmessages}}
        <tr class="table-odd">
            <th scope="row"><a data-id="{{this._id}}" type="button" class="send-delete" href="#send-
delete">삭제</a></th>
            <td>{{this.content}}</td>
        </tr>
    {{/each}}
</tbody>
</table>
</section>
</div>
</section>
<section id="contact" class="four">
    <div class="container">
        <header>
            <h2>내가 쓴 글</h2>
        </header>
        <a href="/review#review">도서 리뷰 전체보기</a>
        <table class="table table-hover">
<thead>
    <tr>
        <th scope="col"></th>
        <th scope="col">한줄평</th>
        <th scope="col">제목</th>
        <th scope="col">저자</th>
    </tr>
</thead>
<tbody>
    {{#each myreviews}}
        <tr class="table-odd">
            <th scope="row"><a href="/readreview/{{this._id}}">보기</a></th>

```

```

                <td>{{this.book}}</td>
                <td>{{this.author}}</td>
                <td>{{this.publisher}}</td>
            </tr>
        {{/each}}
    </tbody>
</table>

<a href="/friends#food">혼밥 싫어요 전체보기</a>
<table class="table table-hover">
    <thead>
        <tr>
            <th scope="col"></th>
            <th scope="col">도서관 위치</th>
            <th scope="col">메뉴</th>
            <th scope="col">내용</th>
        </tr>
    </thead>
    <tbody>
        {{#each myfoods}}
            <tr class="table-odd">
                <th class="showsize" scope="row"><a data-id="{{this._id}}" type="button"
class="food-delete" href="#food-delete">삭제</a></th>
                <td>{{this.location}}</td>
                <td>{{this.food}}</td>
                <td>{{this.content}}</td>
            </tr>
        {{/each}}
    </tbody>
</table>

<a href="/friends#study">혼공 싫어요 전체보기</a>
<table class="table table-hover">
    <thead>
        <tr>
            <th scope="col"></th>
            <th scope="col">도서관 위치</th>
            <th scope="col">대상</th>
            <th scope="col">내용</th>
        </tr>
    </thead>
    <tbody>
        {{#each myfoods}}
            <tr class="table-odd">
                <th class="showsize" scope="row"><a data-id="{{this._id}}" type="button"
class="study-delete" href="#study-delete">삭제</a></th>
                <td>{{this.location}}</td>
                <td>{{this.study}}</td>
                <td>{{this.content}}</td>
            </tr>
        {{/each}}
    </tbody>
</table>

```

```

</thead>
<tbody>
  {{#each mystudies}}
    <tr class="table-odd">
      <th class="showsize" scope="row"><a data-id="{{this._id}}" type="button"
class="study-delete" href="#study-delete">삭제</a></th>

      <td>{{this.location}}</td>
      <td>{{this.target}}</td>
      <td>{{this.content}}</td>
    </tr>
  {{/each}}
</tbody>
</table>

<a href="/friends#groupstudy">스터디 해요 전체보기</a>
<table class="table table-hover">
  <thead>
    <tr>
      <th scope="col"></th>
      <th scope="col">도서관 위치</th>
      <th scope="col">주제</th>
      <th scope="col">내용</th>
    </tr>
  </thead>
  <tbody>
    {{#each mygroups}}
      <tr class="table-odd">
        <th class="showsize" scope="row"><a data-id="{{this._id}}" type="button"
class="group-delete" href="#group-delete">삭제</a></th>

        <td>{{this.location}}</td>
        <td>{{this.subject}}</td>
        <td>{{this.content}}</td>
      </tr>
    {{/each}}
  </tbody>
</table>
</div>
</section>

```

