


# Flyscanner

JSP, AJAX, Django를 활용한 항공권 예매 사이트

1조 이승연, 박경민, 장민성, 김정하

# 목차

## 01 개발 과정

- 
- 1) 프로젝트 개요
  - 2) 보유 기술 및 사용 툴
  - 3) 프로젝트 스케줄
  - 4) 시스템 구성도
  - 5) UML 및 클래스 다이어그램
  - 6) 시퀀스 다이어그램
  - 7) 데이터베이스 설계
  - 8) ERD

## 02 기술 설명

## 03 참여 소감 및 Q&A

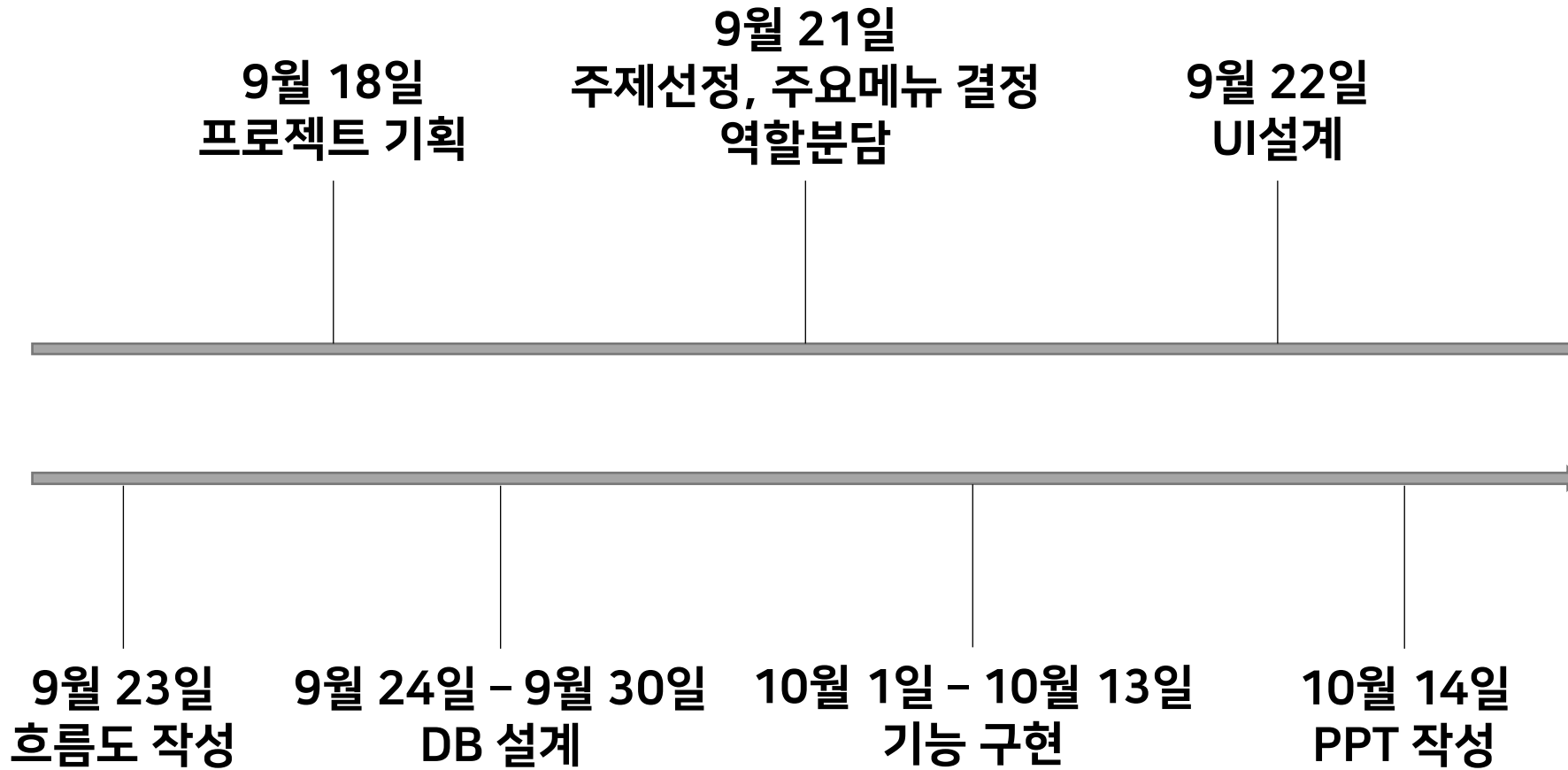
# 1) 프로젝트 개요

프로젝트 주제	항공권 예매 및 여행지 추천 서비스
주요 기능	회원가입, 로그인, 아이디/비밀번호 찾기 항공권 검색 및 조회, 장바구니 항공권 결제 여행지 추천 및 조회
기획 의도	여행을 너무 가고 싶은 4인의 바람이 담긴 프로젝트

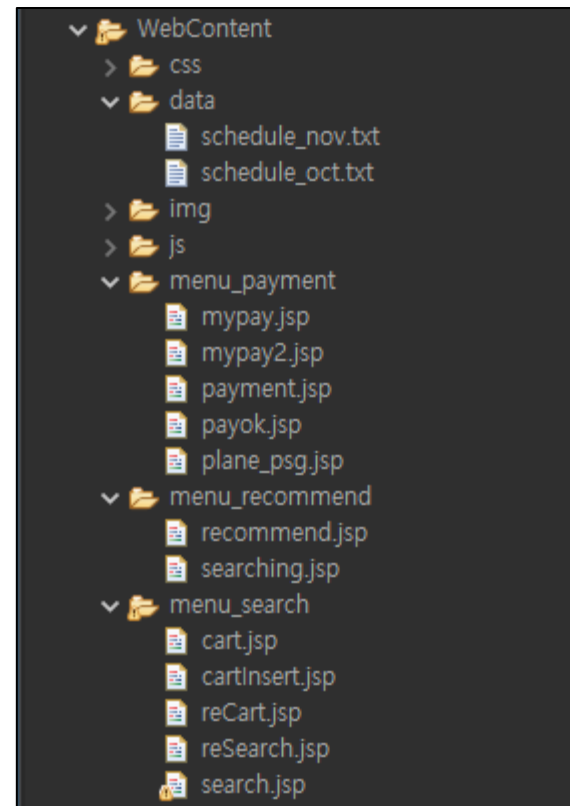
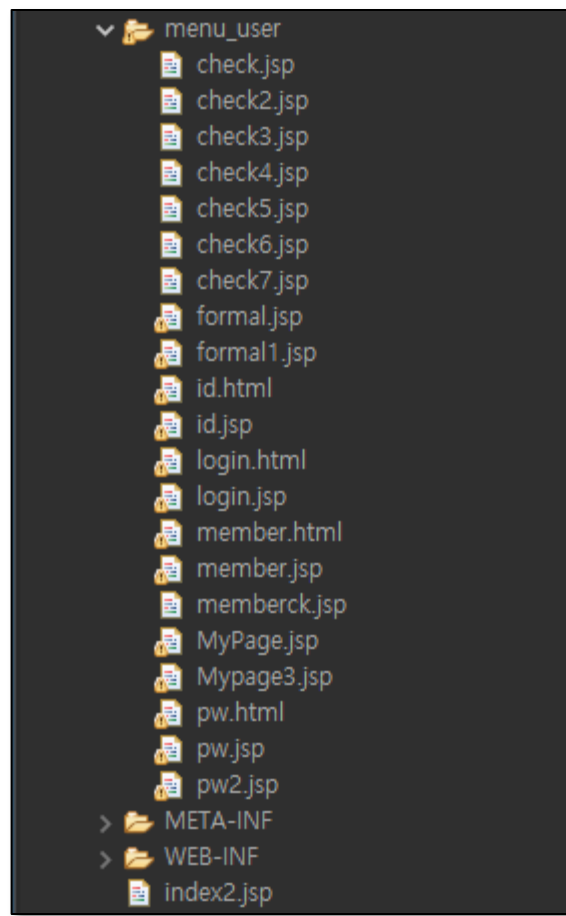
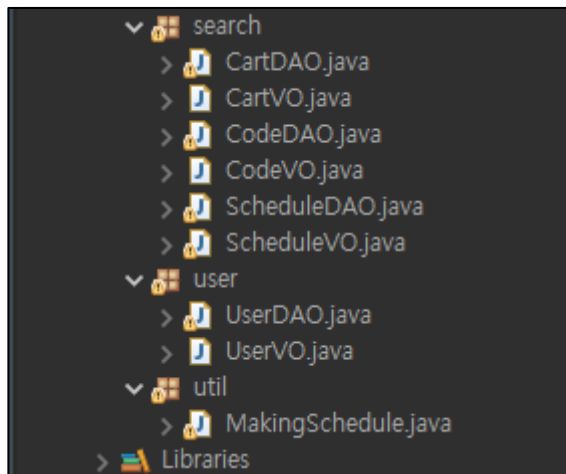
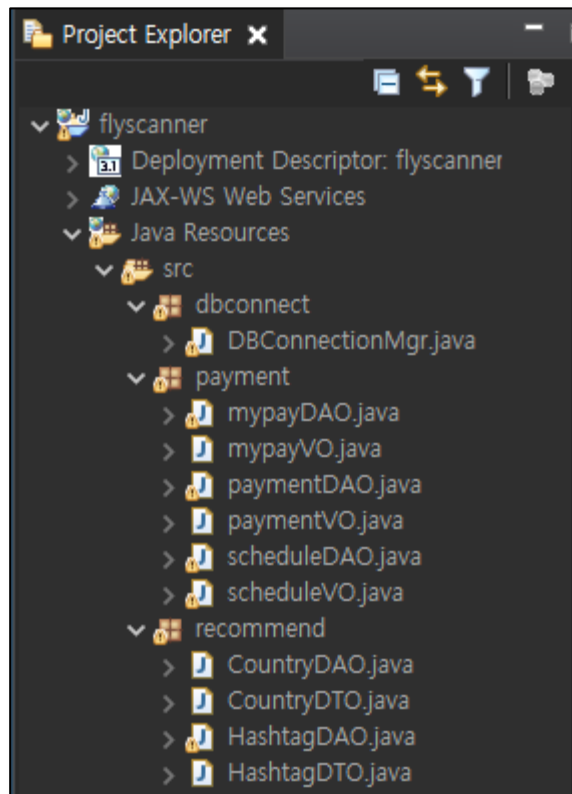
## 2) 보유 기술 및 사용 툴



