PSTM

PSTM(PerSonal Training at hoMe)

코더차야조(2조)

김선아
김지훈
박민석
박준범
배유진
조용승

목차

1. 주제 선정 및 주제 선정 동기

1) 주제
2) 주제 선정 동기
3) 목표 및 기대효과
4) 경쟁사 및 PSTM의 차별점

PSTM(PerSonal Training at hoMe)

PSTM(PerSonal Training at hoMe)은 온라인으로 pt 및 식단관리를 받을 수 있는 플랫폼이다

주제 선정 동기(– PSTM(PerSonal Training at hoMe)

다이어트와 건강한 몸 유지를 위해서는 운동과 식단 관리 모두가 필요하다.

처음 헬스에 입문한 사람들은 식단 관리와 운동에 어려움을 겪는 경우가 많기 때문에 식단관리와 운동 관리가 모두 포함되어 있는 pt 서비스를 이용하는 경우가 많다.

하지만 헬스장을 따로 찾을 시간이 없는 경우, 또는 현 코로나 시국과 같이 헬스장 이용에 어려움을 겪을 경우 부득이하게 집에서 운동을 진행해야 한다.

이러한 “홈트레이닝” 족들을 위해 헬스장에 가지 않고도 pt 및 식단관리를 받을 수 있는 플랫폼을 만들고자 위 주제를 정하게 되었다.

목표 및 기대효과

목표

온라인 pt 서비스를 기간에 따른 지정된 금액을 제시해 합리적인 가격으로 보급하고자 하며, 헬스장 이용이 어려운 사람들에게 서비스를 제공해 장소적인 제약에서 벗어날 수 있도록 한다.

또한 식단 관리를 온라인으로 진행함으로써, 오프라인으로 진행할 시 발생할 불필요한 시간 낭비를 줄이도록 도와준다

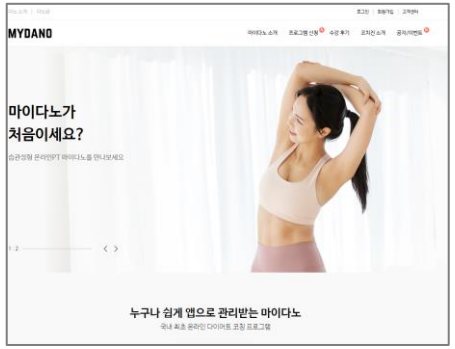
기대효과

직장인, 주부, 학생 등 시간/공간에 제약을 받고 사는 사람들도 운동관리/식단 관리를 받을 수 있을 것이다

경쟁사(시장 조사) 및 기존 제품과 PSTM의 차별점 – 마이다노, 모두의 트레이닝(어플)

마이다노

- 맞춤형 식단/맞춤형 운동 제공
- pt 코치(마이다노 직원)와 1대1 채팅으로 동기부여 훈련과 식단 상담을 받음
- 운동 기록에 기반한 맞춤형 코칭 제공(실시간으로 이루어지지는 않는다)
- 수강 결제 외 운동 기구/음식등을 판매한다
- 채팅 시 부족한 부분을 보완하기 위해 운동 영상을 찍어 보내도록 하며 운동영상 기반으로 자세의 교정을 진행한다



모두의 트레이닝(어플)

-회원의 상태에 맞는 홈 트레이닝 방법 제공, 끝나면 후기를 남기는 게시판 제공

-후기에 대한 코치의 피드백 제공

-식단은 코치가 보고 판단하도록 돼있다

기존 제품과 PSTM의 차별점

1) 식단 칼로리를 객체감지, 식품 api 를 이용해 계산해주어 회원들이 직관적으로 식단의 상태에 대해 알 수 있도록 하며, pt 트레이너의 피드백 제공에도 도움을 준다

2) 화상 채팅을 이용해 실시간으로 자세 교정을 진행할 수 있도록 한다

3) 오프라인 수업으로 전환할 경우를 대비해 pt 강사 등록 시 현재 위치와 가까운 트레이너들을 확인할 수 있도록 한다

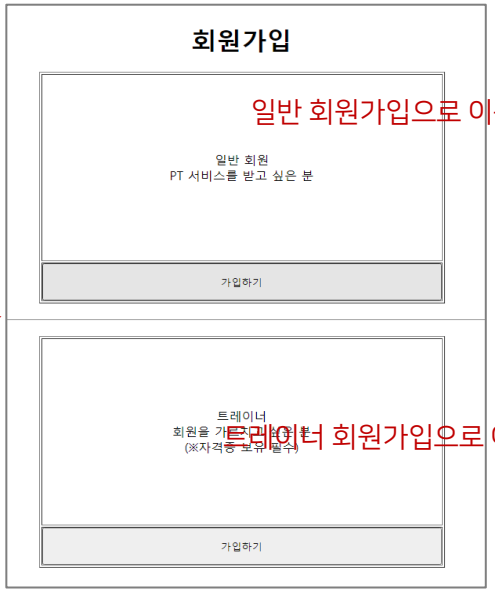
2. 기능 및 사용 기술

1) 회원가입 기능
2) 로그인 기능
3) 결제 기능
4) 지도 기능
5) 채팅 기능(챗봇)
6) 채팅 기능(멀티룸 채팅)
7) 마이페이지/식단 관리 기능
8) 1:1 강의 기능
9) 후기 게시판

회원가입 기능-화면 흐름

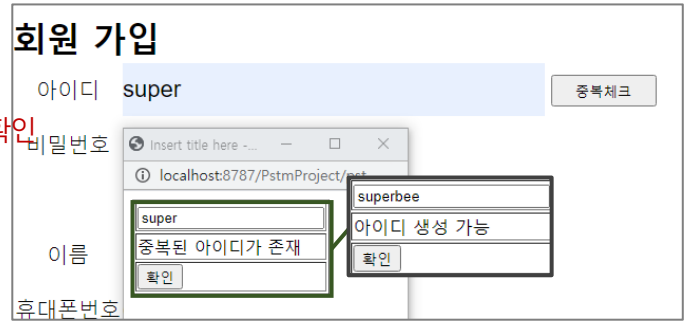
- 회원 가입 절차 및 화면 흐름은 다음과 같다

회원가입 창으로 이동

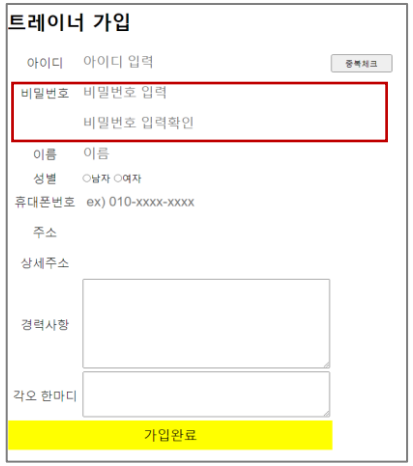


일반 회원가입으로 이동

회원가입 시 아이디 중복 확인



트레이너 회원가입으로 이동



비밀번호 입력 시 비밀번호 일치 여부 확인

회원가입-기술 소개(암호화) 및 코드

PasswordUtil.java

```java
public class PasswordUtil {

    private final static int KEY_LENGTH = 64;
    private final static int ITER_COUNT = 1293;

    public static String[] encrypt(String password) {
        String[] result = new String[2];

        try {
            SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

            char[] chars = password.toCharArray();
            String salt = getRandomKey();

            PBEKeySpec spec = new PBEKeySpec(chars, salt.getBytes(), ITER_COUNT, KEY_LENGTH * 8);
            byte[] hash = skf.generateSecret(spec).getEncoded();

            result[0] = byteToBase64(hash);
            result[1] = salt;

        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }

        return result;
    }
```