```

        <section id="signout" class="six">
        <div class="container">
        <header>
<h2>회원 탈퇴</h2>
<span class="outred">버튼 클릭시 취소할 수 없습니다.</span>
        </header>
{{#each alert.message}}
        <div class="alert alert-success">
            {{this}}
        </div>
{{/each}}
{{#each alert.error}}
        <div class="alert alert-danger">
            {{this}}
        </div>
{{/each}}
<form method="post" action="/signout_user" class="formposition">
<input name="user_email" type="hidden" value="{{user.email}}"/>
        <div class="form-group row">
            <label for="example-date-input" class="">비밀번호</label>
            <div class="login-size">
                <input name="user_password" class="passportcontrol" type="password"/>
            </div>
        </div>
        <button type="submit" class="buttonposition">회원탈퇴</button>
</form></div>section>    </div>

```

main.js

//개인 서재 페이지

```

exports.mypage = function (req, res) {
    const isLogin = req.isAuthenticated();
    Messages.receiveList(req.user.email, function (receivemessages) { //받은 쪽지 목록
    Messages.sendList(req.user.email, function (sendmessages) { //보낸 쪽지 목록
        if(isLogin) {
            User.load(req.user.email, function (user) {
                Review.myList(req.user.email, function (myreviews) { //내가 쓴 리뷰 목록
                    Food.myFood(req.user.email, function (myfoods) { //내가 쓴 밥 메이트 글 목록
                        Study.myStudy(req.user.email, function (mystudies) { //내가 쓴 공부 메이트 글 목록

```

```
Groupstudy.mygroup(req.user.email, function (mygroups) { //내가 쓴 스터디 모집
글 목록

        res.render('main/mypage', {
            receivemessages:receivemessages,          sendmessages:sendmessages,
isUserLoggedIn: isLogin, user: user, status:status, myreviews:myreviews, myfoods:myfoods,
mystudies:mystudies, mygroups:mygroups, alert: req.flash()
        });
    });
});
});
});
});
});
});
}
});
});
};

exports.deletefood = function (req, res) {
    // deletefood 컬렉션에서 해당 documents(레코드) 삭제
    Food.remove({
        _id: req.body.id
    }, function (err, result) {
        if (err) return res.send(err);
        res.sendStatus(200)
    });
};

exports.deletestudy = function (req, res) {
    // deletestudy 컬렉션에서 해당 documents(레코드) 삭제
    Study.remove({
        _id: req.body.id
    }, function (err, result) {
        if (err) return res.send(err);
        res.sendStatus(200)
    });
};

exports.deletegroup = function (req, res) {
    // deletegroup 컬렉션에서 해당 documents(레코드) 삭제
```

```

Groupstudy.remove({
  _id: req.body.id
}, function (err, result) {
  if (err) return res.send(err);
  res.sendStatus(200)
});
};

exports.deletesend = function (req, res) {
  // students 컬렉션에서 해당 documents(레코드) 삭제
  Messages.remove({
    _id: req.body.id
  }, function (err, result) {
    if (err) return res.send(err);
    res.sendStatus(200)
  });
};

//마이페이지에서 확인한 쪽지에 답장을 보내는 경우 실행
exports.reply = function (req, res) {
  const isLogin = req.isAuthenticated();
  if(isLogin) {
    User.load(req.user.email, function (user) {
      res.render('friends/reply', {
        isUserLoggedIn: isLogin, user: user, status:status, receiver:req.params.receiver,
        sender:req.params.sender
      });
    });
  }
};

//답장 정보 저장
exports.sendreplymessage = function (req, res) {
  const messages = new Messages(only(req.body, 'content receiver sender'));
  messages.save(function (err, result) {
    if (err) {
      res.sendStatus(400)
    }
    res.redirect('/mypage');
  });
};

```

```
});  
};
```

■ 화면18

화면18	설명
	<p>- 관리자로 로그인을 하면 프로필 사진이 푸른 계열이다. (일반 회원은 붉은 계열)</p> <p>-관리자는 DB에 저장된 모든 쪽지와 회원 목록을 확인 할 수 있다.</p> <p>-관리자 화면에서 관리자는 쪽지를 삭제할 수 있다.</p>

소스코드

adminpage.hbs