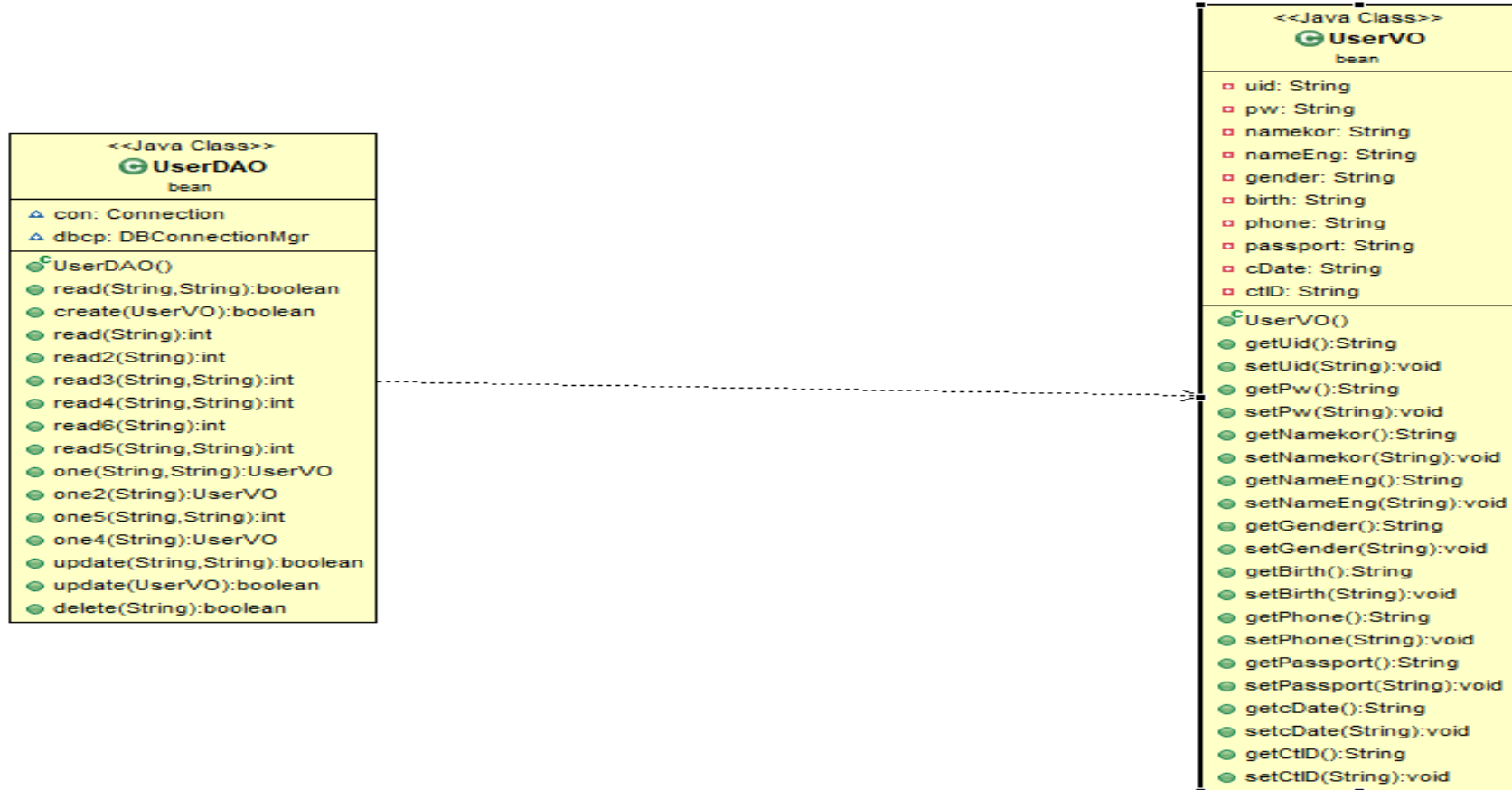
### 3) 프로젝트 스케줄



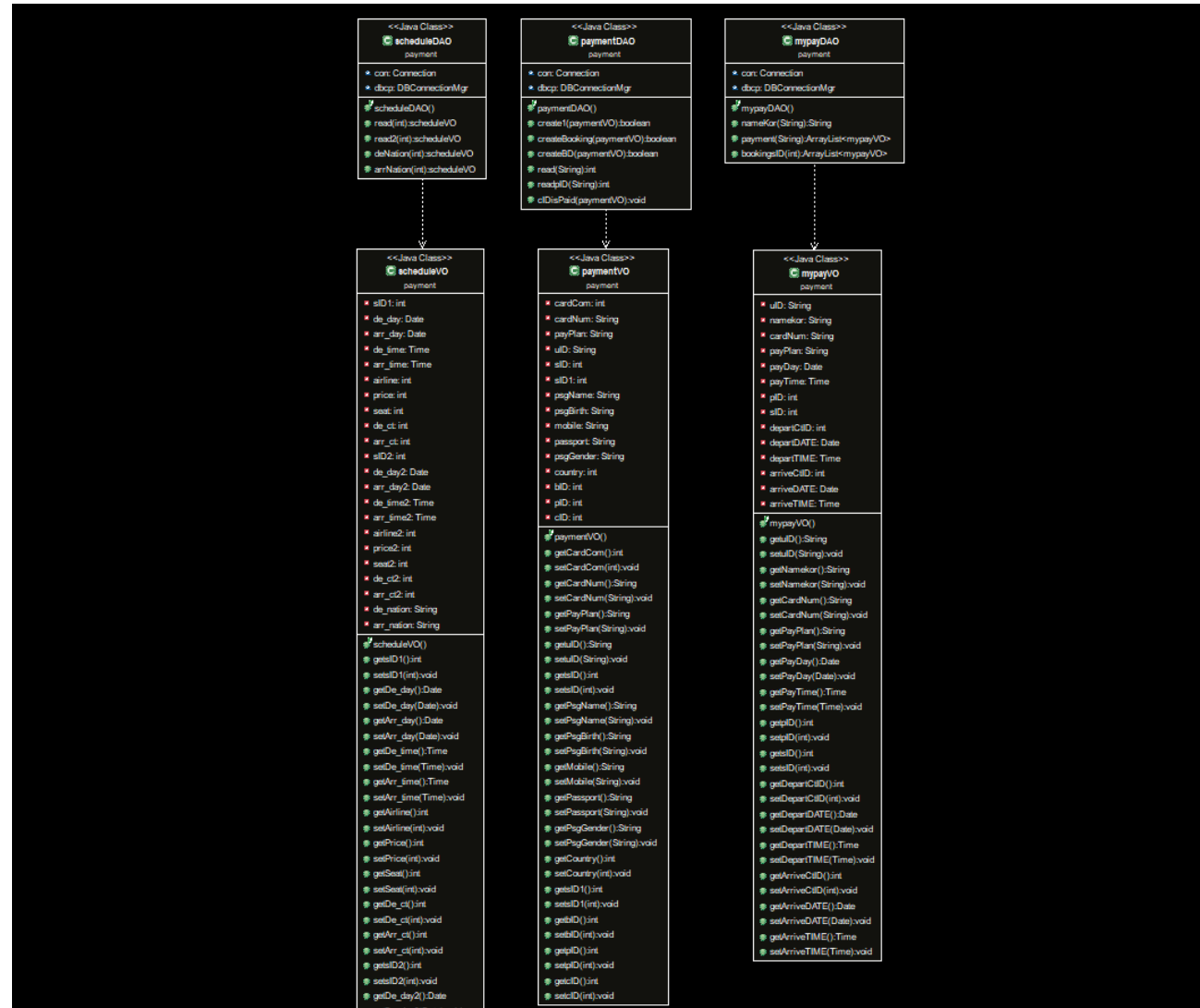
## 4) 시스템 구성도



# 5) UML

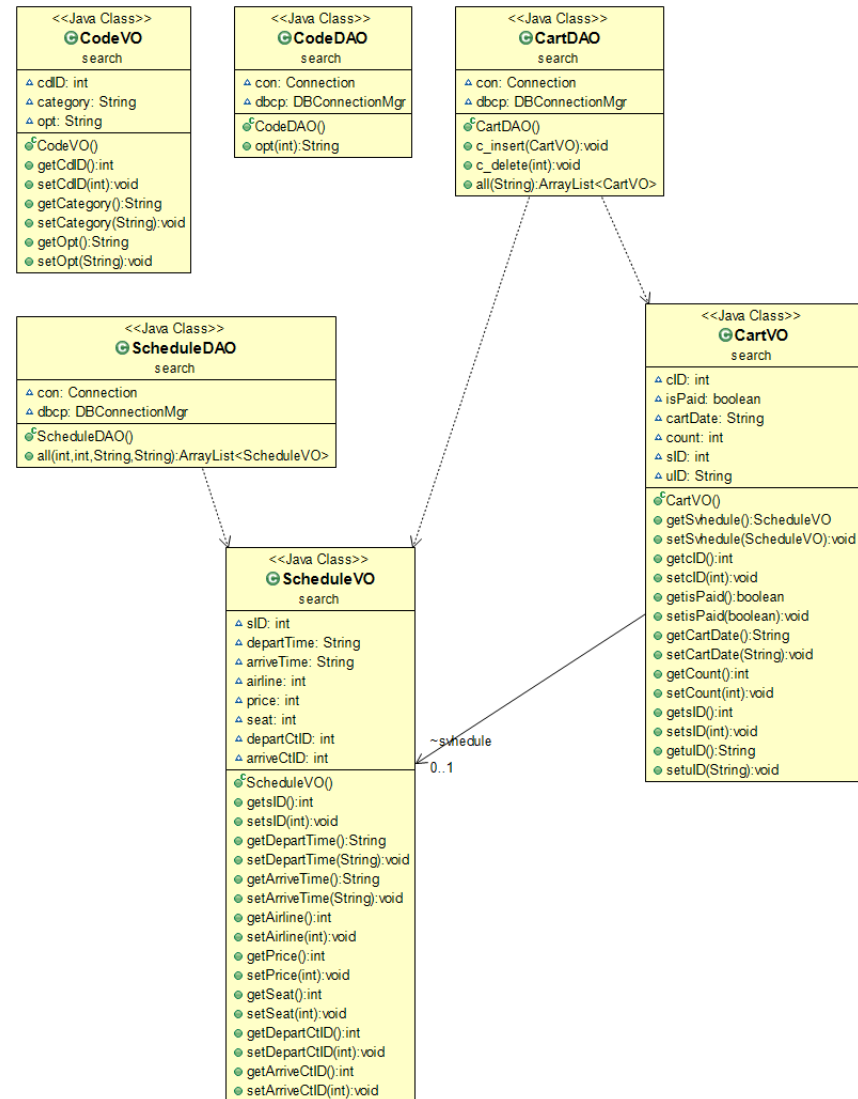


## 5) UML

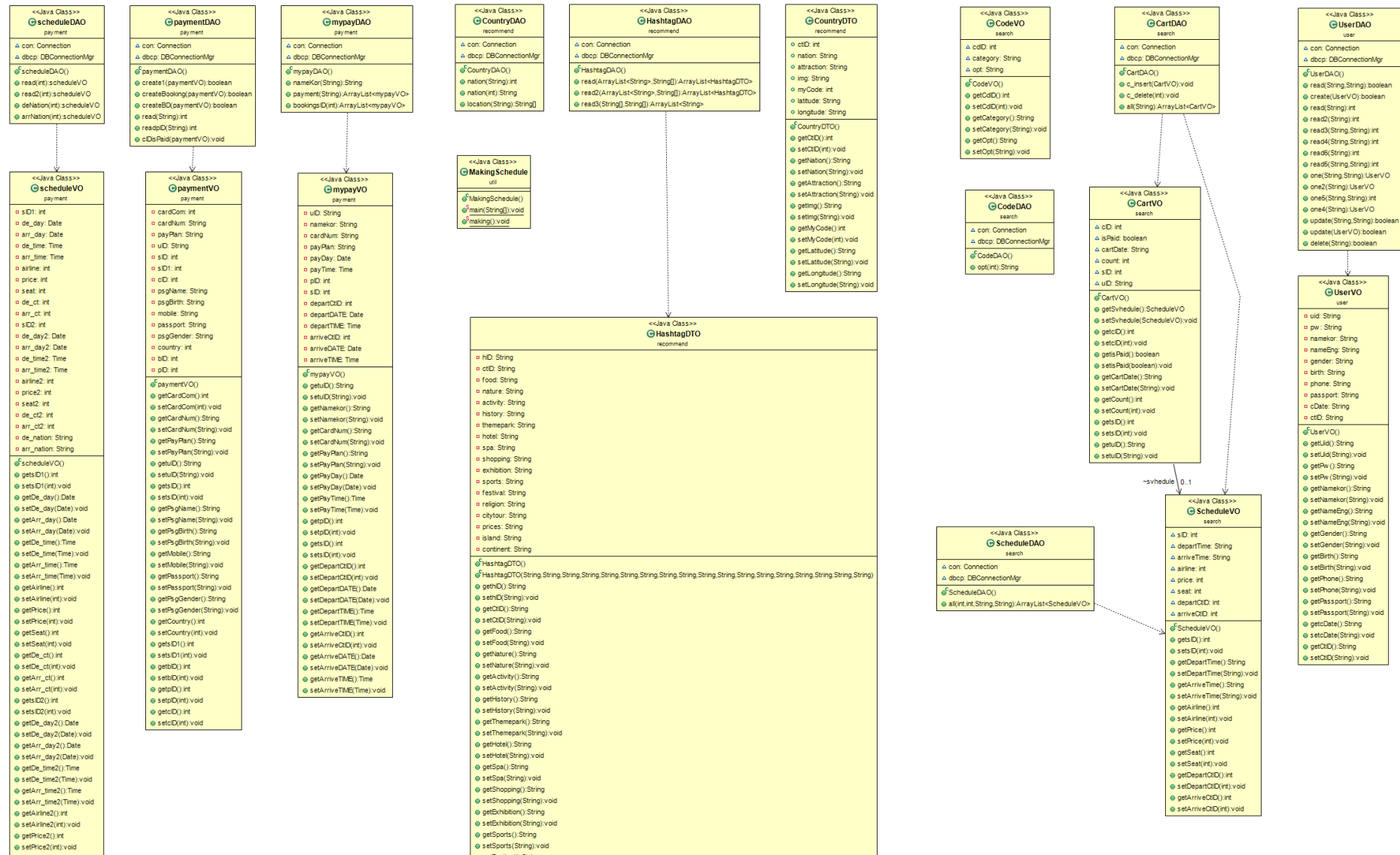




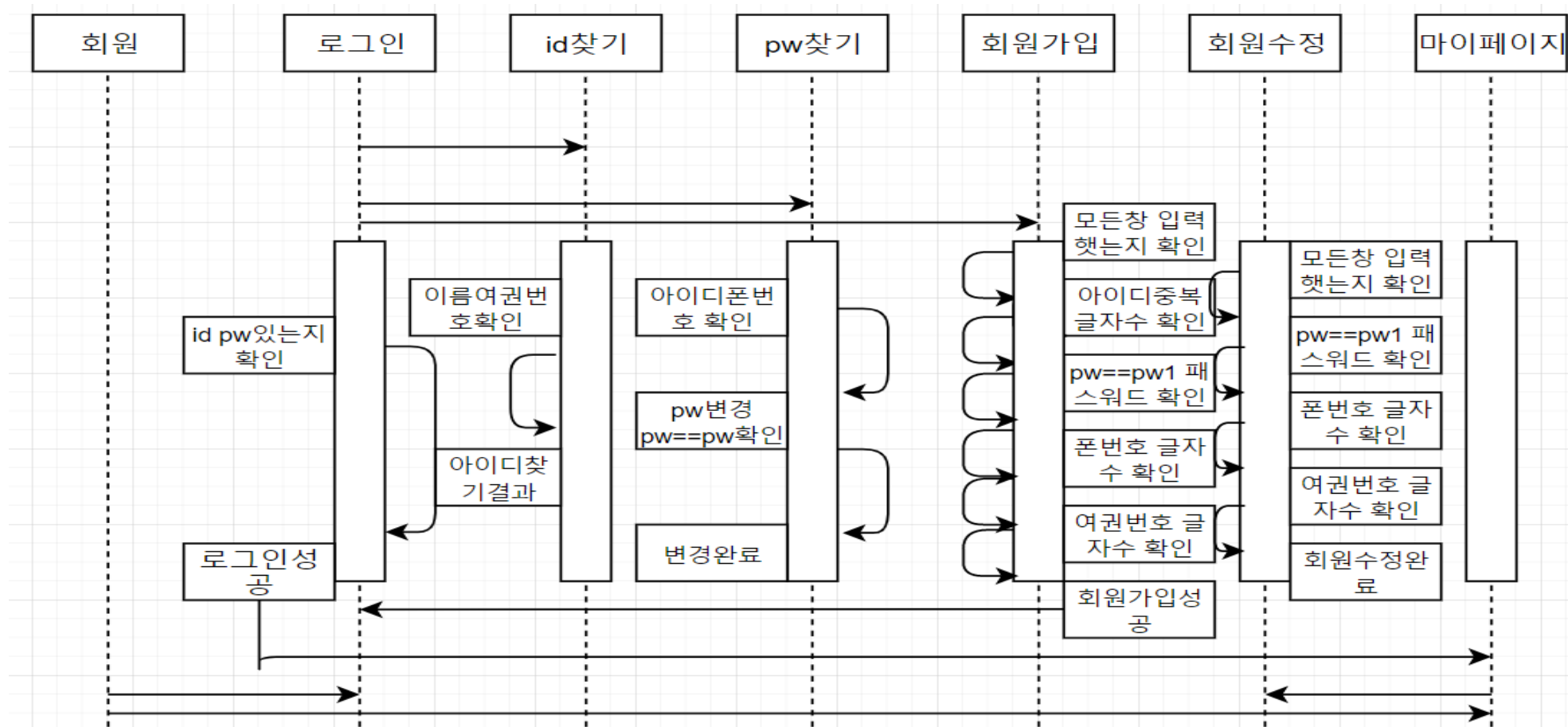
## 5) UML



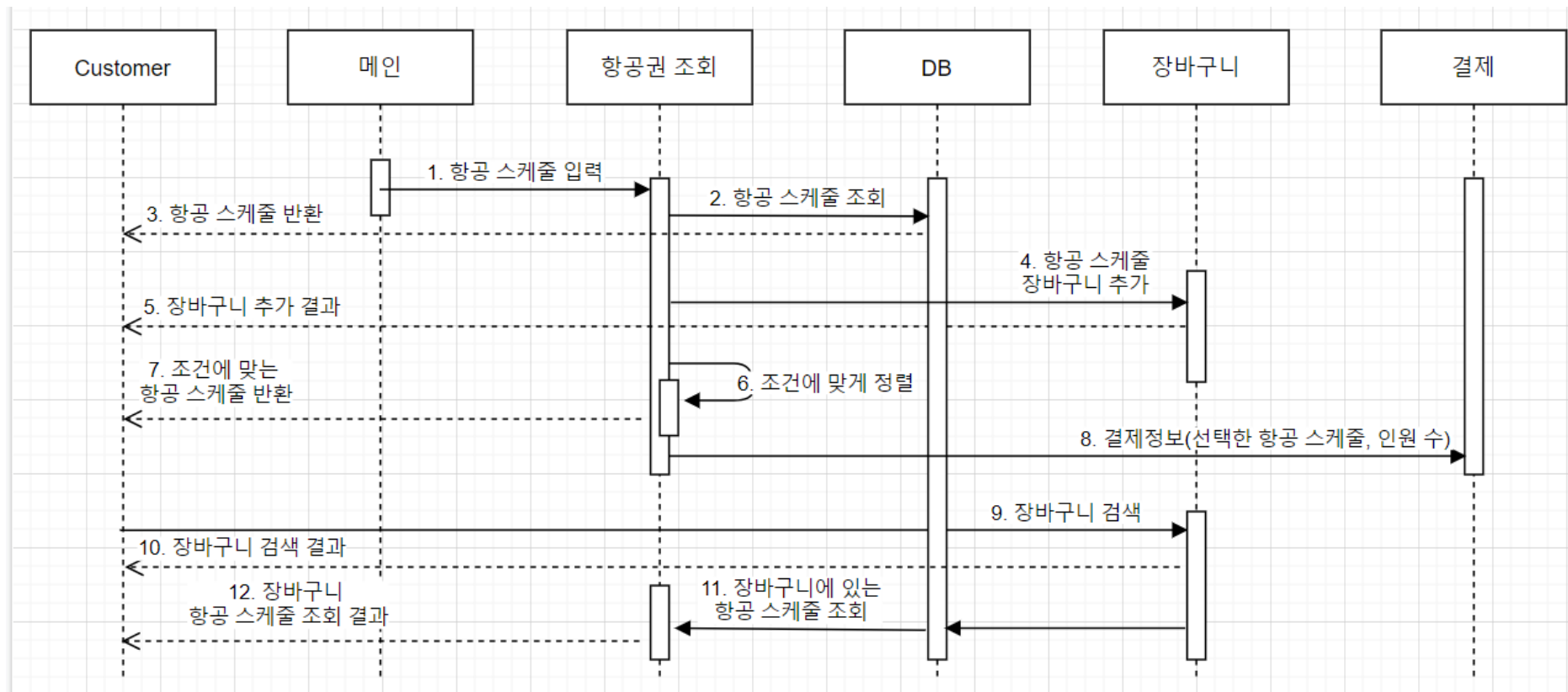
## 5) 클래스 다이어그램



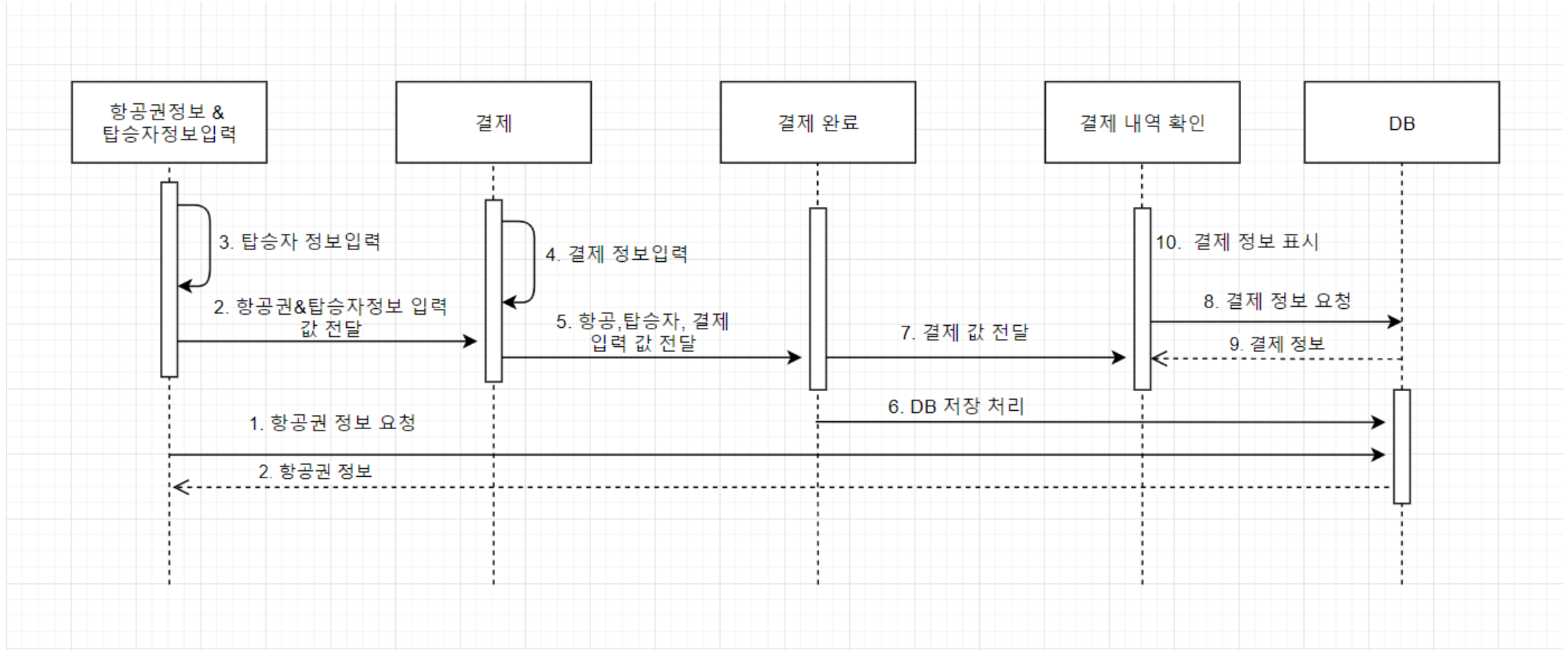
## 6) 시퀀스 다이어그램



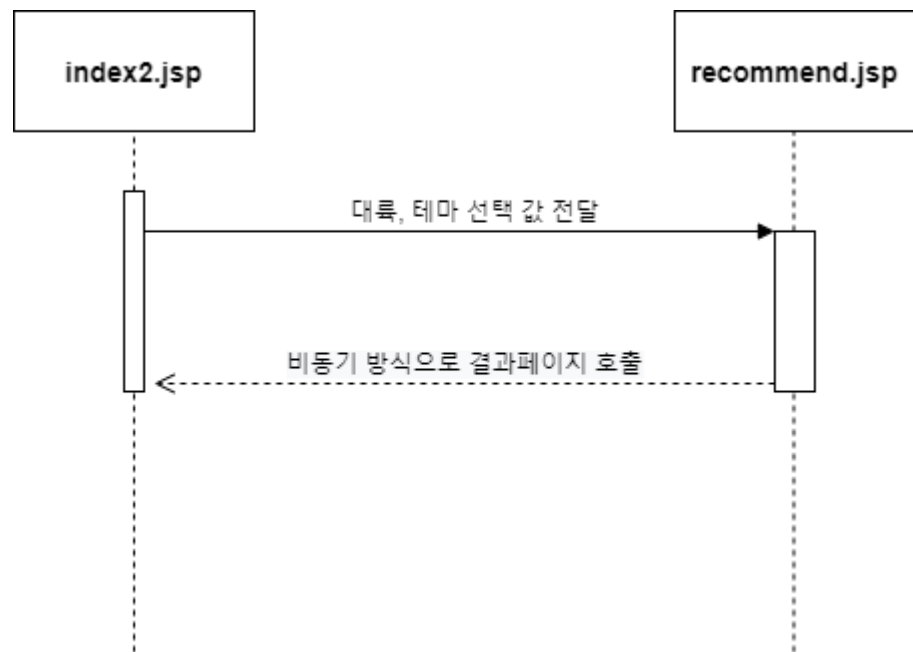
## 6) 시퀀스 다이어그램



## 6) 시퀀스 다이어그램



## 6) 시퀀스 다이어그램



# 7) DB 설계 - code, country, user

스키마 이름	테이블 이름	열 이름	열 설명	데이터 타입	키	널 여부	유일성 여부	기타
	code	cdID	코드 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		category	카테고리	varchar(10)		NOT NULL		
		opt	항목	varchar(20)		NOT NULL	UNIQUE	
	country	ctID	국가 아이디	int	PK	NOT NULL	UNIQUE	
		nation	국가명	varchar(30)		NOT NULL	UNIQUE	
		attraction	관광지	varchar(20)		NULL		
		img	사진	varchar(100)		NOT NULL		
		latitude	위도	varchar(100)		NULL		
		longitude	경도	varchar(100)		NULL		
	user	uID	유저 아이디	varchar(30)	PK	NOT NULL	UNIQUE	
		pw	비밀번호	varchar(20)		NOT NULL		
		nameKor	한글이름	varchar(10)		NOT NULL		
		nameEng	영문이름	varchar(30)		NOT NULL		
		gender	성별	char(1)		NOT NULL		CHECK ('F', 'M')
		birth	생년월일	char(8)		NOT NULL		
		phone	전화번호	char(11)		NOT NULL	UNIQUE	
		passport	여권번호	char(9)		NOT NULL	UNIQUE	
		cDate	가입일	date		NOT NULL		
		ctID	나라 아이디(국적)	int	FK (country)	NOT NULL		CHECK (cdID)

## 7) DB 설계 - schedule

	schedule	sID	스케줄 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		departTime	출발일자	datetime		NOT NULL		
		arriveTime	도착일자	datetime		NOT NULL		
		airline	항공사	int		NOT NULL		CHECK (cdID)
		price	가격	int		NOT NULL		
		seat	좌석	int		NOT NULL		DEFAULT 5
		departCtID	나라 아이디(출발국가)	int	FK (country)	NOT NULL		
		arriveCtID	나라 아이디(도착국가)	int	FK (country)	NOT NULL		



## 7) DB 설계 - cart, booking

flyscanner	cart	<u>cID</u>	장바구니 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		isPaid	결제 여부	tinyint		NOT NULL		DEFAULT 0
		cartDate	생성일	date		NOT NULL		
		count	인원 수	int		NOT NULL		
		sID	스케줄 아이디	int	FK (schedule)	NOT NULL		
	booking	uID	유저 아이디	varchar(20)	FK (user)	NOT NULL		
		<u>bID</u>	예약 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		bookDate	예약일	date		NOT NULL		
		sID	스케줄 아이디	int	FK (schedule)	NOT NULL		
		uID	유저 아이디	varchar(20)	FK (user)	NOT NULL		
		pID	결제 아이디	int	FK (payment)	NOT NULL		

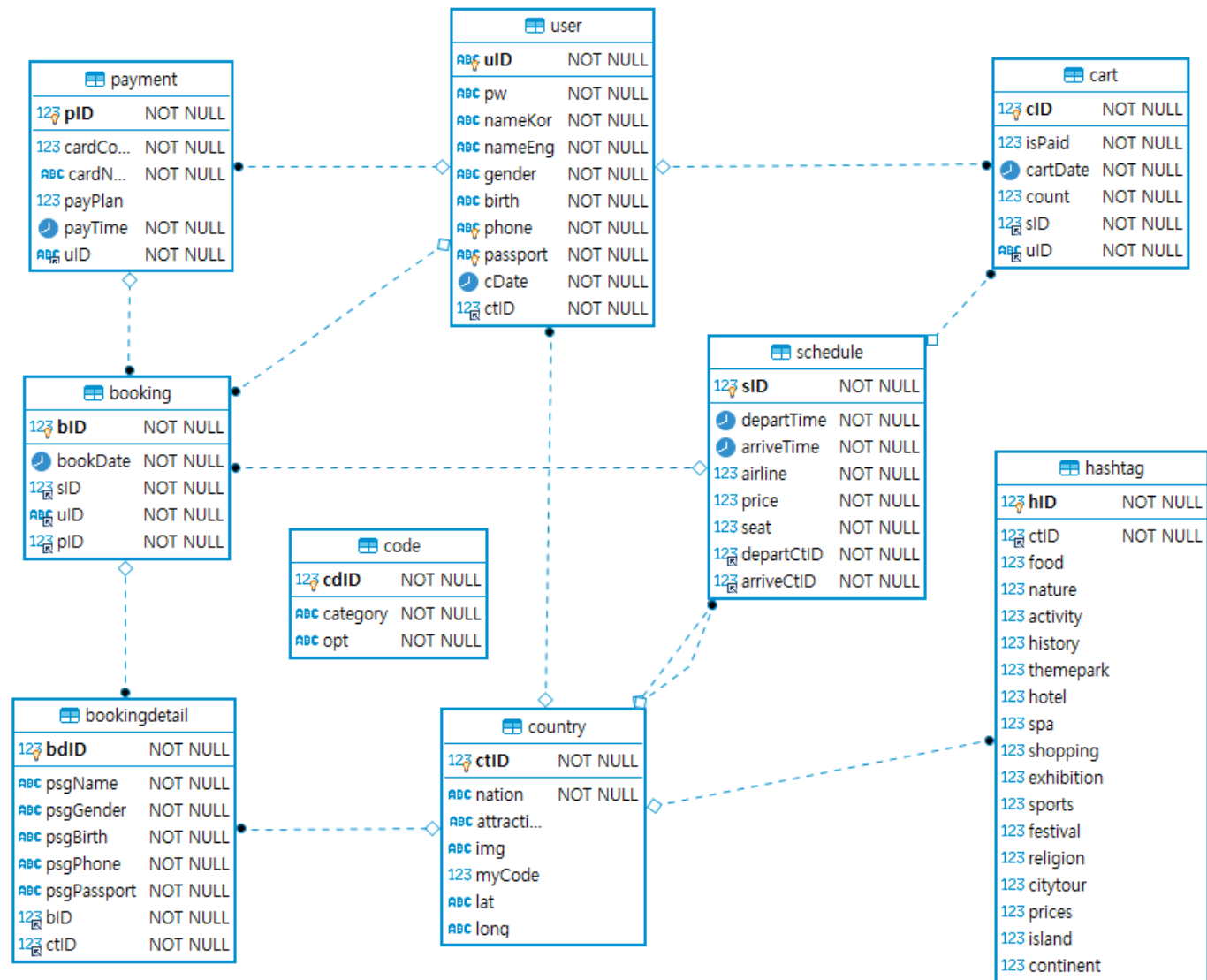
## 7) DB 설계 - bookingDetail, payment

	bookingDetail	bdID	예약 상세 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		psgName	탑승객 이름	varchar(20)		NOT NULL		
		psgGender	탑승객 성별	char(1)		NOT NULL		CHECK ('F', 'M')
		psgBirth	탑승객 생년월일	char(8)		NOT NULL		CHECK (YYYYMMDD)
		psgPhone	탑승객 전화번호	char(11)		NOT NULL	UNIQUE	CHECK (000000000)
		psgPassport	탑승객 여권번호	char(9)		NOT NULL	UNIQUE	CHECK (M000000000)
		bID	예약 아이디	int	FK (booking)	NOT NULL		
		ctID	나라 아이디(탑승객 국적)	int	FK (country)	NOT NULL		
	payment	pID	결제 아이디	int	PK	NOT NULL	UNIQUE	AUTO_INCREMENT
		cardCom	카드사	int		NOT NULL		CHECK (cdID)
		cardNum	카드번호	char(16)		NOT NULL		CHECK (0000000000000000)
		payPlan	할부	varchar(10)		NOT NULL		CHECK (2<= payPlan <=???)
		payTime	결제일	datetime		NOT NULL		
		uID	유저 아이디	varchar(20)	FK (user)	NOT NULL		

# 7) DB 설계 - hashtag, country(Django)

	hashtag	hID	해시태그 아이디	int	PK	NOT NULL	UNIQUE	
		food	음식관광	int		NULL		
		nature	자연관광	int		NULL		