암호 키를 생성하는 팩토리 객체 생성

64byte 솔트 생성

암호 기반 키 파생 기능(PBKDF2) 구현

키 생성

- 회원가입 시 입력 받은 패스워드를 암호화 하여 저장했다
- 암호화에는 해시 알고리즘이 사용됐다

SignUpServlet.java

```java
String[] encryptedPassword = PasswordUtil.encrypt(password);

UserDto dto = new UserDto(0, id, encryptedPassword[0], encryptedPassword[1],
        usertype, gender, 0, null, career, mycomment, null, signout);
```

솔트와 암호화된 비밀번호 저장

회원가입-기술 소개(암호화) 및 코드

PasswordUtil.java

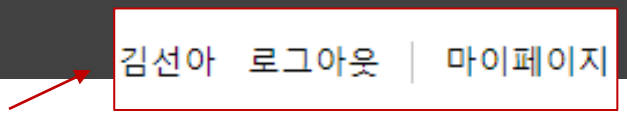
```
public static boolean checkPassword(String password, String salt, String resultPassword) {

    boolean result = false;

    try {

        SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        char[] chars = password.toCharArray();

        PBEKeySpec spec = new PBEKeySpec(chars, salt.getBytes(), ITER_COUNT, KEY_LENGTH * 8);
        byte[] hash = skf.generateSecret(spec).getEncoded();

        String encResult = byteToBase64(hash);

        if(encResult.equals(resultPassword)) {
            result = true;
        }

    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (InvalidKeySpecException e) {
        e.printStackTrace();
    }

    return result;
}
```

입력 받은 패스워드에 같은 과정 반복

입력 받은 패스워드와 저장된 패스워드가 같을 시 로그인 가능

- 새로 입력 받은 패스워드에 같은 과정을 반복해 기존에 저장된 패스워드와 같을 시 true를 반환하여 로그인이 가능하도록 함

로그인 기능-화면 흐름



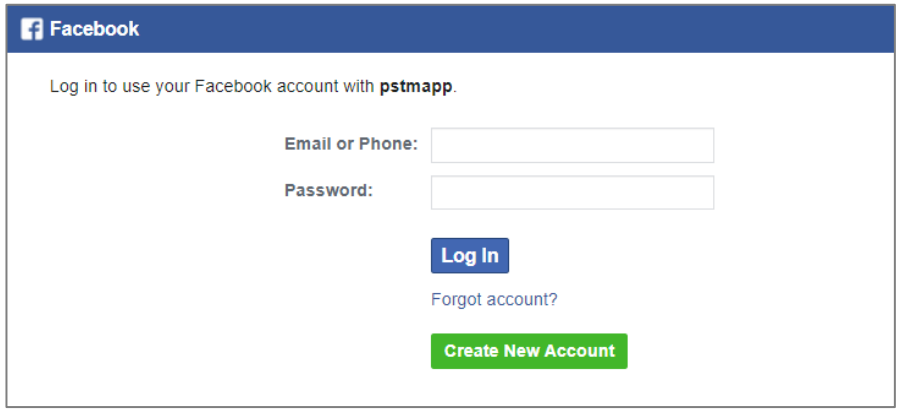
로그인 이후 pstm_header 변경

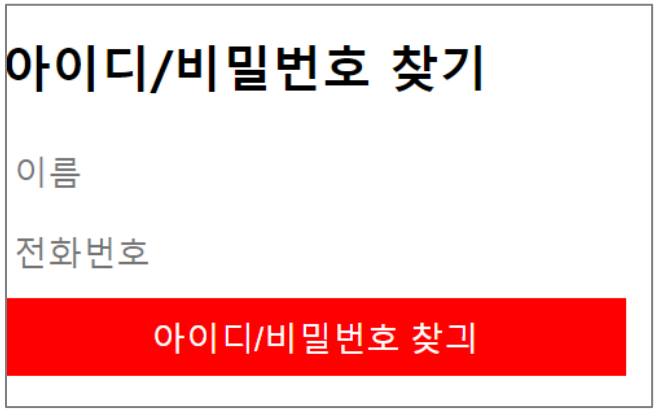
일반 로그인

네이버 로그인

페이스북 로그인

아이디/비밀번호 찾기 기능





-로그인 절차 및 화면 흐름은 다음과 같다

자체 로그인 서비스

- 자체 로그인 서비스와 SNS 로그인 API(네이버, facebook)를 이용했다

로그인 - SNS 로그인 절차 코드

① Client가 A/S에게 redirect URL을 포함하여 인증 요청
(SNS 사이트에 들어가 프로젝트를 생성하고 redirect URL을 입력)

② User가 " SNS로 로그인하기 " 버튼을 클릭하면 A/S는 해당 유저에게 로그인 창 제공

③ 로그인 후 A/S는 인증된 코드를 클라이언트에게 제공

*A/S : Authorization(권한 부여) Server

LoginServlet.java

```
} else if (command.equals("naverLogin")) {
    String clientId = "MtBEFWjjQiBrT7PZ7GhD";// 애플리케이션 클라이언트 아이디값";
    String clientSecret = "Y2ouJi2JiL";// 애플리케이션 클라이언트 시크릿값";
    String code = request.getParameter("code");
    String state = request.getParameter("state");
    String redirectURI =
        URLEncoder.encode("http://localhost:8787/PstmProject/Login.do?command=naverLogin",
        "UTF-8");
    String apiURL;
    apiURL = "https://nid.naver.com/oauth2.0/token?grant_type=authorization_code&";
    apiURL += "client_id=" + clientId;
    apiURL += "&client_secret=" + clientSecret;
    apiURL += "&redirect_uri=" + redirectURI;
    apiURL += "&code=" + code;
    apiURL += "&state=" + state;
    String access_token = "";
    System.out.println("apiURL=" + apiURL);
    try {
        URL url = new URL(apiURL);
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        int responseCode = con.getResponseCode();
        BufferedReader br;
        System.out.println("responseCode=" + responseCode);
        if (responseCode == 200) { // 정상 호출
            br = new BufferedReader(new InputStreamReader(con.getInputStream()));
        } else { // 에러 발생
            br = new BufferedReader(new InputStreamReader(con.getErrorStream()));
        }
        String inputLine;
        StringBuffer res = new StringBuffer();
        while ((inputLine = br.readLine()) != null) {
            res.append(inputLine);
        }
        br.close();
```

로그인-SNS 로그인 절차 코드

④ 클라이언트는 인증코드를 넘겨주며 A/S에 Access Token을 요청

⑤ A/S는 클라이언트에게 Access Token 발급한다.

*A/S : Authorization(권한 부여) Server

LoginServlet.java

```
if (responseCode == 200) {
    // PrintWriter out = response.getWriter();
    out.println(res.toString()); // 웹에 출력함
    JSONParser parsing = new JSONParser();
    Object obj = parsing.parse(res.toString());
    JSONObject jsonObj = (JSONObject) obj;

    access_token = (String) jsonObj.get("access_token");

    String token = access_token; // 네이버 로그인 접근 토큰;
    String header = "Bearer " + token; // Bearer 다음에 공백 추가

    // 네이버 회원 프로필 조회 API
    String apiurl = "https://openapi.naver.com/v1/nid/me";

    Map<String, String> requestHeaders = new HashMap<>();

    requestHeaders.put("Authorization", header);
    String responseBody = get(apiurl, requestHeaders);
    System.out.println(responseBody);

    Object answer = parsing.parse(responseBody);
```

로그인-SNS 로그인 절차 코드(Access Token Parsing)

LoginServlet.java

```
Object answer = parsing.parse(responseBody);

JSONObject JsonObj = (JSONObject) answer;
JSONObject resObj = (JSONObject) JsonObj.get("response");

String id = (String) resObj.get("id");

boolean login = false;

UserDto dto = dao.login(id);
```

id를 String 값으로 추출

- Access Token을 발급/갱신/삭제 요청 시 출력 포맷은 JSON 형식으로 출력 된다.