```
<nav id="nav"><ul>  
  li><a href="#receive" id="top-link"><span class="icon solid fa-file-import">쪽지 목록</span></a></li>  
  <li><a href="#send" id="portfolio-link"><span class="icon solid fa-user-friends">회원 목록</span></a></li></ul></nav></div></div>  
  
  <div id="main">  
    <section id="top" class="five dark cover">  
      <div class="container">  
      </div>  
    </section>  
  
    <section id="receive" class="two">  
      <div class="container">  
        <header>  
          <h2>쪽지 목록</h2>  
        </header>  
  
        <table class="table table-hover">  
          <thead>  
            <tr>  
              <th scope="col"></th>  
              <th scope="col">내용</th>  
            </tr>  
          </thead>  
          <tbody>  
            {{#each messages}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -
```

```

->
        <tr class="table-odd">
            <th scope="row"><a href="/readreview/{{this._id}}">삭제</a></th>
            <td>{{this.content}}</td>
        </tr>
    {{/each}}
</tbody>
</table></div>

</section>
<section id="send" class="three">
    <div class="container">
        <header>
            <h2>회원 목록</h2>
        </header>
        <section id="testimonials">
            <table class="table table-hover">
                <thead>
                    <tr>
                        <th scope="col">관리자</th>
                        <th scope="col">이름</th>
                        <th scope="col">이메일</th>
                    </tr>
                </thead>
                <tbody>
                    {{#each users}} <!-- 는 핸들바의 루프 템플릿으로 students 객체(배열)의 크기만큼 반복 처리됨 -->
                        <tr class="table-odd">
                            <th scope="row">{{this.admin}}</th>
                            <td>{{this.name}}</td>
                            <td>{{this.email}}</td>
                        </tr>
                    {{/each}}
                </tbody>
            </table>
            </section><!-- #testimonials -->
        </div>
    </section>
</div>

```


main.js

//관리자 페이지

```

exports.adminpage = function (req, res) {
  const isLogin = req.isAuthenticated();
  Messages.list(function (messages) {
    if(isLogin) {
      User.load(req.user.email, function (user) {
        User.list(function (users) {
          res.render('main/adminpage', {
            messages:messages, isUserLoggedIn: isLogin, user: user, status:status, users:users
          });
        });
      });
    }
  });
};

```

2. 후기**■ 한다현,김도희****■ 소감**

수업 초반에 html, css에 대해 되짚어주시고 Javascript, jquery, node.js 까지 광범위하게 배울 수 있어서 힘들었지만 보람찼다.

반응형 모바일 웹 프로젝트를 하면서 수업 시간에 배운 다양한 이론들을 실제로 적용하고 구현하는 과정을 통해서 지식이 이론에만 그치지 않고, 실전까지 확장할 수 있었다고 생각한다.

프로젝트를 통해 UI, UX디자인까지 구현을 하게 되면서 프론트엔드와 백엔드 모두를 다룰 수 있었던 좋은 시간이었다. 수업시간에 배운 bootstrap을 통해 더 퀄리티 높은 UI를 디자인 할 수 있었다고 생각한다.

■ 강의를 통해 얻은 것

웹프로그래밍 수업을 수강할때는 단순히 html만을 공부했었기 때문에 실제로 이것들이 어떻게 사용이 되는지 알기가 어려웠으나, 이번 모바일웹 수업을 통해서 서버와 데이터베이스를 함께 다루게 되어 웹에서 어떻게 서버가 사용되고 데이터베이스가 사용되는지 공부할 수 있었다. 서버와 데이터베이스, 사용자 인터페이스를 모두 따로따로 배워 그 개념이 분산되어 있었는데, 이 수업을 통해 그 개념들을 한군데로 모을 수 있었고, 실제 현업에서 내가 배운것들이 어떻게 사용되는지 배울 수 있었다.