		activity	야외 액티비티	int		NULL		
		history	유적지 및 문화유산	int		NULL		
		themepark	테마파크 및 동식물원	int		NULL		
		hotel	휴양지	int		NULL		
		spa	온천 및 스파	int		NULL		
		shopping	쇼핑	int		NULL		
		exhibition	지역 문화예술공연 및 전시회	int		NULL		
		sports	스포츠 경기 관람	int		NULL		
		festival	지역 문화축제	int		NULL		
		religion	종교 및 성지순례	int		NULL		
		citytour	시티투어	int		NULL		
		island	섬	int		NULL		
		continent	대륙	int		NULL		
		ctID	나라 아이디	int	FK (country)	NOT NULL		
SQLite3	country	clD	나라 아이디	IntegerField	PK	NOT NULL	UNIQUE	
		name	나라 이름	CharField		NOT NULL	UNIQUE	

## 8) ERD



## 1-1) 로그인

```
$("#b1").click(function() {
    아이디 값 b1 버튼이 클릭될 경우
$.ajax({
```

```
    url : "check4.jsp",
    Check4.jsp
```

```
    data : {
        Email: Email,
        pw: pw
```

데이터: 이메일, 패스워드

```
    success : function(result)
```

```
    if (result==1)
```

```
location.href='login.jsp?Email='+Email+'&pw='+pw
```

```
String Uid = request.getParameter("Email");
String pw = request.getParameter("pw");
UserDAO dao = new UserDAO();
```

```
    int result = dao.read4(Uid, pw);
```

```
    if(result==1){ // 성공
        session.setAttribute("uID", Uid);
    }
    else { // 실패
        check = "<font color=red>아이디 혹은 비밀번호를 다시 확인해주세요</font>";
    }
}
```

```
%><%=result%>
```

받은값을 getparameter로 불러온후 userDAO를생성해서 int result= dao.read4(Uid , pw)

```
String sql = "select * from user where pw = '"+pw.trim()+"' and uID = '"+Uid.trim()+"'";
```

DAO로보내진 값을 DB에 있는지확인

B1이란 버튼을 클릭하면 ajax로 url : check4.jsp파일로 데이터값 이메일 패스워드를 보낸뒤 맞으면 success으로 리절트값을불러와서 if문으로 (result=1) 값이있으면 login.jsp로 Email값과 pw값 보내준다

localhost:8888 내용:  
아이디를확인해주세요

확인

PW찾기

saffa  
이메일

112  
휴대폰 번호

확인

localhost:8888 내용:  
아이디 비밀번호 를확인해주세요

확인

Email address  
이메일

Password  
패스워드

로그인

회원가입

ID/ PW찾기

Login

localhost:8888 내용:  
아이디를확인해주세요

확인

ID찾기

asd  
이름

1234  
여권번호

확인

마이 페이지

localhost:8888 내용:  
아이디를확인해주세요

확인

1223411234

..

수정

검색내역

# 1-2) id찾기

```
String passport = request.getParameter("passport");  
  
UserDAO dao = new UserDAO();  
UserVO vo = dao.one2(passport);
```

받은값을 getparameter로 불러온후 userDAO를생성해서 int result= dao.red4(Uid , pw)

```
<%=vo.getNamekor() %> 님 찾으신결과는  
ID: <%=vo.getId() %>
```

입력받은값의 아이디와 네임을 가져와서 보여주기

검색하신 id결과는

sgsgsf 님 찾으신결과는

ID: a

```
String sql = "select * from user where passport ='"+ passport.trim()+ "'";  
String namekor1 = rs.getString("namekor");  
String uid = rs.getString("uid");
```

passport 입력값을 받아서 결과가맞는지확인

```
bag.setNamekor(namekor1);  
bag.setUid(uid);  
return bag;
```

가방에 네임과 아이디를넣어서

# 1-3) pw변경

```
String Uid =request.getParameter("Email");
String phone =request.getParameter("phone");
UserDAO dao = new UserDAO();
int result = dao.one5(Uid, phone);
System.out.println("나야나 "+Uid);
```

입력값을받은뒤 dao로보냄

```
con = dbcp.getConnection();
String sql = "select * from user where Uid = ? and phone = ?";
PreparedStatement ps = con.prepareStatement(sql);
ps.setString(1, Uid);
ps.setString(2, phone);
```

아이디와 폰번호를 받아서 입력값이DB에잇는지확인

```
$("#btn").click(function() {
    uid = '<%= Uid%>';
    pw = $('#pw').val();
    pw1 = $('#pw1').val();

    if (pw == pw1) {
        location.href='pw2.jsp?Uid='+uid+'&pw='+pw;
    }
    else {
        alert("비밀번호가 같지 않습니다.");
    }
});
```

입력된값을 DAO로전송

localhost:8888 내용:  
비밀번호가 같지 않습니다.

확인

## 비밀번호변경

....