로그인- Session 처리 코드

LoginServlet.java

```
if (command.equals("login")) {

    String id = request.getParameter("id");
    String password = request.getParameter("password");

    boolean login = false;

    UserDto dto = dao.login(id);

    if (dto != null) {
        if(dto.getUsertype().equals("S") || dto.getUsertype().equals("T") && dto.getPassword().equals(password)) {
            HttpSession session = request.getSession();
            session.setAttribute("login", dto);

            session.setMaxInactiveInterval(-1);

            login = true;

            response.sendRedirect("pstm_mainpage.jsp");
        }

    }
```

세션에 login이라는 dto 저장

세션의 유효시간 무한으로 설정

pstm_header.jsp

```
<%
    UserDto userdto = (UserDto) session.getAttribute("login");
%>
```

웹 페이지 header 부분에 session 정보 불러 옴

- LoginServlet에서 세션 처리를 진행했으며, pstm_header.jsp에서 세션 값을 불러와 모든 페이지에서 세션 값을 이용할 수 있게 하였다

로그인-캐싱 방지 코드

pstm_header.jsp

```jsp
<%
① response.setHeader("Pragma", "no-cache"); //HTTP 1.0
② response.setHeader("Cache-Control", "no-cache"); //HTTP 1.1
③ response.setHeader("Cache-Control", "no-store"); //HTTP 1.1
④ response.setDateHeader("Expires", 0L); // Do not cache in proxy server
%>

<%
    if(session.getAttribute("login") == null){
        String result = "<script> alert('로그인을 먼저 해주세요!'); location.href='pstm_login.jsp'; </script> ";
        response.getWriter().append(result);

    }
```

Http헤더로 브라우저 캐싱 방지

세션 만료 시 alert 창을 띄우도록 함

① ("Pragma", "no-cache"): HTTP/1.0에서 캐시 등을 제어하기 위해 사용
② ("Cache-Control"," no-cache") : 캐시 된 복사본을 사용자에게 보여주기 이전에, 재검증을 위한 요청을 원 서버로 보내도록 강제
③ ("Cache-Control","no-store") : 캐시는 클라이언트 요청 혹은 서버 응답에 관해서 어떤 것도 저장하지 못하게 함
④ ("Expires", 0L) : 프록시 서버의 캐시 저장 방지

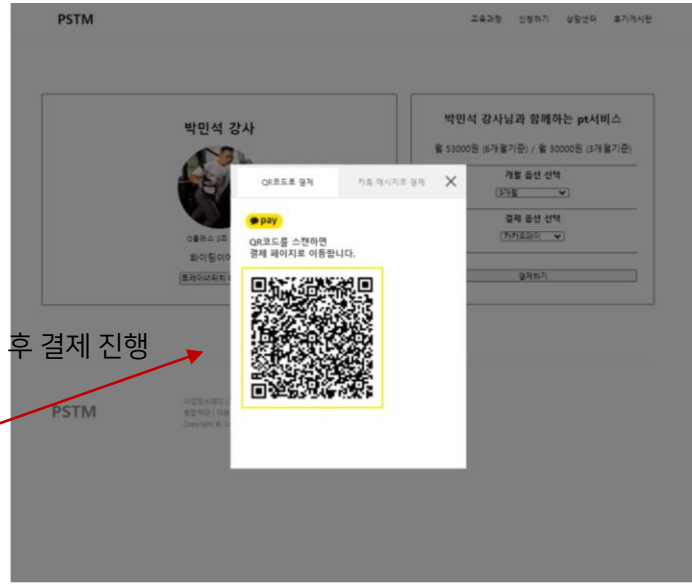
- 캐싱 방지를 하여 로그아웃 상태에서 뒤로 가기 버튼을 눌렀을 때 로그인을 해야 들어갈 수 있는 페이지에 접근하지 못하도록 했다

결제 기능-화면 흐름

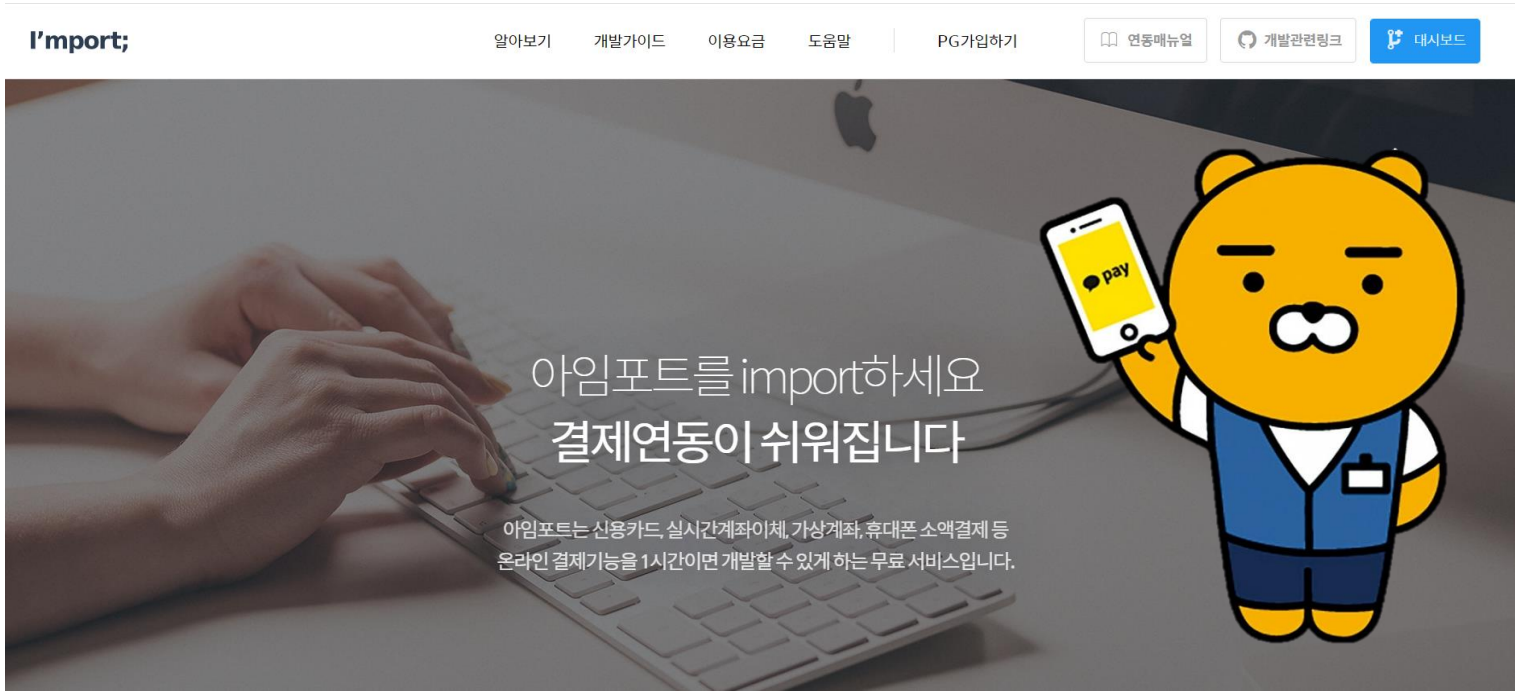
- 결제 기능 절차 및 화면 흐름은 다음과 같다

결제 페이지 이동

로그인 후 결제 진행



결제-기술 및 API 소개(아임포트/카카오 페이, Iamporter)



README.md

Iamporter

A REST API client for I'mport

circleci failing | coverage 93% | npm package 1.4.2 | downloads 43k | npm | licence MIT | Open Source

Iamporter는 아임포트에서 제공하는 REST API를 쉽게 활용하기 위하여 작성된 Node.js 클라이언트 입니다.