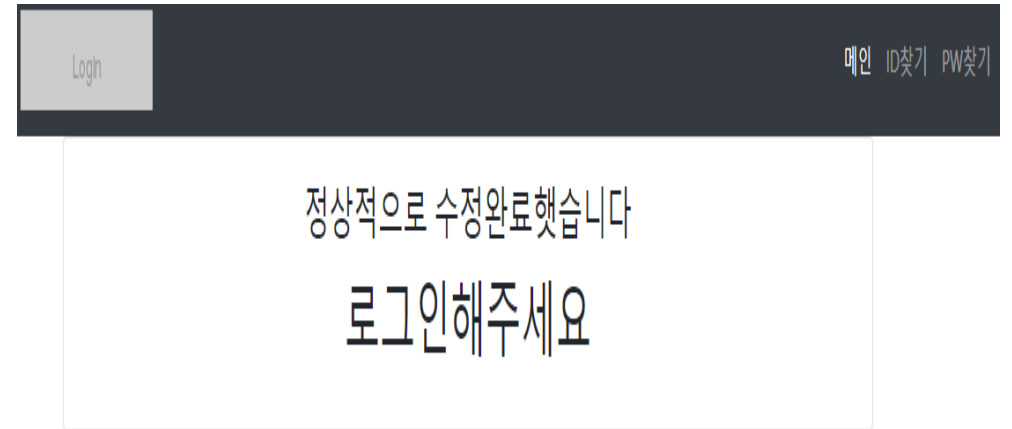
패스워드

.....

패스워드 확인

변경GG

# 1-3) pw변경



```
String sql = "update user set pw = ? where Uid = ?";
```

Uid 입력을받아 보여준후 Uid의 pw결과를 업데이트

```
String pw = request.getParameter("pw");  
String Uid = request.getParameter("Uid");  
UserDAO dao = new UserDAO();  
boolean result = dao.update(pw, Uid);
```

입력값들을 받아서 DAO를생성후 dao.update 보내준다



# 1-4) 회원가입

회원가입

sdgdsg  
최소 8글자입력하세요  
이메일

.....  
패스워드

.....  
패스워드 확인해주세요  
패스워드 확인

dfhfdhdfh  
이름

fdhfhddfh  
영문 이름

☒ 남자 ☐ 여자

성별

20/10/29  
생년월일

0102221111111  
최대글자수 11입니다  
휴대폰 번호

대한민국  
국가

11111111111  
최대글자수 9입니다  
여권번호

회원가입

```

    '#passport'
    '#phone'
    $('#Email').blur(function() {
        '#pw1'

        $.ajax({
            url : "check.jsp",
            data : {
                id : $('#Email').val()

            if (n1 == n2)        if (phone.length >=12)
            if (result == 1)    if (id.length <= 7)        if (passport.length >=10)

        $('#d1').html('<font color =red>이미 가입된 아이디입니다</font>')

        Id값=d1,d2,d3,d4  div에 글이나올수있
        도록

```

패스워드 여권번호 폰번호 이메일 각각 blur을줘서 입력을하면 나타나게하는 방식으로

중복체크를 위한

각각 조건물을줘서

```

    $('#b1').click(function() {
        B1이란 아이디값인 버튼을 누르면

        if (Email.lenth == "" || pw.length== "" || nameKor.length == "" || nameEn
            alert("값을 모두 입력해주세요")
        }else if (id.length <= 7) {
            //alert("아이디를 5글자 이상 입력하세요")
            alert("아이디 확인해주세요");

        }else if(result == 1) {
            alert("아이디를확인해주세요");

        }else if (passport.length >=10) {
            alert("여권번호 확인해주세요");
        } else if (phone.length >= 12) {
            alert("전화번호 확인해주세요")
        } else if (n1==n2) {
            location.href='member.jsp?Email='+Email+'&pw='+pw+'&nameKor='
        }else{
            alert("패스워드 확인해주세요")
        }
    });

```

이메일 패스워드 이름 영문이름 영문이름 성별 생년월일 휴대폰번호 국가 여권번호가입력이안될경우 alert로 알림창 뜸

Id가 중복될경우

Passport입력값이 10보다크거나갈을경우

Phone입력값이 12보다크거나 작을경우

N1==n2같은경우 member.jsp로입력값 전송

# 1-4) 회원가입

회원가입

sdgdsd  
최소 8글자입력하세요

이메일  
.....@......co.kr  
.....@......co.kr

비밀번호  
비밀번호를 확인해주세요  
비밀번호

dfhfdhdfh  
이름

fdhfhddfh  
영문 이름

☒ 남자 ☐ 여자  
성별

20/10/29  
생년월일

0102221111111  
최대글자수 11입니다  
휴대폰 번호

대한민국  
국가

11111111111  
최대글자수 9입니다  
여권번호

회원가입

```
UserVO vo = new UserVO();
//2. 전달되는 값 받은 다음, VO에 넣어야 함.
String id = request.getParameter("Email");
String pw = request.getParameter("pw");
String nameKor = request.getParameter("nameKor");
String nameEng = request.getParameter("nameEng");
String birth = request.getParameter("birth");
String gender = request.getParameter("gender");
String phone = request.getParameter("phone");
String passport = request.getParameter("passport");
String ctID = request.getParameter("ctID");
```

```
vo.setUid(id);
vo.setPw(pw);
vo.setNameKor(nameKor);
vo.setNameEng(nameEng);
vo.setBirth(birth);
vo.setGender(gender);
vo.setPhone(phone);
vo.setPassport(passport);
vo.setCtID(ctID);
```

```
UserDAO dao = new UserDAO();
boolean result = dao.update(vo);
```

```
String uid = request.getParameter("id");
```

```
UserDAO dao = new UserDAO();
```

```
int result = dao.read(uid);
```

입력된 id값을 받아서 userDAO를 생성후 입력값을 넣어준다

```
con = dbcp.getConnection();
```

```
String sql = "select * from user where uID = ?";
```

```
PreparedStatement ps = con.prepareStatement(sql);
ps.setString(1, uid);
```

입력된 id값을 받아서 userDAO를 생성후 입력값을 넣어준다

```
con = dbcp.getConnection();
```

```
String sql = "update user set namekor = ?, pw = ?, nameEng = ?";
```

```
PreparedStatement ps = con.prepareStatement(sql);
```

```
ps.setString(1, vo.getNamekor());
```

```
ps.setString(2, vo.getPw());
```

```
ps.setString(3, vo.getNameEng());
```

```
ps.setString(4, vo.getGender());
```

```
ps.setString(5, vo.getBirth());
```

```
ps.setString(6, vo.getPhone());
```

```
ps.setString(7, vo.getPassport());
```

```
ps.setString(8, vo.getCtID());
```

```
ps.setString(9, vo.getUid());
```

받은 값을 UserVO 에 넣어서 UserDAO로 VO를 전송

받은 값을 업데이트 시켜준다

# 1-4) 회원수정

```
<div class="col-sm-12">
  <input type="Email" id="Email" name="Email" class="form-control"
    required autofocus class="col-sm-7" value="<%=session.getAttribute("uID") %>" readonly>
  <div class="col-sm-30" id="d1"></div>
  <label for="inputEmail">이메일 </label>
</div>
```

Session 로그인한 id값을 value=session.getAttribute("uID")

## 회원수정

1223411234

이메일

....

패스워드

.....

패스워드 확인해주세요

패스워드 확인

FGDG

이름

DGFDG

영문 이름

☐ 남자 ☒ 여자

성별

20/10/21

생년월일

DGDFGASGASGSAGSAG

최대글자수 11입니다

휴대폰 번호

대한민국 ▼

국가

DFGDFGSAFSAGSASAGSAG

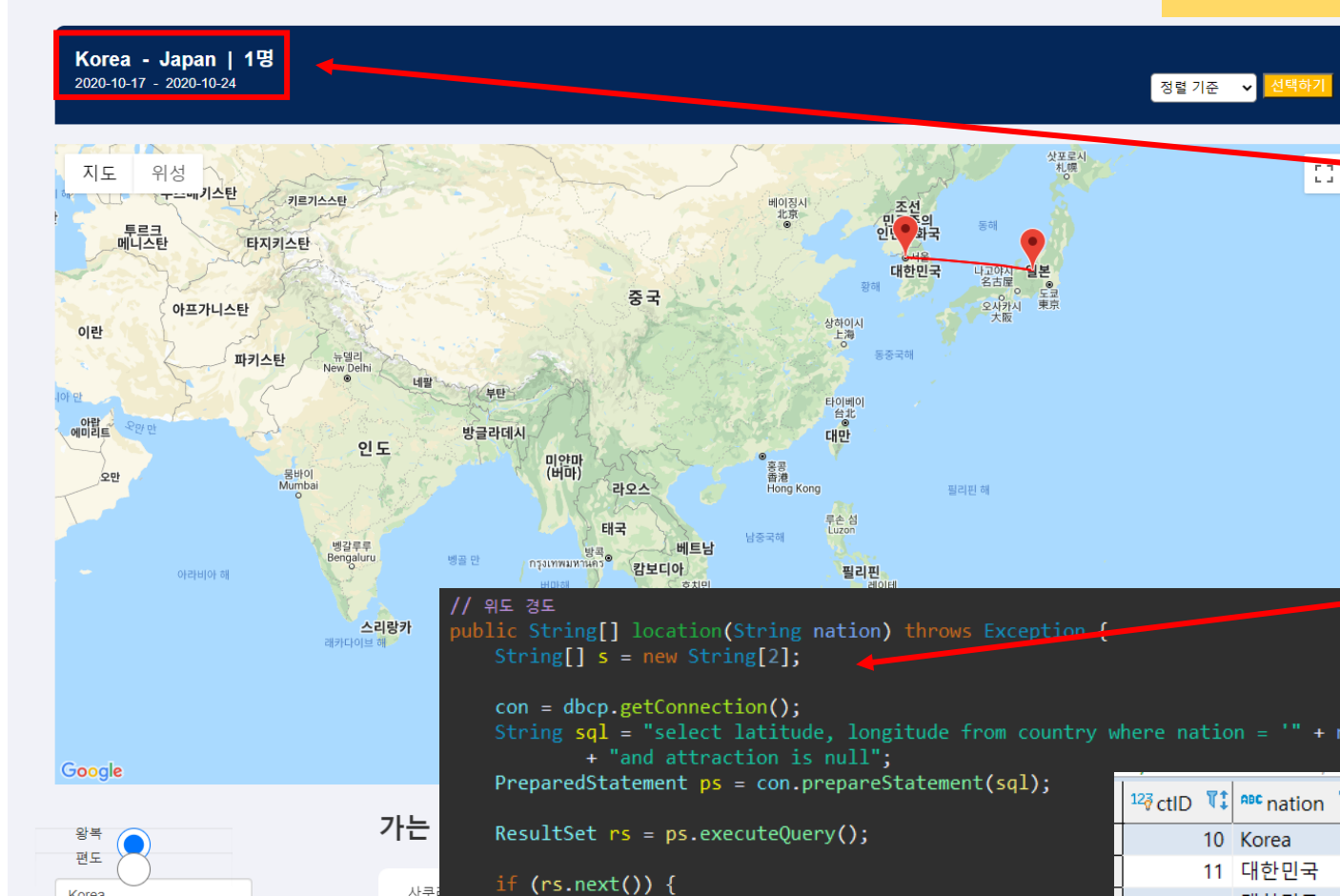
최대글자수 9입니다

여권번호

회원가입

## 2-1) 항공권 조회

1. 메인에서 입력한 값들을 request.getParameter를 사용하여 출력



```
// 받아온 값들
String depart_ct = request.getParameter("departCtID"); // 출발국가
String arrive_ct = request.getParameter("arriveCtID"); // 도착국가
String depart_time = request.getParameter("departTime"); // 출발일자
String depart_time2 = request.getParameter("departTime"); // 출발일자
String arrive_time = request.getParameter("arriveTime"); // 도착일자
int way = Integer.parseInt(request.getParameter("way")); // 왕복, 편도
String seat = request.getParameter("seat"); // 인원수
```

```
function initMap() {
    <%
        CountryDAO dao = new CountryDAO();

        String[] s = new String[2];
        s = dao.location(depart_ct);

        String[] e = new String[2];
        e = dao.location(arrive_ct);
    %>
}
```

2. 출발국가, 도착국가에 대한 위도 경도 정보를 배열에 저장

```
// 위도 경도
public String[] location(String nation) throws Exception {
    String[] s = new String[2];

    con = dbcp.getConnection();
    String sql = "select latitude, longitude from country where nation = '" + nation + "'"
        + "and attraction is null";
    PreparedStatement ps = con.prepareStatement(sql);

    ResultSet rs = ps.executeQuery();

    if (rs.next()) {
        s[0] = rs.getString("latitude");
        s[1] = rs.getString("longitude");
    } else {
        System.out.println("결과x");
    }
    dbcp.freeConnection(con, ps, rs);
    return s;
}
```

ctID	nation	attraction	img	myCode	latitude	longitude
10	Korea	[NULL]	korea.jpg	0	37.551048	126.990637
11	대한민국	해운대	korea.jpg	0	[NULL]	[NULL]

3. 입력한 나라에 대한 위도와 경도를 배열에 넣어서 반환

## 2-1) 항공권 조회



1. 출발지와 도착지에 대한 위도와 경도를 가져옴

```
var locations = [
  ['출발지', <%=s[0]%>, <%=s[1]%>],
  ['도착지', <%=e[0]%>, <%=e[1]%>]
];

var map = new google.maps.Map(
  document.getElementById('map'), {
    zoom: 4,
    center: new google.maps.LatLng(21.736790, 109.017049)
  });

for (var i = 0; i < locations.length; i++) {
  marker = new google.maps.Marker({
    id: i,
    position: new google.maps.LatLng(locations[i][1], locations[i][2]),
    map: map
  });
}

var flightPlanCoordinates = [
  { lat: locations[0][1], lng: locations[0][2] },
  { lat: locations[1][1], lng: locations[1][2] },
];

var flightPath = new google.maps.Polyline({
  path: flightPlanCoordinates,
  geodesic: true,
  strokeColor: "#FF0000",
  strokeOpacity: 1.0,
  strokeWeight: 2
});

flightPath.setMap(map);
}
```

4. geodesic를 사용하면 선을 곡선으로 그려줌

## 2-1) 항공권 조회

### 1. 스케줄을 다시 검색할 수 있게 함

왕복

☒

편도

☐

검색

가는 날 오는 날

사쿠라항공	2020-10-17 03:00:00.0	→	2020-10-17 08:00:00.0	₩500900
Korea	5시간	Japan	잔여좌석 5 / 5	선택
사쿠라항공	2020-10-17 03:00:00.0	→	2020-10-17 08:00:00.0	₩791200
Korea	5시간	Japan	잔여좌석 5 / 5	선택
밍밍항공	2020-10-17 03:00:00.0	→	2020-10-17 08:00:00.0	₩553200
Korea	5시간	Japan	잔여좌석 5 / 5	선택

가는 날

항공사

출발시간

도착시간

가격

### 2. 검색하고자하는 값들을 입력하면 입력된 값들을 form태그를 사용하여 조회 페이지를 다시 호출하여 스케줄을 출력함

```

<!-- 항공권 다시 검색 -->
<form method="get" class="re" id="d1" action="search.jsp">
  <div class="radio">
    <input type="radio" value="2" name="way" checked="checked" id="twoway" class="form-control">
    <label>왕복</label>
  </div>
  <div class="radio">
    <input type="radio" value="1" name="way" id="oneway" class="form-control">
    <label>편도</label>
  </div>
  <input type="text" name="departCtID" placeholder="출발국가" class="form-control" value="<%=depart_ct%>">
  <input type="text" name="arriveCtID" placeholder="도착국가" class="form-control" value="<%=arrive_ct%>">
  <input type="text" name="departTime" placeholder="출발일" id="departTime" class="form-control" value="<%=depart_time%>">
  <input type="text" name="arriveTime" placeholder="도착일" id="arriveTime" class="form-control" value="<%=arrive_time%>">
  <input type="text" name="seat" placeholder="인원" value="1" class="form-control" value="<%=seat%>">
  <button type="submit" id="btnSearch" class="btn btn-info btn-lg">검색</button>
</form>

```

## 2-1) 항공권 조회

1. 정렬하고자하는 항목을 선택하면  
항공 스케줄 항목이 정렬됨

가는 날 오는 날

사쿠라항공	2020-10-17 03:00:00.0	→ 5시간	2020-10-17 08:00:00.0	₩500900
Korea			Japan	잔여좌석 5 / 5
사쿠라항공	2020-10-17 03:00:00.0	→ 5시간	2020-10-17 08:00:00.0	₩791200
Korea			Japan	잔여좌석 5 / 5
밍밍항공	2020-10-17 03:00:00.0	→ 5시간	2020-10-17 08:00:00.0	₩553200
Korea			Japan	잔여좌석 5 / 5

₩651300

2. ajax로 값을 넘겨 받은 후 선택한 값에  
대한 항목으로 order by해서 넘겨줌

```
// '정렬하기' 버튼을 클릭 시
$("#sb").click(function() {
    $('#tt').empty()
    var selectOption = $("#sel option:selected").text();
    $.ajax({ /* 비동기통신 */
        url: "reSearch.jsp",
        data: {
            de: <%=de%>,
            ar: <%=ar%>,
            year: <%=year%>,
            month: <%=month%>,
            day: <%=day%>,
            depart_ct: "<%=depart_ct%>",
            arrive_ct: "<%=arrive_ct%>",
            sortt: selectOption
        },
        success: function(result) {
            $('#tt').append(result)
        }
    })
})// sb click end
```

```
// 넘겨받은 값
int de = Integer.parseInt(request.getParameter("de"));
int ar = Integer.parseInt(request.getParameter("ar"));
String year = request.getParameter("year");
String month = request.getParameter("month");
String day = request.getParameter("day");
String depart_time = year + "-" + month + "-" + day + "%";
String or = request.getParameter("sortt");
String depart_ct = request.getParameter("depart_ct").trim();
String arrive_ct = request.getParameter("arrive_ct");

ScheduleDAO sdao = new ScheduleDAO();
if(or.equals("낮은 가격순")){
    or = "price";
} else if(or.equals("출발 시간순")) {
    or = "departTime";
}
List<ScheduleVO> list = sdao.all(de, ar, depart_time, or);

String[] c = new String[list.size()]; // 장비구니 id값
String[] t = new String[list.size()];

for (int i = 0; i < list.size(); i++) {
    ScheduleVO svo = list.get(i);
    CodeDAO codedao = new CodeDAO();
    String airline = codedao.opt(svo.getAirline());
    %>
```

```
public ArrayList<ScheduleVO> all(int depart, int arrive, String departtime, String ob) throws Exception {
    con = dbcp.getConnection();
    String sql = "SELECT * FROM schedule WHERE departCtID = ? "
        + "AND arriveCtID = ? "
        + "AND departTime LIKE ? "
        + "ORDER BY "+ob;
```



## 2-1) 항공권 조회

1. 테이블 태그를 empty()해서 비움

```
success: function(result) {
    $('#tt').empty()
    $('#tt').append(result)
}
```

2. 정렬된 db를 테이블 태그에 넣어서 append()

```
for (int i = 0; i < list.size(); i++) {
    ScheduleVO svo = list.get(i);
    CodeDAO codedao = new CodeDAO();
    String airline = codedao.opt(svo.getAirline());
    %>
    <table>
    <tr>
    <td class="a"><%=airline %></td><!-- 항공사 -->
    <td class="b"><%=svo.getDepartTime() %></td><!-- 출발시간 -->
    <td rowspan="2" class="c">
        <br>
        <p class="time">결린시간</p>
    </td><!-- 결린시간 -->
    <td class="d"><%=svo.getArriveTime() %></td><!-- 도착시간 -->
    <td rowspan="2" class="e"><%=svo.getPrice() %><br> <br>
        <input type="button" value="선택" class="checkBtn"><br> <br>
        잔여좌석 <%=svo.getSeat() %> / 5
    </td>
    </tr>
    <tr>
    <td class="f"><%=svo.getId() %></td>
    <!-- 출발국가 -->
    <td><%=depart_ct %></td>
    <!-- 도착국가 -->
    <td><%=arrive_ct %></td>
    <td class="icon">
        <button class="cart" ></button>
    </td>
    </tr>
    </table>
```

3. 낮은 가격순으로 선택하면 price로 출발  
시간순으로 선택하면 departTime으로  
order by된 값들 출력



## 2-1) 항공권 조회

1. '가는날'에 대한 스케줄을 선택하면 그 스케줄 값을 가져옴

가는 날 오는 날

시아항공	2020-10-01 10:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩550000	선택
	Korea		Singapore	잔여좌석 5 / 5	
밍밍항공	2020-10-01 09:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩540000	선택
	Korea		Singapore	잔여좌석 5 / 5	
사쿠라항공	2020-10-01 08:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩530000	선택
	Korea		Singapore		

```

<table>
<tr>
<td class="a"><%=airline %></td><!-- 항공사 -->
<td class="b"><%=svo.getDepartTime() %></td><!-- 출발시간 -->
<td rowspan="2" class="c">
<br>
<p class="time">5시간</p>
</td><!-- 결원시간 -->
<td class="d"><%=svo.getArriveTime() %></td><!-- 도착시간 -->
<td class="f"><%=svo.getId() %></td>
<td rowspan="2" class="e">₩<%=svo.getPrice() %><br> <br>
<input type="button" value="선택" class="checkBtn"><br> <br>
잔여좌석 <%=svo.getSeat() %> / 5
</td>
</tr>
<tr>
<td class="f"><%=svo.getId() %></td>
<!-- 출발국가 -->
<td><%=depart_ct %></td>
<!-- 도착국가 -->
<td><%=arrive_ct %></td>

```

가는날

항공사  
 시아항공  
 출발시간  
 2020-10-01 10:00:00.0  
 도착시간  
 2020-10-01 13:00:00.0  
 가격  
 550000  
 오는날  
 항공사  
 출발시간  
 도착시간  
 가격  
 인원  
 1  
 결제하기

2. 클릭한 테이블의 행의 값

```

$(document).on("click", ".checkBtn", function () {
  var checkBtn = $(this);

  // checkBtn.parent() : checkBtn의 부모는 <td>이다.
  // checkBtn.parent().parent() : <td>의 부모이므로 <tr>이다.
  var tr = checkBtn.parent().parent();
  var td = tr.children();

  // console.log("클릭한 Row의 모든 데이터 : "+tr.text());
  var airline = td.eq(0).text(); // 항공사
  var depart = td.eq(1).text(); // 출발시간
  var arrive = td.eq(3).text(); // 도착시간
  var ss = td.eq(4).text(); // 스케줄id
  var price = td.eq(5).text(); // 가격, 좌석

  $("#departTime2").val(depart); // 출발시간
  $("#arriveTime2").val(arrive); // 도착시간
  $("#de_air").val(airline); // 항공사
  $("#price2").val(price.substring(1, 7)); // 가격
  $("#sID2").val(ss); // 스케줄id
})

```

3. '선택'버튼을 클릭하면 클릭한 테이블의 행의 값들을 가져와서 넣어줌

4. '결제하기'버튼을 클릭하면  
입력된 값들을 결제페이지로 넘겨줌

## 2-1) 항공권 조회

1. 장바구니 버튼을 클릭하면 선택한 스케줄 값을 db에 넣어줌

가는 날 오는 날

시아항공	2020-10-01 10:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩550000	선택
	Korea		Singapore		장바구니
밍밍항공	2020-10-01 09:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩540000	선택
	Korea		Singapore		장바구니
사쿠라항공	2020-10-01 08:00:00.0	→ 5시간	2020-10-01 13:00:00.0	₩530000	선택
	Korea		Singapore		장바구니

```
$(document).on("click", ".cart", function () {
    var cartBtn = $(this)
    var tr = cartBtn.parent().parent();
    var td = tr.children();
    var ss = td.eq(0).text(); // 스케줄id
    console.log(ss)
    $.ajax({
        url: "cartInsert.jsp",
        data: {
            count: <%=seat%>,
            sid: ss,
            uid: '<%=uID%>'
        },
        success: function() {
            alert("장바구니에 담겼습니다!")
        }
    })// ajax enx
})
```

2. 장바구니 db insert에 필요한 항목들을 ajax로 넘겨받아 insert

```
String count = request.getParameter("count");
String sid = request.getParameter("sid");
String uid = request.getParameter("uid");

CartDAO cartdao = new CartDAO();
CartVO cvo = new CartVO();
cvo.setisPaid(false);
cvo.setCartDate(sdf.format(now)); // 오늘 날짜
cvo.setCount(1); // 인원수
cvo.setsID(Integer.parseInt(sid)); // 스케줄 아이디
cvo.setuID(uid); // 유저 아이디
cartdao.c_insert(cvo);
```

결제하기

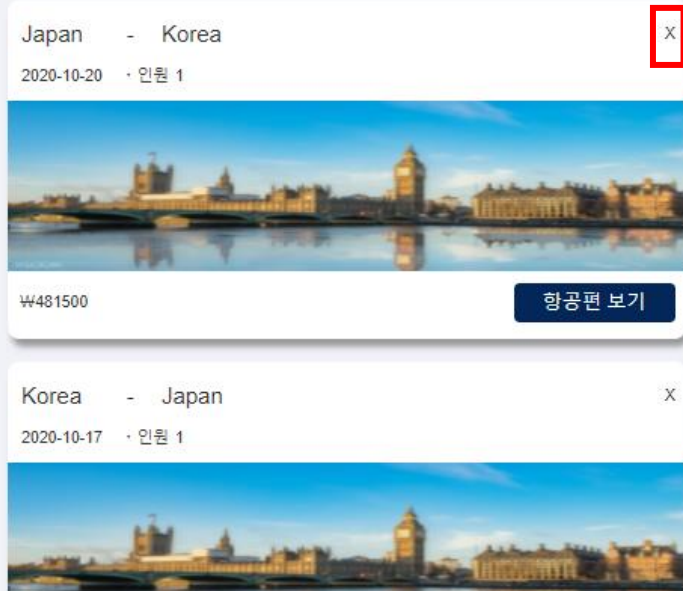
3. 장바구니는 결제를 하지않은 항목들이므로 isPaid는 false(0)로 들어감

123	cid	123	isPaid	clock	cartDate	123	count	123	sid	ABC	uid
	1		0		2020-10-09		1		1		a@a.com

## 1. 로그인 한 유저의 장바구니들만 가져옴

## 2-2) 장바구니

a@a.com



```
CartDAO dao = new CartDAO();
String cid = request.getParameter("cid");
dao.c_delete(Integer.parseInt(cid.trim()));
String uid = request.getParameter("uid");
List<CartVO> list = dao.all(uid);
```

3. 값을 넘겨받아 삭제하고자하는 항목을 삭제한 후 다시 select 함

```
<%
String uID = (String)session.getAttribute("uID");
CartDAO dao = new CartDAO();
List<CartVO> list = dao.all(uID);
%>

<body>
  <div id="all">
    <div id="uu"><%=uID %></div>
    <div id="tt">

      <%=for(int i=0; i<list.size(); i++) {
        CartVO vo = list.get(i);
        int depart = vo.getSvhdchedule().getDepartCtID();
        int arrive = vo.getSvhdchedule().getArriveCtID();
        CountryDAO countrydao = new CountryDAO();
      %>
```

```
$(document).on("click", ".rem", function () {
  var checkBtn = $(this);

  var tr = checkBtn.parent().parent();
  var td = tr.children();

  var cid = td.eq(1).text();
  var uid = $('#uu').text()

  $.ajax({
    url: "reCart.jsp",
    data:{
      cid: cid, // cart id
      uid: uid // user id
    },
    success: function(result) {
      $('#tt').empty()
      $('#tt').append(result)
    }
  })
});
```

2. 삭제 버튼을 클릭하면 삭제하고자 하는 장바구니 항목들의 값을 ajax로 보냄

## 2-2) 장바구니

### 1. 테이블 항목들을 empty()해서 비움

a@a.com

Korea - Japan

2020-10-17 · 인원 1



₩500900

항공

Korea - Vietnam

2020-10-20 · 인원 1



₩788100

항공

```
success: function(result) {
    $('#tt').empty()
    $('#tt').append(result)
}
```

### 1. 삭제하고 난 후 다시 select된 항목들을 append()하여 화면에 출력

```
<table class="cct">
  <tr>
    <td class="title">
      <input type="text" name="departCtID" value="<%=countrydao.nation(depart) %>" readonly="readonly">
      <br>
      <input type="text" name="arriveCtID" value="<%=countrydao.nation(arrive) %>" readonly="readonly" id="arriveCtID">
    </td>
    <td class="hh"><%=vo.getCtID()%></td>
    <td id="de">
      <input type="button" class="rem" value="X">
    </td>
  </tr>
  <tr>
    <td class="da"><input type="text" name="departTime" value="<%=vo.getSvhedule().getDepartTime().substring(0, 10)%>" readonly="readonly">
    <input type="hidden" name="arriveTime" value=" " readonly="readonly">
    <input type="text" name="seat" value="<%=vo.getCount() %>" readonly="readonly" id="seat">
    <td></td>
    <td style="color: #FFFFFF"><input type="hidden" name="way" value="1"></td>
  </tr>
  <tr class="ii">
    <td class="cct_i" colspan="3">
      
    </td>
  </tr>
  <tr>
    <td class="price"><%=vo.getSvhedule().getPrice() %></td>
    <td><input type="submit" value="항공권 보기" class="sel"></td>
  </tr>
</table>
```

# 3-1) 항공권 결제 (항공권상세정보)

항공권 상세 정보

가는날

대한민국

→ 다낭항공

일본

인원: 1

2020-10-16

15:00

일본

가는날

2020-10-16

10:00

대한민국

→ 5시간

오는날

일본

→ 새우리항공

대한민국

인원: 1

2020-10-17

08:00

대한민국

오는날

2020-10-17

03:00

일본

→ 5시간

탑승객 정보 입력

탑승객1

이름

이름을 입력해 주세요

성별

성별선택

생년월일

생년월일 8자리 입력 ex) 19900305

여권번호

여권 번호를 입력해 주세요

국적

국적선택

연락처

연락처를 '-' 없이 입력해 주세요 ex) 01012341234

입력정보확인

JSP

```
int psg = Integer.parseInt(request.getParameter("seat")); //인원수
int sID2 = Integer.parseInt(request.getParameter("sID2")); //가는날 스케줄 ID

//세션에 넣음
session.setAttribute("psg", psg);
session.setAttribute("sID2", sID2);

//가는날 스케줄 ID로 schedule 테이블 read
scheduleDAO dao = new scheduleDAO();
scheduleVO vo = dao.read(Integer.parseInt(request.getParameter("sID2")));
```

```
<!-- way 1은 편도.. 편도일 경우 실행 X -->
<% if(Integer.parseInt(request.getParameter("way"))==1){ %>

<!-- 왕복일 경우 -->
<%}else{
    //오는날 스케줄 ID로 DB연결 후 read
    int sID3 = Integer.parseInt(request.getParameter("sID3"));
    session.setAttribute("sID3", sID3);
    scheduleVO vo2 = dao.read2(Integer.parseInt(request.getParameter("sID3")));
}%>
```

```
//가는날 스케줄 ID로 country 테이블의 나라명 read
scheduleVO deNation= dao.deNation(Integer.parseInt(request.getParameter("sID2")));
scheduleVO arrNation = dao.arrNation(Integer.parseInt(request.getParameter("sID2")));
```

```
// 출발나라
public scheduleVO deNation(int sID1) throws Exception {
    con = dbcp.getConnection();
    String sql = "SELECT country.nation FROM country INNER JOIN schedule ON country.ctID = schedule.departCtID WHERE schedule.sID = ?";

// 도착나라
public scheduleVO arrNation(int sID1) throws Exception {
    con = dbcp.getConnection();
    String sql = "SELECT country.nation FROM country INNER JOIN schedule ON country.ctID = schedule.arriveCtID WHERE schedule.sID = ?";
```

DAO

```
// 가는날 스케줄
public scheduleVO read(int sID1) throws Exception {
    con = dbcp.getConnection();
    String sql = "select * from schedule where sID = ?";

    PreparedStatement ps = con.prepareStatement(sql);

    ps.setInt(1, sID1);
```

```
// 오는날 스케줄
public scheduleVO read2(int sID2) throws Exception {
    con = dbcp.getConnection();
    String sql = "select * from schedule where sID = ?";

    PreparedStatement ps = con.prepareStatement(sql);

    ps.setInt(1, sID2);

    ResultSet rs = ps.executeQuery();
    System.out.println("4. SQL문 전송 성공!!");

    scheduleVO bag = new scheduleVO();

    if (rs.next()) { // 결과가 있는지 없는지 체크
```

DB(schedule)

123 sID	departTime	arriveTime	123 airline	123 price	123 seat	123 departCtID	123 arriveCtID
63	2020-10-16 10:00:00.0	2020-10-16 15:00:00.0	9	501,400	5	10	50
96	2020-10-17 03:00:00.0	2020-10-17 08:00:00.0	8	400,000	5	50	10

DB(country)

123 ctID	ABC nation
10	대한민국
50	일본

선택한 항공권 값으로 DAO에서 선택 값에 맞는 DB에  
연결 후 가져온 항공권 정보를 화면에 출력

# 3-1) 항공권 결제 (탑승객 정보입력)

항공권 상세 정보

가는날

대한민국

가는날: 2020-10-16

10:00

대한민국

다낭항공

→ 5시간 →

일본

인원: 1

2020-10-16

15:00

일본

오는날

일본

오는날: 2020-10-17

03:00

일본

사쿠라항공

→ 5시간 →

대한민국

인원: 1

2020-10-17

08:00

대한민국

탑승객 정보 입력

① 탑승객1

이름

박경민

성별

남

생년월일

19900305

여권번호

KM9900999

국적

대한민국

연락처

01090909090

\* 모든 정보가 잘 입력되었습니다.