- 아임포트는 국내 PG사와의간편한 연동을 제공하는 서비스입니다.
- 이용 중 발생한 문제에 대하여 책임지지 않습니다.
- 최초 작성은 자동차 렌트 플랫폼 CARPLAT에서 사용하기 위하여 작성되었습니다.

Features

- Written in ES6 Syntax
- Promise Support
- Exception Handling with a custom error class `IamporterError`

- 결제를 진행하기 위해 결제 연동 API인 아임 포트를 이용했으며, 카카오페이로 결제를 진행할 수 있도록 설정했다
- REST API 이용 시 Node.js 클라이언트인 iamporter을 이용했다
- 결제 성공 시, 결제가 위조 여부를 판별하기 위한 절차를 진행해야 하며, 이 때 iamporter을 이용했다

Import : https://admin.iamport.kr/
Importer 오픈소스 : https://github.com/posquit0/node-iamporter

결제-아임포트/iamporter 코드

PSTM_Payment.jsp

```
IMP.request_pay({
    pg : 'kakao',
    pay_method : 'card',
    merchant_uid : 'merchant_' + new Date().getTime(),
    name : "주문명 :" + jsonTrainerDto["name"]+ "과 함께하는 강좌",
    amount : userpay,
    buyer_name : jsonNormalUserDto["name"],
    buyer_tel : jsonNormalUserDto["phone"],
    buyer_addr : jsonNormalUserDto["addr"]

}, function(rsp) {
    if (rsp.success) {

        alert('imp_uid' + rsp.imp_uid);
        $.ajax({
            url : "http://localhost:9999/aaa",
            method : "POST",
            dataType : "text",
            data : {
                'imp_uid' : rsp.imp_uid,
                'amount' : userpay
            }
        }).done(function(data){

            if(data == 'W'){
                alert('오류입니다. 관리자에게 문의해주세요');
            }else{
                alert('결제가 완료되었습니다');
            }
            isright = data;
        });
```

결제 요청

결제 성공 시

결제 정보 저장

Ajax로 서버에 데이터 전송

Callback으로 위조 여부 반환

paymentcheck.js

```
function findbyimpuid(Imp_uid){
    iamporter.findByImpUid(Imp_uid).
    then(function(response){
        confirmObject = eval(response);
        console.log(confirmObject);
        console.log(typeof(confirmObject));
        console.log(confirmObject.data.amount);
        serveramount = confirmObject.data.amount;

        isRight = isTrue(serveramount, amount);
    });
}

function isTrue(serveramount, realamount){

console.log('serveramount' + serveramount);
console.log('amount'  + realamount);
    if(realamount != serveramount){
        isRight = 'W';
    }
    console.log(isRight);
}
```