결제 페이지로 >

```

$('#psgForm').submit(function(event) {

    event.preventDefault(); //form submit 중지

    //결제 페이지 이동 컨펌 (확인, 취소)
    var result = confirm("결제페이지로 이동 하시겠습니까?");

    // 확인 클릭시, 각 input의 조건
    // 조건 안맞을 시 alert & 조건 안맞는 input으로 focus
    if ($('#p_name').val()=='') {
        $('#p_name').focus();
        alert('이름을 입력해주세요.');
```

input 입력 후 버튼 클릭 시 결제페이지로 이동 확인 confirm 창 뜸.  
input의 값이 맞지 조건과 같지 않은 경우 alert로 알림 & focus로 해당 input으로 이동  
모든 조건이 맞을 경우 버튼 컬러,텍스트 변경 되며 변경 된 버튼 클릭 시 결제 페이지로 이동

# 3-1) 항공권 결제

JSP

//API 연결

```
var IMP = window.IMP;
IMP.init('imp25919458');
```

//확인 후 모든 조건 만족 시 결제 API 연결

```
}else if (payok) {
    IMP.request_pay({
        pg : 'inicis',
        pay_method : 'card',
        merchant_uid : 'merchant_' + new Date().getTime(),
        name : 'Flyscanner 항공권',
        amount : $('#money').text(), //최종 결제 금액 값 가져오기
        buyer_email : 'iamport@siot.do',
        buyer_name : '구매자이름',
        buyer_tel : '010-1234-5678',
        buyer_addr : '서울특별시 강남구 삼성동',
        buyer_postcode : '123-456',
        m_redirect_url : 'https://www.yourdomain.com/payments/complete'
    }, function(rsp) {
        if ( rsp.success ) {
            var msg = '결제가 완료되었습니다.';
            msg += '고유ID : ' + rsp.imp_uid;
            msg += '상점 거래ID : ' + rsp.merchant_uid;
            msg += '결제 금액 : ' + rsp.paid_amount;
            msg += '카드 승인번호 : ' + rsp.apply_num;

        } else {
            //가상 결제: 임의로 결제창 종료시 결제완료
            $('#formm').unbind('submit'); //submit 중지 이벤트 해제
            //버튼 컬러, 텍스트 변경 후 클릭 -> submit
            $('#paybutton').text('결제 완료 하기 »');
            $('#paybutton').css("background", "#00b359");
        }
    });
}
```

결제

최종결제정보

인원 1명  
최종결제금액 901400원

카드 정보 입력

카드사

카드선택

할부기간

일시불

카드번호

16자리입력 0000-0000-0000-0000

비밀번호

유효기간

비밀번호  MM  YY

최종결제금액: 901400원

결제하기 »

결제

최종결제정보

인원 1명  
최종결제금액 901400원

카드 정보 입력

NICEPAY

SK pay

산한 KB국민 삼성 비씨(메이백)

롯데 현대 하나 NH재중

씨티 우리 카카오뱅크 +더보기

☐ 포인트 ☒ 무이자

\* 카드사별 무이자 할부가능 개월 수 상이  
\* 무이자할부 제외: 할인/가입/체크/선물/GIFT/분행계열카드

다음

상품명  
Flyscanner 항공권  
제기간  
별도제공기간없음  
상품금액  
901,400원

최종결제금액  
901,400 원

최종결제금액: 901400원

결제 완료 하기 »

모든 입력 조건 만족 시 아임포트 결제 API 연동



# 3-1) 항공권 결제

결제 완료 페이지로 넘어가면서  
모든 정보 순차적으로 DB에 저장

JSP

```
//payment 테이블 저장
paymentVO vo = new paymentVO();
paymentDAO dao = new paymentDAO();
int cardCom1 = Integer.parseInt(request.getParameter("cardCom"));
String cardNum = request.getParameter("cardNum");
String payPlan1 = request.getParameter("payPlan");
String uID = (String)session.getAttribute("uID");

vo.setCardCom(cardCom1);
vo.setCardNum(cardNum);
vo.setPayPlan(payPlan1);
vo.setuID(uID);

dao.create1(vo);

//booking 저장
paymentVO vo_b = new paymentVO();
paymentDAO dao_b = new paymentDAO();

//booking 저장(가난날)
int sID = (int)session.getAttribute("sID2");
int pID = dao_b.readpID(uID); //pID 불러오기

vo_b.setsID(sID);
vo_b.setuID(uID);
vo_b.setpID(pID);

dao_b.createBooking(vo_b);

//bookingDetail 저장(가난날)
String psgName = request.getParameter("psgName");
String psgBirth = request.getParameter("psgBirth");
String mobile = request.getParameter("mobile");
String passport = request.getParameter("passport");
String psgGender = request.getParameter("psgGender");
int country = Integer.parseInt(request.getParameter("country"));
int sID1 = (int)session.getAttribute("sID2");
int bID = dao_b.read(uID); //booking ID 불러오기
```

DAO & DB

1

```
// payment 테이블 저장
public boolean create1(paymentVO vo) throws Exception {
    con = dbcp.getConnection();
    // 3. sql문을 만든다(create)
    String sql = "insert into payment values(null,?,?,?,now(),?)";
    PreparedStatement ps = con.prepareStatement(sql);
    payment
```

	123 pID	123 cardCom	ABC cardNum	ABC payPlan	payTime	ABC uID
94	109	2	1234123412341234	3개월	2020-10-14 18:58:11.0	d@d.com

2

```
// payment pID 불러오기
public int readpID(String uID) throws Exception {
    System.out.println(uID);
    con = dbcp.getConnection();
    // 3. sql문을 만든다(create)
    String sql = "SELECT * FROM payment where uID = '" + uID.trim() + "' ORDER BY pID DESC LIMIT 1";
```

3

```
// booking 테이블 저장
public boolean createBooking(paymentVO vo) throws Exception {
    con = dbcp.getConnection();
    // 3. sql문을 만든다(create)
    String sql = "insert into booking values(null,now(),?,?,?)";
    booking
```

	123 bID	bookDate	123 sID	ABC uID	123 pID
179	204	2020-10-14	63	d@d.com	109
180	205	2020-10-14	96	d@d.com	109

4

```
// booking bID 불러오기
public int read(String uID) throws Exception {
    System.out.println(uID);
    con = dbcp.getConnection();
    // 3. sql문을 만든다(create)
    String sql = "SELECT * FROM booking where uID = '" + uID.trim() + "' ORDER BY bID DESC LIMIT 1";
```

5

```
// bookingdetail 테이블 저장
public boolean createBD(paymentVO vo) throws Exception {
    con = dbcp.getConnection();
    // 3. sql문을 만든다(create)
    String sql = "insert into bookingdetail values(null,?,?,?,?,?,?,?)";
    bookingdetail
```

	123 bID	ABC psgName	ABC psgGender	ABC psgBirth	ABC psgPhone	ABC psgPassport	123 bID	123 cID
176	189	박경민	M	19900305	01090909090	KM9900999	204	10
177	190	박경민	M	19900305	01090909090	KM9900999	205	10



## 3-1) 항공권 결제

### JSP

```
//카드아이디
if(session.getAttribute("cID")==null){

}else{
int cID = (int)session.getAttribute("cID");
//카드 isPaid 값 1로 바꾸기 (isPaid 1은 카트에서 보이지 않음)
paymentVO cIDvo = new paymentVO();
paymentDAO cIDdao = new paymentDAO();
cIDvo.setcID(cID);
cIDdao.cIDisPaid(cIDvo);
}
```

장바구니에 있는 항공권을 결제 한 경우  
장바구니 ID로 DB cart테이블의 값 변경

### DAO

```
// 장바구니 값 변경
public void cIDisPaid(paymentVO vo) throws Exception {

    con = dbcp.getConnection();

    String sql = "update cart set isPaid=1 where cID = ?";
```

### DB

Enter a SQL expression to filter results (use Ctrl+Space)

	123 cID	123 isPaid	cartDate	123 count	123 slID	ABC ulID
1	1	0	2020-10-10	1	1	a@a.com
2	2	1	2020-10-10	1	1	a@a.com

# 3-1) 항공권 결제 (내역 확인)

## 결제 내역 확인

결제 ID

d@d.com

이름

박경민

결제 내역

1. 결제일시: 2020-10-14 18:49:39	카드번호:1001200230033030	할부기간:3개월	상세보기
2. 결제일시: 2020-10-14 18:50:45	카드번호:8809090002221111	할부기간:2개월	상세보기
3. 결제일시: 2020-10-14 18:52:27	카드번호:7708086652231111	할부기간:4개월	상세보기
4. 결제일시: 2020-10-14 18:53:08	카드번호:5645545423111111	할부기간:2개월	상세보기
5. 결제일시: 2020-10-14 18:58:11	카드번호:1234123412341234	할부기간:3개월	상세보기

DB

! payment | Enter a SQL expression to filter results (use Ctrl+Space)

	123 pID	123 cardCom	abc cardNum	abc payPlan	payTime	uid
90	105	1	1001200230033030	3개월	2020-10-14 18:49:39.0	d@d.com
91	106	4	8809090002221111	2개월	2020-10-14 18:50:45.0	d@d.com
92	107	4	7708086652231111	4개월	2020-10-14 18:52:27.0	d@d.com
93	108	2	5645545423111111	2개월	2020-10-14 18:53:08.0	d@d.com
94	109	2	1234123412341234	3개월	2020-10-14 18:58:11.0	d@d.com

유저 ID로 DB의 payment 테이블에서  
해당 ID의 결제 건 vo에 담아 jsp에서 반복문으로 출력

JSP

```
<%
String uID = (String) session.getAttribute("uID"); //유저ID

mypayDAO dao = new mypayDAO();
mypayVO vo = new mypayVO();
//유저ID로 한글이름 불러오기
String name = dao.nameKor(uID);
//유저ID로 결제 건 불러오기
ArrayList<mypayVO> list = dao.payment(uID);
%>

//결제건 수 만큼 반복하며 출력
for (int i = 0; i < list.size(); i++) {
    mypayVO vo = list.get(i);
%>

<form action="mypay2.jsp" method="get">
    <span><%=i + 1%>. </span><span>결제일시: </span><%=vo.getPayDay()%>
    &ensp;
    <%=vo.getPayTime()%>
    &ensp;&ensp;<span>카드번호:</span><%=vo.getCardNum()%>
    <span>&ensp;&ensp;할부기간:</span><%=vo.getPayPlan()%>
    &ensp; <input name="pID" type="hidden" value="<%=vo.getpID()%>">
    <button
        style="background: #FEBB02; color: white; width: 150px; border-radius: 5px;">상
        세 보 기</button>
```

DAO

// 결제내역불러오기 ArrayList

```
public ArrayList<mypayVO> payment(String uID) throws Exception {
```

```
    ArrayList<mypayVO> list = new ArrayList();
    con = dbcp.getConnection();
    System.out.println("2. db연결 성공!!");
```

```
    String sql = "select pID, cardNum, payPlan, payTime from payment where uID = ?";
```

```
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setString(1, uID);
```

# 3-1) 항공권 결제 (내역 상세 보기)

JSP

결제 내역 상세 보기

```
<% //Time 포맷
SimpleDateFormat f = new SimpleDateFormat("HH:mm");

String uID = (String) session.getAttribute("uID"); //유저 ID
int pID = Integer.parseInt(request.getParameter("pID")); //전 페이지에서 선택한 결제내역 ID

mypayDAO dao = new mypayDAO();
mypayVO VO = new mypayVO();

//결제내역 ID로 스케줄 불러오기
ArrayList<mypayVO> list = dao.bookingsID(pID);
%>
```

DAO

```
//각 결제 건 스케줄
public ArrayList<mypayVO> bookingsID(int pID) throws Exception {

    ArrayList<mypayVO> list = new ArrayList();
    con = dbcp.getConnection();
    System.out.println("2. db연결 성공!!");

    String sql = "select departTime, arriveTime, departCtID, arriveCtID from schedule where sID in (select sID from booking where pID = ?)";

    PreparedStatement ps = con.prepareStatement(sql);
    ps.setInt(1, pID);
```

예약1.

출발: 2020-10-16 10:00

도착: 2020-10-16 15:00

출발지 → 도착지  
대한민국 → 일본

예약2.

출발: 2020-10-17 03:00

도착: 2020-10-17 08:00

출발지 → 도착지  
일본 → 대한민국

payment | Enter a SQL expression to filter results (use Ctrl+Space)

	123 pID	123 cardCom	ABC cardNum	ABC payPlan	payTime	ABC uID
93	108	2	5645545423111111	2개월	2020-10-14 18:53:08.0	d@d.com
94	109	2	1234123412341234	3개월	2020-10-14 18:58:11.0	d@d.com

DB

booking | Enter a SQL expression to filter results (use Ctrl+Space)

	123 bID	bookDate	123 sID	ABC uID	123 pID
179	204	2020-10-14	63	d@d.com	109
180	205	2020-10-14	96	d@d.com	109

schedule | Enter a SQL expression to filter results (use Ctrl+Space)

	123 sID	departTime	arriveTime	123 airline	123 price	123 seat	123 departCtID	123 arriveCtID
63	63	2020-10-16 10:00:00.0	2020-10-16 15:00:00.0	9	501,400	5	10	50
96	96	2020-10-17 03:00:00.0	2020-10-17 08:00:00.0	8	400,000	5	50	10

pID(결제 테이블 ID)-> bID(booking)  
->sID(schedule) ->시간,나라 값 가져옴

# 4-1) 여행지 추천

**데이터 모델:** 국민여행실태조사 2015~2017 총 3개년

**원본 데이터의 문제점:**

10,000개 이상의 row와 70여개의 column X 파일 9개  
무수히 많은 null값과 중복되는 column..

2017년 국민여행 실태조사  
유저가이드



원데이터와 함께 제공되는 유저가이드, 코드북을 참고하여  
<개인고유번호, 나라번호, 만족도>값을 중심으로  
데이터를 재구성하기로 결정

The images show three different views of the data. The top image is a wide Excel spreadsheet with columns labeled A through Z, containing various numerical and categorical data points. The middle image shows a more structured table with columns for user identification (HID, PID, sex, age) and travel-related information (M\_ID, M\_D, M\_T, M\_S, M\_E, M\_F, M\_G, M\_H, M\_I, M\_J, M\_K, M\_L, M\_M, M\_N, M\_O, M\_P, M\_Q, M\_R, M\_S, M\_T, M\_U, M\_V, M\_W, M\_X, M\_Y, M\_Z). The bottom image is another table with similar columns, showing a different set of data values for the same variables.