결제 정보 요청

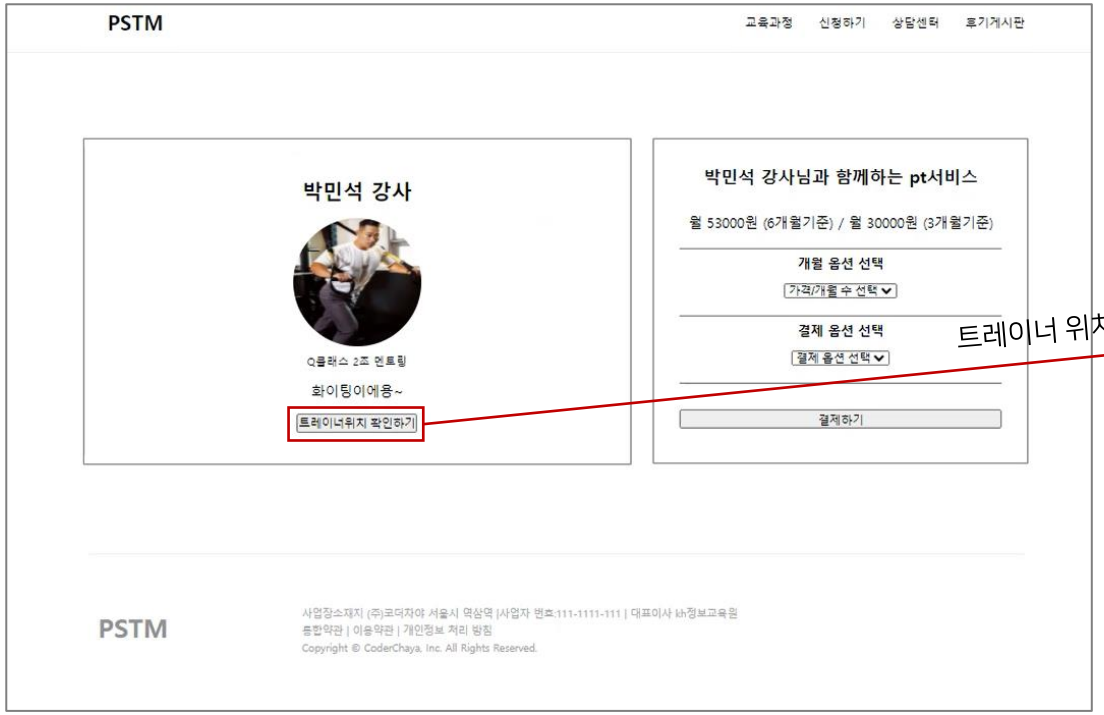
결제 정보 비교

아임포트 Server

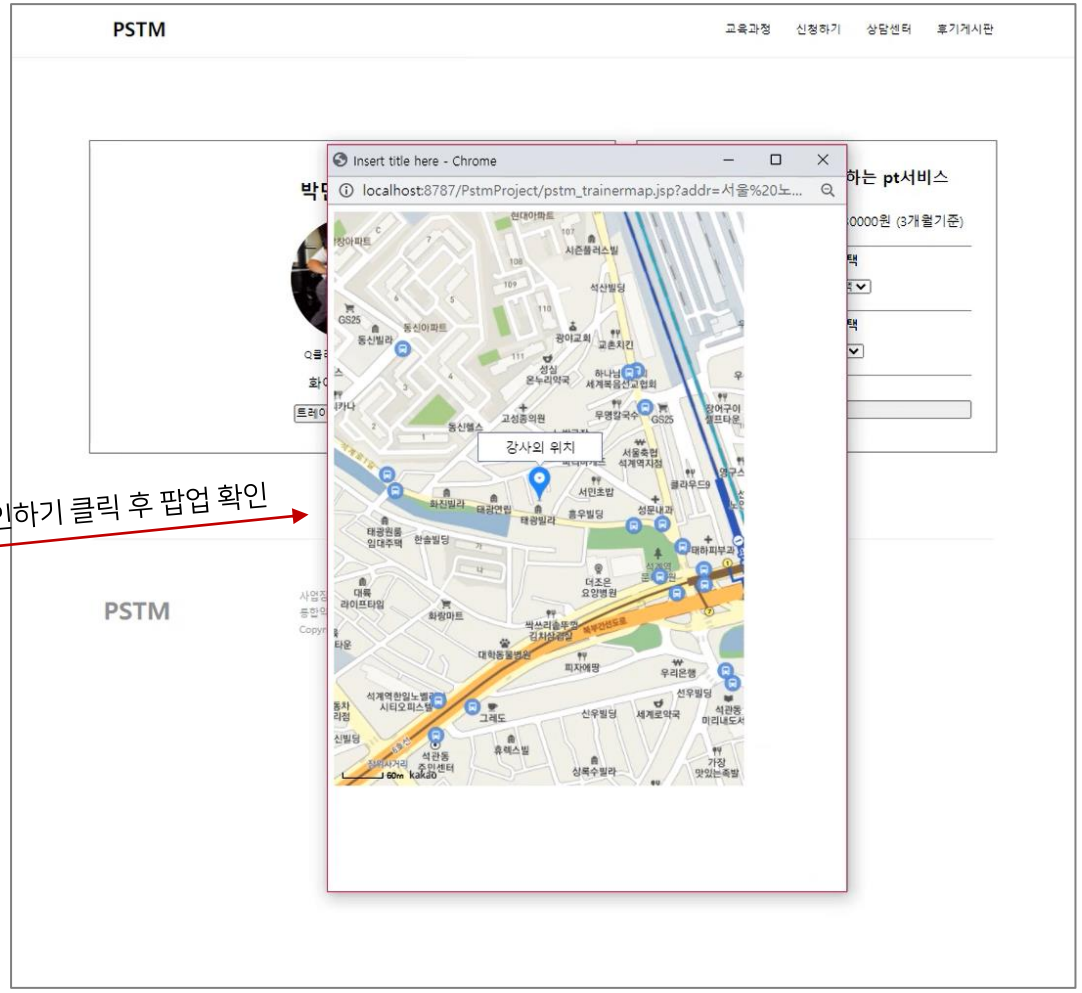
- Jquery Library “아임포트”를 이용한 일반 결제(카카오 페이) 구현
- Ajax로 결제 금액과 결제 번호(imp_uid)를 서버에 전송했다
- 서버에서는 API를 이용해 아임포트 서버에 저장된 결제 정보를 불러와 정보를 비교했다
- 이후 콜백으로 위변조 여부를 반환받았다

지도 기능-화면 흐름

- 지도 기능 이용 절차 및 화면 흐름은 다음과 같다



트레이너 위치 확인하기 클릭 후 팝업 확인



카카오 지도 API

- 카카오 지도 API를 이용해 유저 회원가입 시 주소 입력을 진행했다
- 트레이너의 주소를 받아와 위치 정보를 카카오 지도 위에 표시했다

```
<body>

    <div id="map" style="width: 500px; height: 700px;"></div>
    <script type="text/javascript"
        src="//dapi.kakao.com/v2/maps/sdk.js?appkey=18de9bf1d4dd5e0fc3db2f13a73abc7d&
        libraries=services,clusterer,drawing"></script>
    <script type="text/javascript">


    var Container = document.getElementById('map'); //지도를 담을 영역의 DOM 레퍼런스

    var options = { //지도를 생성할 때 필요한 기본 옵션

            center: new kakao.maps.LatLng(37.345479, 126.736520),    //지도의 중심좌표
            level: 3    // 지도의 레벨(확대, 축소 정도)
    };

    // 지도를 생성합니다
    var map = new kakao.maps.Map(Container, options);

    // 주소-좌표 변환 객체를 생성합니다
    var geocoder = new kakao.maps.services.Geocoder();

    // 주소로 좌표를 검색합니다
    geocoder.addressSearch(<%=wholeaddr%>, function(result, status) {

          // 정상적으로 검색이 완료됐으면
         if (status === kakao.maps.services.Status.OK) {

            var coords = new kakao.maps.LatLng(result[0].y, result[0].x);

            // 결과값으로 받은 위치를 마커로 표시합니다
            var marker = new kakao.maps.Marker({
                map: map,
                position: coords
            });
            // 인포윈도우로 장소에 대한 설명을 표시합니다
            var infowindow = new kakao.maps.InfoWindow({
                content: '<div style="width:150px;text-align:center;padding:6px 0;">강사의 위치</div>'
            });
            infowindow.open(map, marker);

            // 지도의 중심을 결과값으로 받은 위치로 이동시킵니다
            map.setCenter(coords);
        }
    });
```

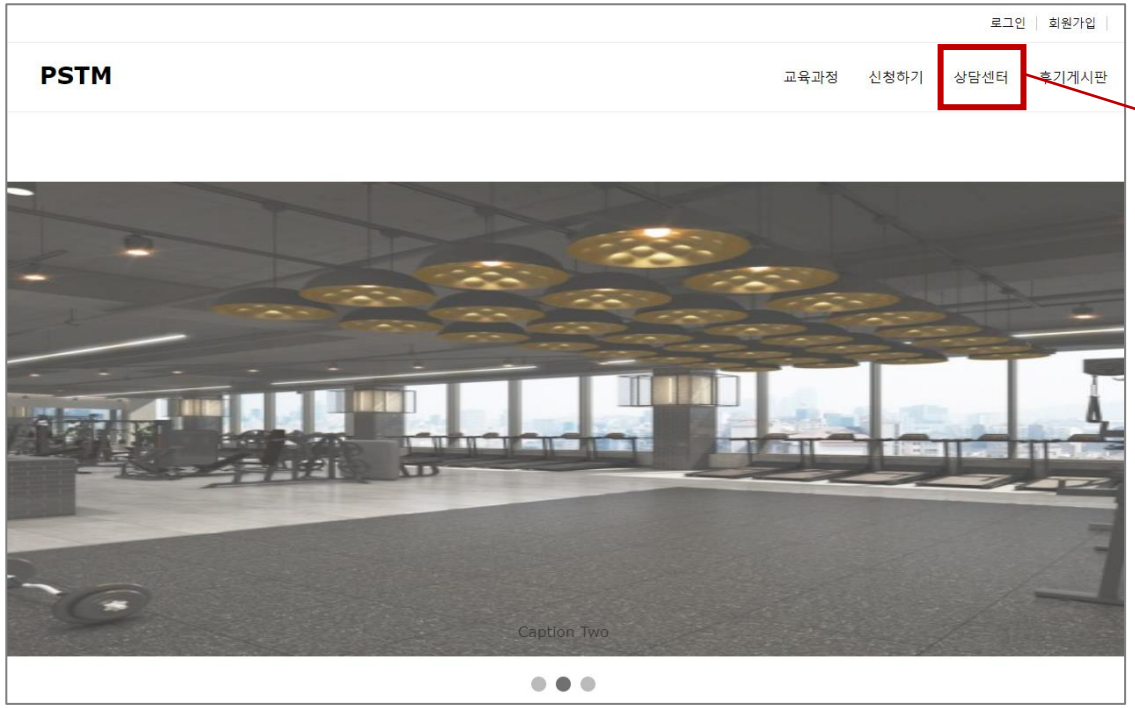
주소로 좌표 검색

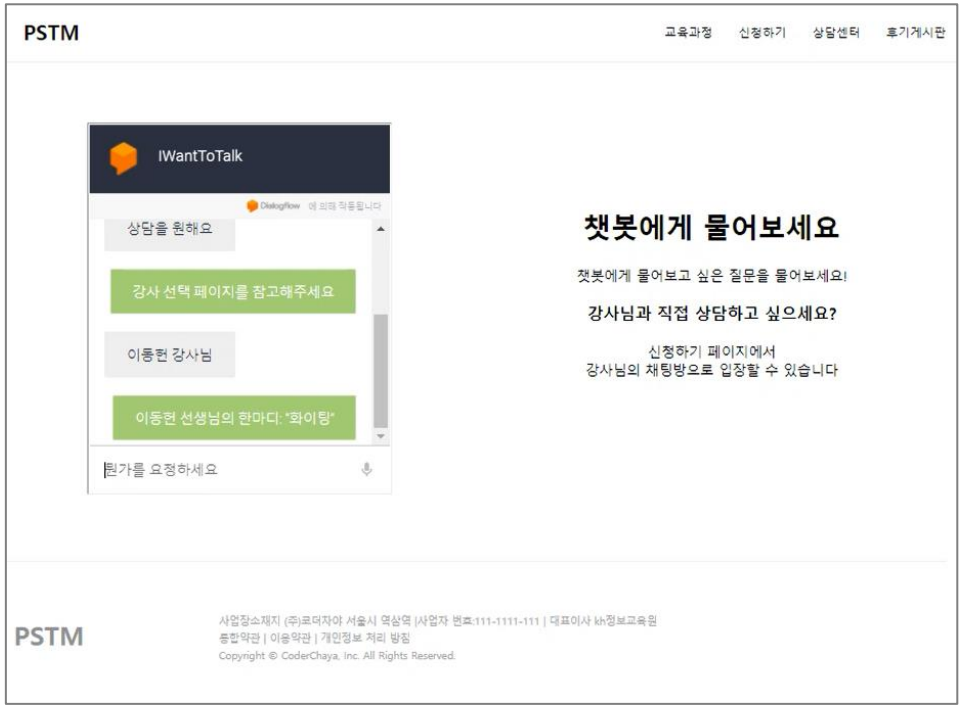
지도 마커 표시

채팅 기능(챗봇)-화면 흐름

- 챗봇 기능 이용 절차 및 화면 흐름은 다음과 같다

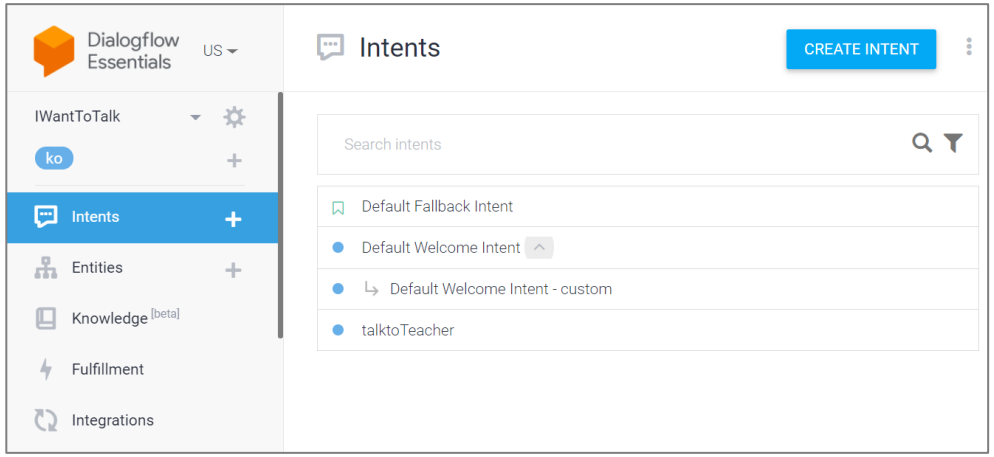
비회원/일반회원 로그인 상태로 상담센터 버튼을 클릭하면 챗봇 페이지로 이동된다



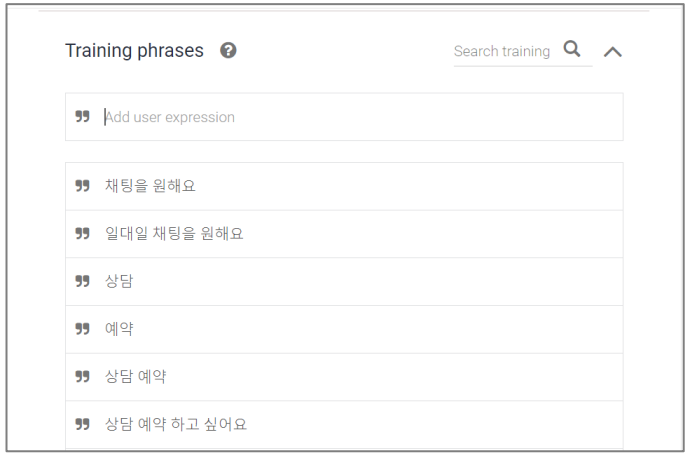


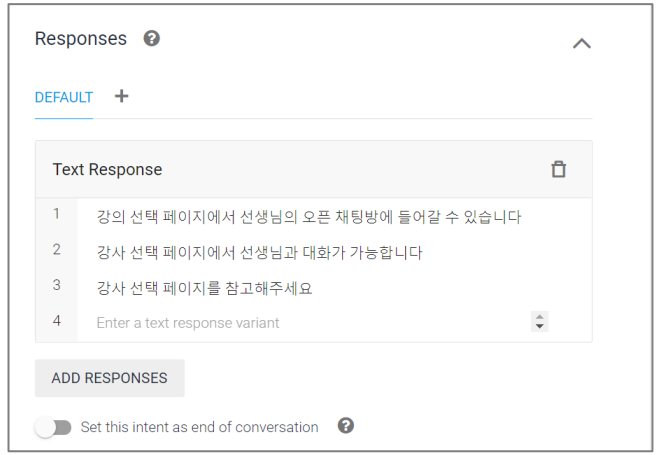
채팅- API 소개(챗봇/google DialogFlow)

DialogFlow는 자연어 처리를 통해 Intent(의도) ,Entity(속성), Context(문맥)을 파악하는 대화형 인터페이스이다



- 대화의 수행 단위인 Intent 생성





- 사용자가 할 것이라 예상되는 질문을 작성
- 질문에 알맞은 대답을 학습시킴
- Entity는 json 파일로 업로드 가능

채팅 기능(멀티룸 채팅)-화면 흐름

- 강사 오픈채팅 이용 절차 및 화면 흐름은 다음과 같다

강사 회원으로 채팅방 입장

비회원/일반 회원으로 채팅방 입장

채팅-기술 소개(오픈 채팅/socket.io 멀티룸)