# 4-1) 데이터 전처리 1

## 1. csv 형식으로 변환한 엑셀파일을 jupyter notebook 에서 pandas import한 후 read\_csv()

```
In [4]: import pandas as pd

In [3]: person=pd.read_csv('data/person.csv')

In [4]: unit=pd.read_csv('data/unit.csv')

In [5]: spot=pd.read_csv('data/spot.csv')
```

## 2. 불러온 파일 출력해서 정보 확인

```
In [7]: unit.head(10)

Out[7]:
```

	hid	PID	type1	type2	Month	M_ID	q1	q2_a	q2_a_1	q2_a_2	...	q10_4	q
0	10001	1000101	1	2.0	8.0	115439.0	2.0	2017.0	8.0	4.0	...	4.0	
1	10001	1000102	1	2.0	12.0	115441.0	1.0	2017.0	12.0	23.0	...	9.0	
2	10002	1000201	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
3	10002	1000202	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
4	10002	1000203	1	2.0	10.0	123587.0	2.0	2017.0	10.0	3.0	...	3.0	
5	10003	1000301	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
6	10003	1000303	1	1.0	4.0	110795.0	2.0	2017.0	4.0	8.0	...	4.0	
7	13089	1000304	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
8	10004	1000401	1	1.0	4.0	110798.0	2.0	2017.0	4.0	28.0	...	4.0	
9	10004	1000402	1	2.0	10.0	123596.0	2.0	2017.0	10.0	16.0	...	4.0	

10 rows x 74 columns

```
In [9]: person.shape
```

```
Out [9]: (6170, 17)
```

```
In [10]: unit.shape
```

```
Out [10]: (11089, 74)
```

```
In [11]: spot.shape
```

```
Out [11]: (11305, 72)
```

rows X columns

```
In [19]: person.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6170 entries, 0 to 6169
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   HID          6170 non-null   int64
1   PID          6170 non-null   int64
2   sex          6170 non-null   int64
3   age          6170 non-null   int64
4   sr_type      6170 non-null   int64
5   side         6170 non-null   int64
6   ara_size     6170 non-null   int64
7   school1      6170 non-null   int64
8   school2      6170 non-null   int64
9   occ0         0 non-null      float64
10  occ1         6170 non-null   int64
11  occ2         6170 non-null   int64
12  fac          6170 non-null   int64
13  marry        6170 non-null   int64
14  incl_1       6170 non-null   int64
15  incl_2       6170 non-null   int64
16  wt           6170 non-null   float64
dtypes: float64(2), int64(15)
memory usage: 819.6 KB
```

일부만 다른 데이터 타입

```
In [22]: spot.isnull().sum()
```

```
Out [22]: hid      0
PID      0
type1     0
type2    3541
Month    3541
...
q6_7     3541
q6_8     3541
q6_9     3541
q7_2     3541
WT       3541
```

NULL 개수

```
Length: 72, dtype: int64
```

# 4-1) 데이터 전처리2

3. 해외여행에 대한 응답이 아닌 사람(이상치)과 중복인 사람 찾아서(중복값) 행 삭제

4. 개인고유번호(PID)를 중심으로 각 파일을 join한 후 필요한 column만 빼고 drop하여 index reset

```
In [62]: unit[unit['type1']!=2].index
Out[62]: Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
...,
11079, 11080, 11081, 11082, 11083, 11084, 11085, 11086, 11087,
11088],
dtype='int64', length=10365)

In [63]: idx=unit[unit['type1']!=2].index

In [64]: idx
Out[64]: Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
...,
11079, 11080, 11081, 11082, 11083, 11084, 11085, 11086, 11087,
11088],
dtype='int64', length=10365)

In [65]: unit.shape
Out[65]: (11089, 74)

In [66]: unit=unit.drop(idx)
```

```
In [175]: df=person[person['PID'].isin(unit['PID'])]
```

```
In [98]: all=all.drop(columns='type1')
```

5. 각 연도마다 파일 3개에서 1개로 줄이고, 3개년을 합쳐서 최종적으로 csv 파일 1개로 병합 후 저장

6. null값(결측값)인 column이 존재하면 분석 결과가 왜곡될 수 있기 때문에 값을 0으로 채워준다

```
In [7]: reunit=pd.merge(unit,person,how='left',on='pid')
```

```
In [56]: s2015=s2015.fillna(0)
```

```
In [89]: all=pd.concat([s5, s6, s7], axis=0)
```

```
In [91]: all.reset_index()
```

```
In [108]: df.index = df.index + 1
```

# 4-1) 데이터 전처리3

**7. 최종적으로 <사람, 나라, 만족도>로 이루어진 데이터로 재구성. 이를 바탕으로 유사도를 도출할 기준으로 각 나라의 테마를 정함(카테고리)**  
 → 자연경관, 휴양지, 야외 액티비티 등 13개의 기준을 가진 데이터프레임 생성 후 전처리 반복

```
In [2]: person=pd.read_csv('data/person_dataset.csv')
```

```
In [3]: person
```

```
Out [3]:
```

	pid	q6_1	q7
0	1	207	4
1	2	70	3
2	3	176	4
3	4	30	3
4	5	30	5
...	...	...	...
2136	2137	70	4
2137	2138	207	4
2138	2139	236	4
2139	2140	171	4
2140	2141	171	4

2141 rows x 3 columns

```
In [4]: theme=pd.read_csv('data/country_theme2.csv')
```

```
In [5]: theme
```

```
Out [5]:
```

	q6_1	name	nature	food	activity	historical	thermal	hotel	spa	shopping	recreation
0	1	가나	1.0	NaN	NaN	Nat					
1	8	광	1.0	NaN	1.0	Nat					
2	11	그리스	1.0	1.0	NaN	1.0					
3	16	나미비아	1.0	NaN	NaN	Nat					
4	18	나이지리아	1.0	NaN	NaN	Nat					
...	...	...	...	...	...	...					
82	230	피지	1.0	NaN	1.0	Nat					
83	231	핀란드	1.0	NaN	NaN	Nat					
84	232	필리핀	1.0	1.0	1.0	Nat					
85	235	헝가리	1.0	NaN	NaN	Nat					
86	236	홍콩	NaN	1.0	NaN	Nat					

87 rows x 15 columns

```
In [6]: df_rating
```

```
Out [6]:
```

	pid	cid	rating
0	1	207	4
1	2	70	3
2	3	176	4
3	4	30	3
4	5	30	5
...	...	...	...
2136	2137	70	4
2137	2138	207	4
2138	2139	236	4
2139	2140	171	4
2140	2141	171	4

2141 rows x 3 columns

## 4-1) 데이터 분석

### 1. 라이브러리와 사용할 파일 import 후 전처리

```
In [29]: import pandas as pd
import numpy as np
import glob
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPage
import time
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
from scipy import spatial
import operator
```

```
In [3]: df_rating=pd.read_csv('data/person_dataset.csv')
```

```
In [4]: df_country=pd.read_csv('data/country_theme2.csv')
```

```
In [5]: df=pd.read_csv('data/person_dataset2.csv')
```

【데이터 전처리】

```
df_rating=pd.read_csv('data/person_dataset.csv') # [pID, cID, rating] 사람들이 나라에 대해 평가한 만족도 파일
df_country=pd.read_csv('data/country_theme2.csv') # [cID, name, nature...] 나라를 테마별로 분류
```

# pivot: 행데이터를 열데이터로 회전시키는 것 --> 데이터 재구조화

```
df_country_feature=df_rating.pivot(
    index='cID', # 인덱스는 cID로 잡고
    columns='pID', # 행이었던 pID를 열로
    values='rating' # 열 값은 rating
).fillna(0) # 빈곳은 0으로 채우기
```

# M\*N형식으로 만들. M: 나라, N: 유저

# 값이 0과 1밖에 없으니까 희소행렬(sparse matrix)로 만들기

# 메모리가 효율적으로 저장된다고 함

```
mat_country_feature=csr_matrix(df_country_feature.values)
```



## 4-1) 추천 알고리즘 KNN

### [KNN 구현]

```
# KNN(K-Nearest Neighbor) 최근접 이웃법
# 새로운 데이터를 입력받았을 때 가장 가까이 있는 것이 무엇이냐를 중심으로 새로운 데이터의 종류를 정해주는 알고리즘
# K는 주변의 개수를 의미. 하지만 K값이 너무 크면 분류 효율이 떨어짐(범위가 커져서 이웃이 개나 소나 되는...)

# cID(나라)를 중심으로 size(출현횟수), mean(평균)을 구함
countryProperty=df.groupby('cID').agg({'rating':[np.size, np.mean]})
countryNumRating=pd.DataFrame(countryProperty['rating']['size'])

# rating 점수를 scalar minmax 정규화
# 정규화: norm으로 평균을 나누어주는 작업을 의미한다. 데이터 내 모든 관측치들을 0과 1 사이에 위치하도록
# 만들어주기 위해 재조정해주는 작업. 기준 개념을 만들겠다는 뜻! 모든 데이터들을 동일한 정도의 스케일(중요도)로 만든다.
# 숫자로 된 관측치들을 가지고 있는 벡터 값을 동일한 척도(0~1)사이로 배열하는 결과가 나타남
# minmax 공식:  $X(\text{norm}) = (X - X(\text{min})) / (X(\text{max}) - X(\text{min}))$ 
# 서로 다른 척도로 측정된 변수를 그대로 사용한다면 같은 선상에서 비교하는 것은 무리일것(예: 연령, 소득)
countryNormalizedNumRating=countryNumRating.apply(lambda x: (x-np.min(x)) / (np.max(x)-np.min(x)))
```

## 4-1) 추천 알고리즘 KNN

```
# 나라리스트를 담은 셋 생성
countryDic={}
for index,row in df_country.iterrows(): # iterrows: 행의 열을 하나씩 접근해줌. itertuples()가 더 빠르다른 소리가..
    cID=int(row['cID'])
    name=row['name']
    theme=list(row[2:]) # 앞에 0,1은 이미 지정했으니까 2부터 끝까지가 theme라서 [2:]라고 표현함
    countryDic[cID] = (name, np.array(list(theme)), countryNormalizedNumRating.loc[cID].get('size'), countryProperty.loc[cID].rating.get('mean'))
# countryDic에는 나라이름, theme array, 정규화된 rating 점수, 평균 rating 점수
```

```
# 유사도 측정
def ComputeDistance(a, b):
    themeA = a[1] # theme array가 들어옴
    themeB = b[1] # theme array가 들어옴
    themeDistance = spatial.distance.cosine(themeA, themeB) # cosine..연산을 하는 까닭은?..ㅋㅋ https://wikidocs.net/24603
    popularityA = a[2] # 정규화된 rating 점수
    popularityB = b[2] # 정규화된 rating 점수
    popularityDistance = abs(popularityA - popularityB) # 둘의 차이를 절대값으로..
    return themeDistance + popularityDistance # 둘을 구해서 더한게 유사도..각 나라에 대해서 유사도를 구할 수 있다
```

## 4-1) 추천 알고리즘 KNN

```
# 유사도를 바탕으로 비슷한 나라 구하기
def getNeighbors(cID, K):
    distances = []
    for country in countryDic:
        if (country != cID): # 자기자신은 빼고 구하기
            dist = ComputeDistance(countryDic[cID], countryDic[country])
            distances.append((country, dist)) # 배열에 나라번호랑 거리 저장
    distances.sort(key=operator.itemgetter(1)) # 배열의 1번째값인 dist를 기준으로 sort
    neighbors = []
    for x in range(K):
        neighbors.append(distances[x][0])
    return neighbors
```

```
# 추천 목록
def recommend(cID, K):
    avgRating = 0
    print(countryDic[cID], '\n') # 일단 선택한 나라 정보 찍어주고
    neighbors = getNeighbors(cID, K) # 이웃 출력
    for neighbor in neighbors:
        avgRating += countryDic[neighbor][3] # 이걸 이웃들의 평균 점수
        print(countryDic[neighbor][0] + " " + str(countryDic[neighbor][3])) # str() 문자열
    avgRating /= K
    print("평균 Rating: ", avgRating)
```

## 4-1) 추천 알고리즘 결과

In [34]: `recommend(57,10)`

( '몰타', array([1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]), 0.0, 3.0)

스리랑카 4.0

파나마 4.0

피지 4.0

몰디브 3.8

라오스 3.869565217391304

뉴질랜드 4.0

괌 4.3478260869565215

사이프러스 5.0

팔라우 4.0

남아프리카공화국 4.5

\*\*평균 만족도: 4.151739130434782 \*\*

In [36]: `recommend(171,5)`

( '일본', array([1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1]), 1.0, 4.055670103092783)

중국 3.868327402135231

태국 4.081967213114754

미국 4.061224489795919

홍콩 3.8764044943820224

프랑스 4.276595744680851

\*\*평균 만족도: 4.032903868821755 \*\*

171 일본	1	1			1	1	1	1	1		1		1
176 중국	1	1		1	1	1		1	1		1		1

207 태국	1	1	1			1		1	1		1		1
--------	---	---	---	--	--	---	--	---	---	--	---	--	---

70 미국	1	1	1	1	1			1	1	1	1		1
-------	---	---	---	---	---	--	--	---	---	---	---	--	---

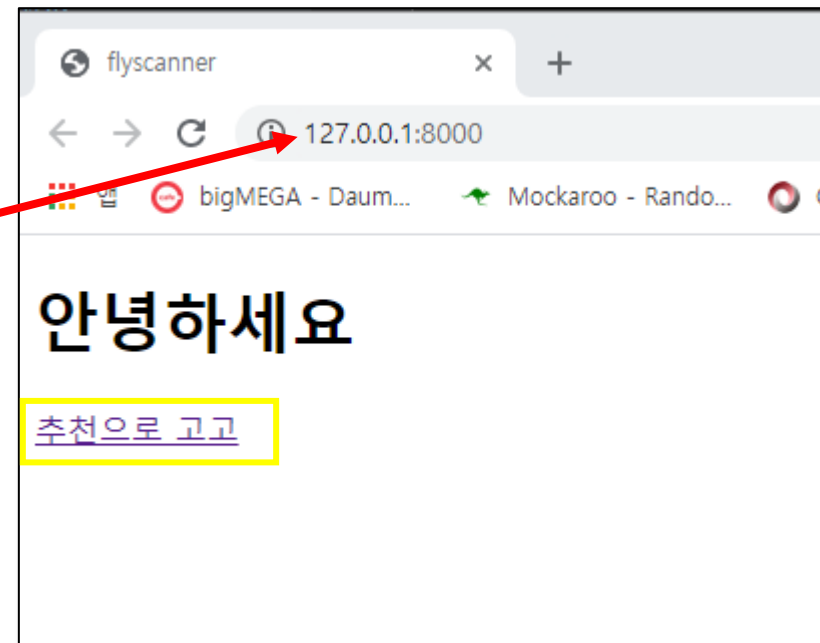
## 4-1) 로컬 서버

프로젝트

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', HomeView.as_view(), name='home'),  
    path('recommender/', include('recommender.urls'))  
]
```

```
class HomeView(TemplateView):  
    template_name = 'home.html'
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>flyscanner</title>  
</head>  
<body>  
    <h1>안녕하세요</h1>  
    <a href="/recommender/">추천으로 고고</a>  
</body>  
</html>
```



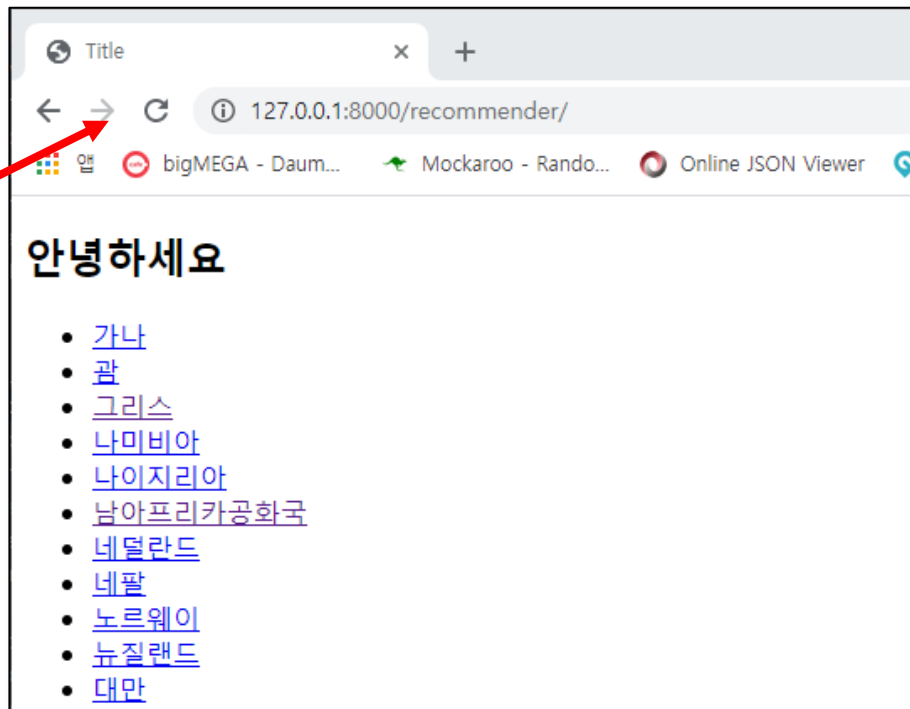
## 4-1) 로컬 서버

앱

```
urlpatterns = [  
    path('', views.index),  
    path('<int:cID>/', views.detail)
```

```
def index(request):  
    countries = Country.objects.all()  
    return render(request, 'recommender/index.html', {  
        'index': countries  
    })
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Title</title>  
</head>  
<body>  
    <h2>안녕하세요</h2>  
  
    <ul>  
        {% for c in index %}  
        <li><a href='{ c.cID }'>{{ c.name }}</a>  
        </li>  
        {% endfor %}  
    </ul>  
  