MultiThread / TCP 통신 → JSP도 서버 스레드 위에서 동작하기 때문에 java 스레드로 화면 관리 어려움 ✖

Html5 WebSocket → 구조적으로 멀티룸 채팅 구현이 어려움 ✖

↓

위 두 기술 모두 우리 프로젝트에 이용하기에 적합하지 않았기 때문에
Node.js socket.io 멀티 룸 방식을 이용하여 채팅방을 만들었다

채팅-멀티룸 채팅(socket.io) 코드

pstm_chatdirect.js

```
io.on('connection', (socket) => {
```

```
socket.on('adduser', (username) =>{
    socket.room = specificroom;
    // store the username in the socket session for this client
    if(istrainer == 'true'){
        console.log("specificroom"  +specificroom);
    //specificroom  = trainername + trainernum이므로 이렇게 설정했음
    socket.username = specificroom  + "트레이너";
    console.log( "socket.username" + socket.username);
    usernames[username] = specificroom;
    }else{
    console.log("elseistrainer"  +istrainer);
    socket.username = username + num;
    num++;
    usernames[username] = username + num;
    }
```

adduser : 입장 시 이름을 "익명"으로 전달

```
    // send client to room 1
    socket.join(specificroom);
    socket.emit('updatechat', 'SERVER', 'you have connected to' + specificroom);
    socket.emit('updaterooms', rooms,specificroom);
});
```

updaterooms : [트레이너이름+트레이너 번호]인 채팅 룸을 생성한다

```
  // when the client emits 'sendchat', this listens and executes
  socket.on('sendchat', function (data) {
      // we tell the client to execute 'updatechat' with 2 parameters
      io.sockets.in(socket.room).emit('updatechat', socket.username, data);
  });
```

sendchat : 사용자가 입력한 텍스트 전달

updatechat : 채팅 화면을 업데이트

```
// when the user disconnects.. perform this
socket.on('disconnect', function(){
    delete usernames[socket.username];
    // update list of users in chat, client-side
    io.sockets.emit('updateusers', usernames);
    socket.emit('updatechat', 'SERVER', socket.username + 'leave the chat room')
    socket.leave(socket.room);
});
```

socket.leave : 사용자 퇴장시 소켓룸에서 삭제

pstm_chatuser.html(화면)

```
var socket = io.connect('http://localhost:9999');
//접속 완료
socket.on('connect', function(istrainer, trainername, trainernum) {
    socket.emit('adduser', "익명");

});
```

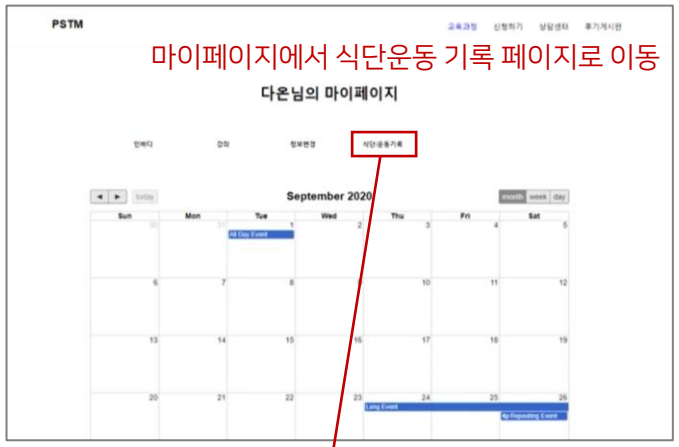
```
// listener, whenever the server emits 'updaterooms', this updates the room the client is in
socket.on('updaterooms', function(rooms, current_room) {
    $('#rooms').empty();
    $.each(rooms, function(key, value) {

        if (value == current_room) {
            $('#roomtitle').text( "트레이너\n" + current_room + "의 \n채팅방");
        }

    });
});
```

```
// on load of page
$(function() {
    // when the client clicks SEND
    $('#datasend').click(function() {
        var message = $('#data').val();
        $('#data').val('');
        // tell server to execute 'sendchat' and send along one parameter
        socket.emit('sendchat', message);
    });
```

```
// listener, whenever the server emits 'updatechat', this updates the chat body
socket.on('updatechat', function(username, data) {
    $('#conversation').append('<b>' + username + ':</b> ' + data + '<br>');
});
```