</body>  
</html>
```

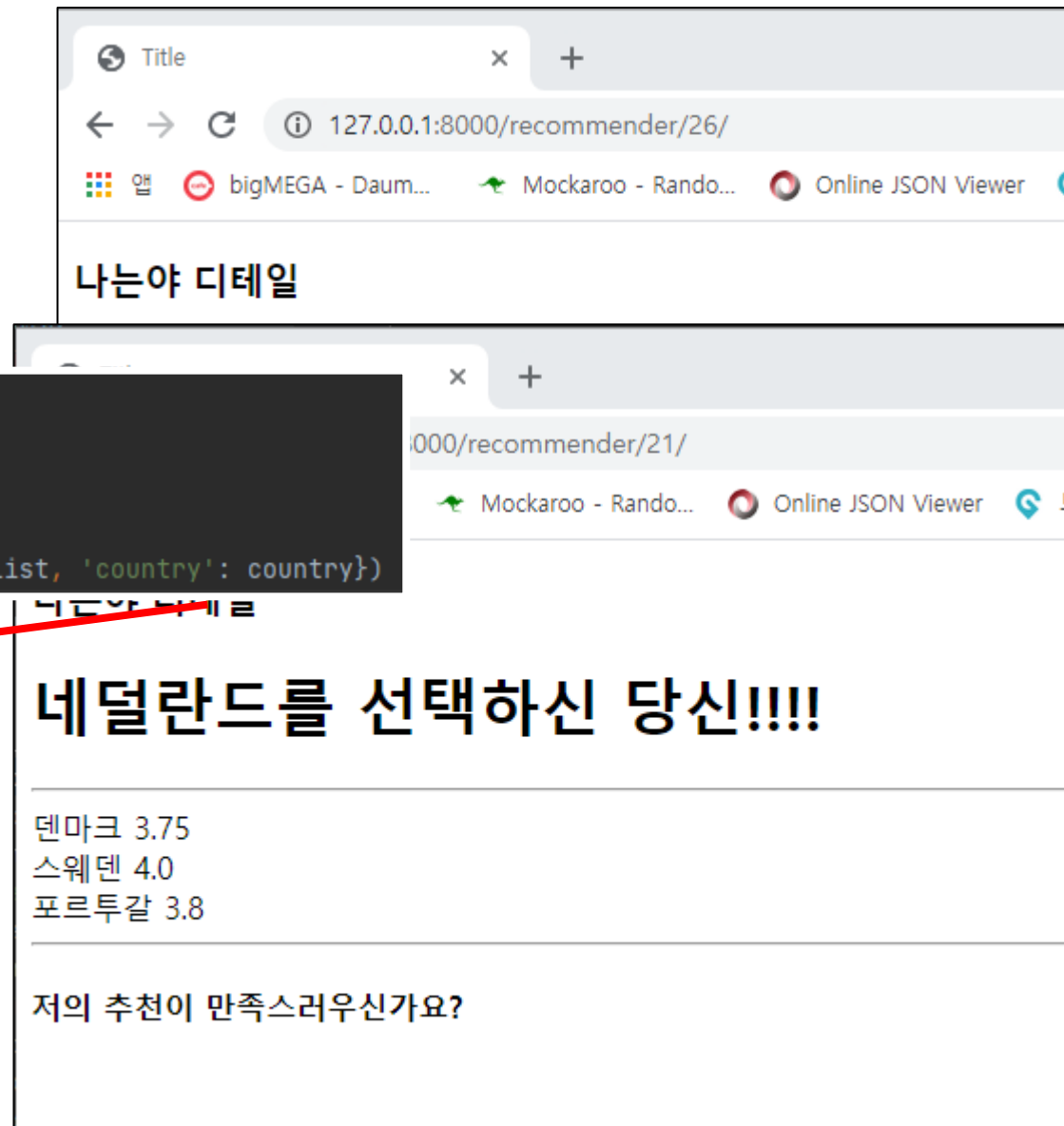


## 4-1) 로컬 서버

```
urlpatterns = [  
    path('', views.index),  
    path('<int:cID>/', views.detail)  
]
```

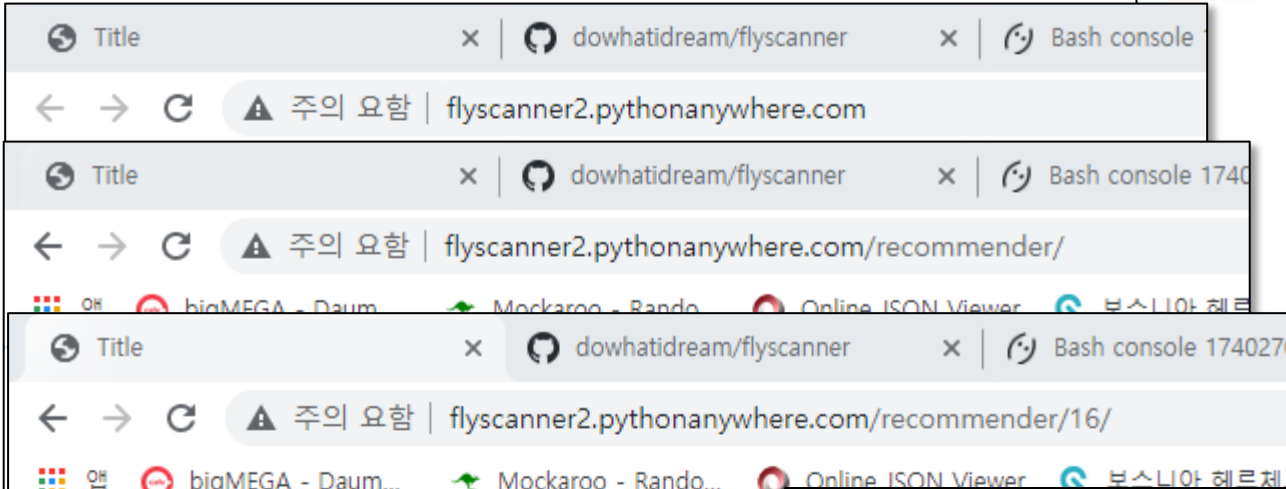
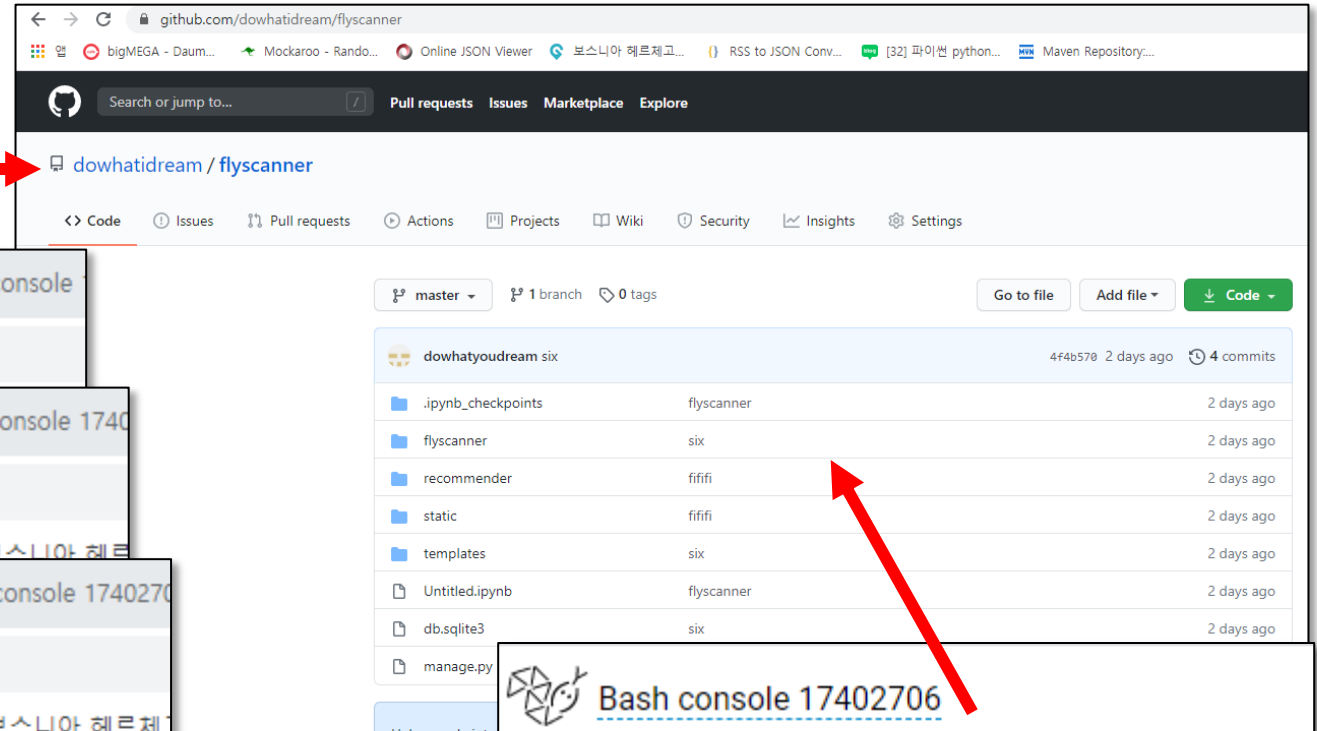
```
def detail(request, cID):  
    country = Country.objects.get(pk=cID)  
    recommend_list = recommend(cID, 3)  
  
    return render(request, 'recommender/detail.html', {'detail': recommend_list, 'country': country})
```

```
</head>  
<body>  
    <h3>나는야 디테일</h3>  
    <h1>{{ country }}를 선택하신 당신!!!!</h1>  
    <hr>  
  
    {% for con in detail %}  
        {{ con }}<br>  
    {% endfor %}  
    <hr>  
    <h4>저의 추천이 만족스러우신가요?</h4>  
</body>  
</html>
```



# 4-1) 웹 서버

```
(venv2) D:\LSY\python\workspace4\flyscanner>git push
```



나는야 디테일

나미비아

pythonanywhere

**Warning** You have not confirmed your email address yet. This means that you will not be able to receive important emails anymore, send yourself a new one [here](#).

flyscanner2.pythonanywhere.com

+ Add a new web app

Configuration for flyscanner2.pythonanywhere.com

Reload:

Reload flyscanner2.pythonanywhere.com

Bash console 17402706

```
00:44 ~ $ cd flyscanner
00:44 ~/flyscanner (master)$ git pull
Already up-to-date.
00:44 ~/flyscanner (master)$
```



## 4-2) 여행지 조회

### 어디로 가고 싶으세요?

지역

☐아프리카 ☐아시아 ☒남아메리카 ☒유럽 ☐오세아니아 ☐북아메리카

여행테마

☐음식관광 ☒액티비티 ☐역사유적지 ☒테마파크/국립공원 ☒호캉스 ☐온천/스파 ☐쇼핑 ☐스포츠경기 관람 ☐지역축제 ☐종교/성지순례 ☐도시투어 ☐섬 ☐문화예술/공연

검색

볼리비아 브라질 아르헨티나 칠레 페루 러시아 체코 크로아티아 폴란드 헝가리 그리스 네덜란드 덴마크 독일 스웨덴 스위스  
스페인 영국 이탈리아 터키 포르투갈 프랑스 핀란드 노르웨이 보스니아 헤르체고비나 슬로바키아 오스트리아

```
$.ajax({
  url: "menu_recommend/searching.jsp",
  data: {
    continent: conList,
    theme: themeList
  },
  traditional: true,
  success : function(data) {
    list = data.split(',')

    for(var i=0; i < list.length-1; i++) {
      $("#result2").append("<button id='btnCountry'>" + list[i] + "</button>&nbsp;&nbsp;&nbsp;");
    }
  }
})
```

```
7 ArrayList<String> data = dao.read3(continent, theme);
8 for(String s : data){
9 %><%=s%>, <%}%>
```

```
$("#input[name='continent']:checked").each(function(){
  conList.push($(this).val())
})

$("#input[name='theme']:checked").each(function(){
  themeList.push($(this).val())
})
```

## 4-2) 여행지 조회

### 어디로 가고 싶으세요?

지역

☐아프리카 ☐아시아 ☒남아메리카 ☒유럽 ☐오세아니아 ☐북아메리카

여행테마

☐음식관광 ☒액티비티 ☐역사유적지 ☒테마파크/국립공원 ☒호캉스 ☐온천/스파 ☐쇼핑 ☐스포츠경기 관람 ☐지역축제 ☐종교/성지순례 ☐도시투어 ☐섬 ☐문화예술/공연

검색

볼리비아 브라질 아르헨티나 칠레 페루 러시아  
스페인 영국 이탈리아 터키 포르투갈 프랑스

```
$(document).on("click", "#btnCountry", function (e) {  
    console.log(e)  
    name = $("#btnCountry").text()  
    document.getElementById("arriveCtID").value = name  
});
```

출발국가

라오스

왕복

편도

출발일

도착일

1

검색

# 1) 장민성

2차 프로젝트를 통해 진도를 따라가지 못하고 많이 뒤떨어져서 할 수 없을 줄 알았지만 팀 분들이 많이 도와주시고 많은 설명을 해주시고 많이 이끌어 주셔서 이번 프로젝트 에서 포인트는 얻지 못했지만 1차 프로젝트 때 얻지 못한 것, 이 해하지 못한 것 이해를 남들보다 느리지만 하고 있다는 것을 느꼈다.

앞으로도 열심히 더 잘 해보고 싶다.

## 2) 김정하

구글 지도 API를 단순히 지도로 출력하는 것으로는 써봤지만 마커를 찍고 곡선을 연결하는 것은 처음 사용해봐서 흥미로웠습니다. 다음 프로젝트부터는 구글 지도 API의 부가적인 기능을 조금 더 세 부적으로 사용해보고 싶습니다.

이번 프로젝트에서 ajax를 처음 사용해보았습니다. ajax는 페이지를 다시 로드 하지 않고 필요한 부분만 다시 불러와 사용하여 자원과 시간을 낭비하지 않는다는 점이 좋았습니다. 필요한 데이터만 받아 갱신하다 보니 페이지의 속도를 향상시킬 수 있었습니다. 조회 페이지의 sort나 장바구니 페이지에서 항목을 삭제 한 후 다시 로드할 때 빠르게 화면에 출력할 수 있었습니다.

팀원들과 소통이 잘되어서 서로서로 도움을 주고 받으며 프로젝트를 수월하게 진행할 수 있었습니다. 조회와 바로 이어지는 결제를 담당하시는 경민씨와 중간중간에 대화와 서로의 코드를 합쳐보며 오류를 빠르게 처리할 수 있었습니다.

## 3) 박경민

2차 프로젝트 진행 시 여러 테이블이 연결되어 있는 DB를 처음 써보며 초반에 DB 설계 의도와 구성을 이해 하는데 어려움을 겪었으나 조원 분들의 도움과 DB를 여러 번 확인하고 검색해보는 과정을 통해 어려움을 해결할 수 있었다.

연결되어 있는 여러 DB 테이블을 사용하기 위해 많은 sql문을 검색하고 사용하고 시도해보며 sql문을 이해하는데도 많은 도움이 되었다.

## 4) 이승연

### Garbage In, Garbage Out(GIGO)

"쓰레기가 들어가면 쓰레기가 나온다"는 뜻으로 컴퓨터 과학이나 정보통신기술 분야에서 컴퓨터가 논리 프로세스에 의해 운영되기 때문에 결함이 있는, 심지어는 터무니없는 입력 데이터(쓰레기가 들어감)라도 의심을 품지 않고 처리하며, 생각하지도 않던 터무니없는 출력(쓰레기가 나옴)을 만들어낸다는 사실을 가리킨다. 이 원칙은 전제에 결함이 있다면 논증은 오류가 있을 수 있다는 점에서 모든 분석, 논리에 더 일반적으로 적용된다.

정보처리 자격증 공부를 하면서 뇌리에 깊게 박혔던 단어가 하나 있습니다. 공부할 때 인상 깊게 남았던 부분이 계속 되뇌곤 했는데, 이번 프로젝트에서 추천 알고리즘을 구현하며 비로소 단어의 뜻을 제대로 이해할 수 있었습니다. 범람하는 데이터 속에서 쓸만한 것을 찾아 원하는 결과를 얻기 위해 가공하는 일이 얼마나 많은 인사이트를 요구하는지 깨달을 수 있었던 계기가 되었습니다.

예약이라는 주제를 구현한 것은 이번이 처음이어서 구현 시 패러독스가 일어나지 않도록 설계단부터 많이 고민했습니다. 또한 오류를 최초로 발견한 사람만 고치고 수정안을 배포하는 것이 아니라 다른 팀원들과 오류라고 생각하는 이유, 이에 대한 올바른 수정 방향에 대해 서로 의견을 나누며 진행하였더니 한층 더 수월하게 진행할 수 있었습니다. 작업은 힘들었지만..즐겁게 할 수 있었던 프로젝트였습니다.