- 트레이너 이름+트레이너 번호인 멀티채팅룸을 만들어, 채팅을 진행했다

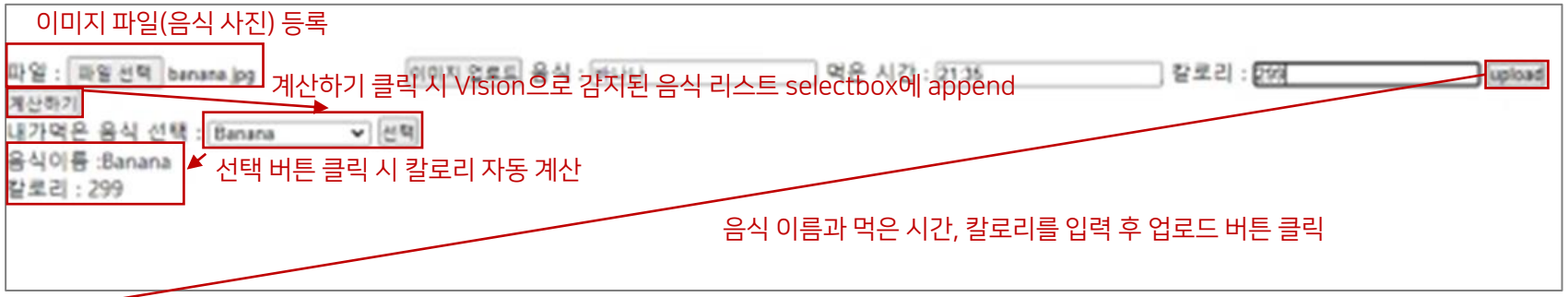
마이페이지/식단 관리 기능 -화면 흐름



마이페이지에서 식단운동 기록 페이지로 이동

새 글 등록하기

등록된 글이 스크롤 페이징 방식으로 보여짐



계산하기 클릭 시 Vision으로 감지된 음식 리스트 selectbox에 append

선택 버튼 클릭 시 칼로리 자동 계산

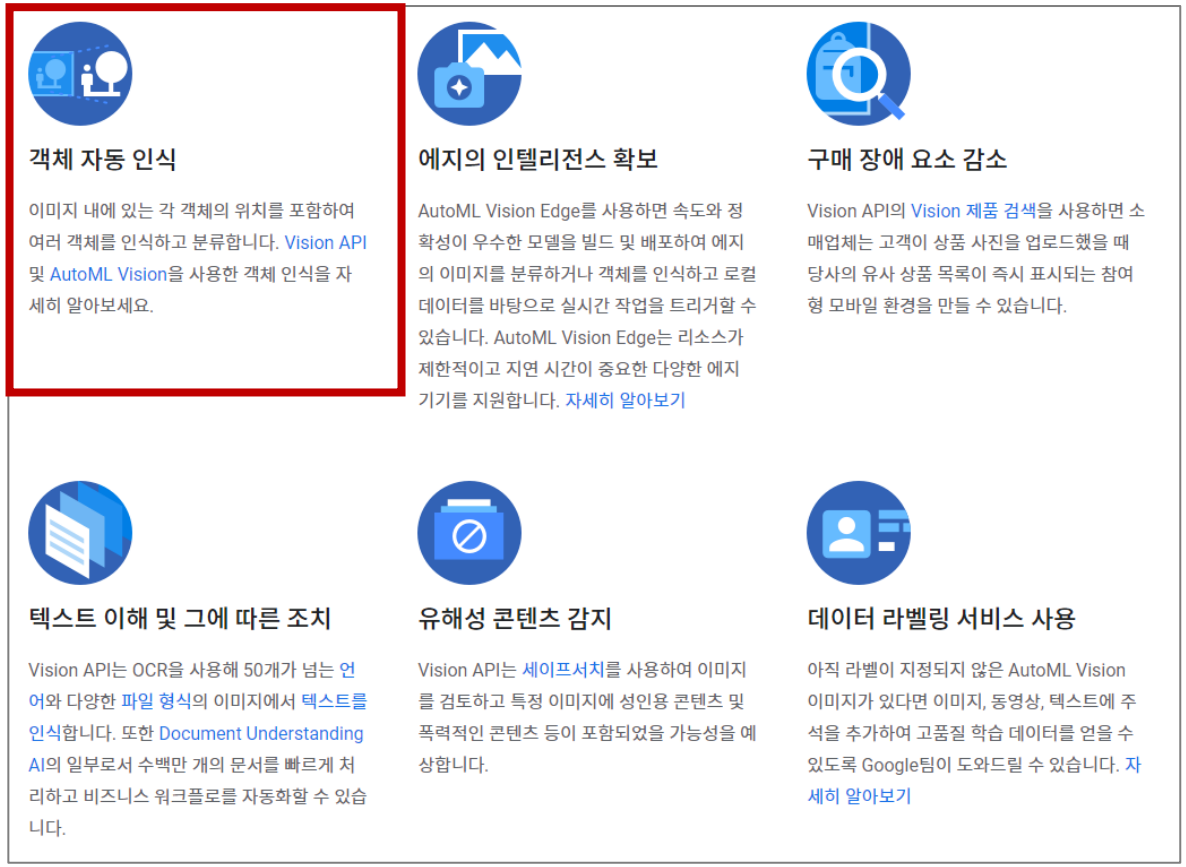
음식 이름과 먹은 시간, 칼로리를 입력 후 업로드 버튼 클릭

-마이페이지/식단 관리 기능의 이용 절차 및 화면 흐름은 위와 같다

Vision AI

AutoML Vision을 사용하여 클라우드나 에지 이미지에서 유용한 정보를 도출하거나 선행 학습된 Vision API 모델을 사용하여 이모티콘 인식, 텍스트 이해 등을 수행합니다.

Console로 이동



객체 자동 인식

이미지 내에 있는 각 객체의 위치를 포함하여 여러 객체를 인식하고 분류합니다. Vision API 및 AutoML Vision을 사용한 객체 인식을 자세히 알아보세요.

에지의 인텔리전스 확보

AutoML Vision Edge를 사용하면 속도와 정확성이 우수한 모델을 빌드 및 배포하여 에지의 이미지를 분류하거나 객체를 인식하고 로컬 데이터를 바탕으로 실시간 작업을 트리거할 수 있습니다. AutoML Vision Edge는 리소스가 제한적이고 지연 시간이 중요한 다양한 에지 기기를 지원합니다. 자세히 알아보기

구매 장애 요소 감소

Vision API의 Vision 제품 검색을 사용하면 소매업체는 고객이 상품 사진을 업로드했을 때 당사의 유사 상품 목록이 즉시 표시되는 참여형 모바일 환경을 만들 수 있습니다.

텍스트 이해 및 그에 따른 조치

Vision API는 OCR을 사용해 50개가 넘는 언어와 다양한 파일 형식의 이미지에서 텍스트를 인식합니다. 또한 Document Understanding AI의 일부로서 수백만 개의 문서를 빠르게 처리하고 비즈니스 워크플로를 자동화할 수 있습니다.

유해성 콘텐츠 감지

Vision API는 세이프서치를 사용하여 이미지를 검토하고 특정 이미지에 성인용 콘텐츠 및 폭력적인 콘텐츠 등이 포함되었을 가능성을 예상합니다.

데이터 라벨링 서비스 사용

아직 라벨이 지정되지 않은 AutoML Vision 이미지가 있다면 이미지, 동영상, 텍스트에 주석을 추가하여 고품질 학습 데이터를 얻을 수 있도록 Google팀이 도와드릴 수 있습니다. 자세히 알아보기

- 음식 사진 업로드 시 Vision AI의 객체 자동 인식 기능을 이용해 사진 속 음식의 정보(이름)을 추출했다

마이페이지/식단 관리 기능- API 소개(국가 표준 식품 성분 DB)

농촌진흥청 국립농업과학원 농식품올바로

요청 변수(Request Parameters)

요청변수	값	설명
apiKey	String(필수)	OpenAPI 이용신청을 통해 받은 KEY
serviceType	String:기본값 AA001 [AA001:XML, AA002:JSON]	서비스 타입 코드
nowPage	Integer:기본값 1	현재 페이지
pageSize	Integer:기본값 10 [10, 20, 30, 40, 50, 100, 200, 500]	페이지당 출력 건수
fdGrupp	String [A~T] ※하단 식품군 코드표 참조	식품군
fdNm	String	국문 식품명 검색어
fdEngNm	String	영문 식품명 검색어
examinYear	Integer [2자리 형태]	조사연도(일치 검색)
originNm	String	출처

응답 결과(Response Element)

응답변수	값	설명
fdCode	String	식품코드
fdGrupp	String	식품군
fdNm	String	국문 식품명
fdEngNm	String	영문 식품명
originNm	String	출처
examinYear	String	조사연도
irdntSeNm	String	성분구분명
irdntNm	String	국문 성분명
irdntEngNm	String	영문 성분명
contInfo	String	함량
irdntUnitNm	String	함량 단위

- 국가 표준 식품 성분 DB를 이용해 사진에서 추출된 음식의 칼로리를 계산했다
- 요청 변수로는 국문 식품명 검색어가 이용됐으며, 응답 결과는 칼로리 및 영양성분을 반환하도록 했다

마이페이지/식단 관리-파일 업로드 및 DB 저장 코드

pstm_dailyinsert.jsp

<form>요소가 파일이나 이미지를 서버로 전송할 때 사용됨

```
<form method="post" enctype="multipart/form-data" id="createupload" action="daily.do">
    <input type="hidden" name="command" value="insertres">
    파일 : <input type="file" name="uploadimg" id="imput_img" class="excelFile">
     <button onclick="upload(event)">이미지 업로드</button>
    음식 : <input type="text" name="result">
    먹은 시간 : <input type="text" name="timeeat">

    칼로리 : <input type="text" name="kcal">
```

- 음식 사진을 업로드 하기 위해 파일 업로드를 진행했다
- 이미지 업로드를 진행하기 위해 multipart/form-data 설정을 했다

마이페이지/식단 관리-객체 감지(Vision API) 코드

DailyController.java

```
String command = request.getParameter("command");
System.out.println("command : " + command);
if (command == null) {
    String path = request.getSession().getServletContext().getRealPath("imgfolder");
    int size = 1024 * 1024 * 5;
    MultipartRequest multi = new MultipartRequest(request, path, size, "UTF-8", new DefaultFileRenamePolicy());
    command = multi.getParameter("command");
    System.out.println("이미지경로 : "+path);

try {

    String uploadimg = multi.getParameter("uploadimg");
    String originimg = multi.getParameter("originimg");
    String timeeat = multi.getParameter("timeeat");
    int kcal = Integer.parseInt(multi.getParameter("kcal"));
    String result = multi.getParameter("result");

    Enumeration files = multi.getFileNames();
    String str = (String) files.nextElement();

    uploadimg = multi.getFilesystemName(str);
    originimg = multi.getOriginalFileName(str);
```

Path : Path를 이용해 파일이 업로드 될 실제 경로를 잡아준다

MultipartRequest : 파일을 서블릿에서 받았을 때 파라미터 값을 MultipartRequest로 받는다

Enumeration : 업로드 된 파일의 이름을 반환 할 때 Enumeration 메서드를 사용한다

- 이미지를 업로드한 후 이미지 실제 저장 경로를 잡아주었다

마이페이지/식단 관리-객체 감지(Vision API) 코드

DailyController.java

```
protected List<String> Vision(String Path, String filename) throws ServletException, IOException {

① ImageAnnotatorClient vision = ImageAnnotatorClient.create();

    String fileName = Path + "\\" + filename;

② Path path = Paths.get(fileName);

    System.out.println(path.toAbsolutePath());

③ byte[] data = Files.readAllBytes(path);
④ ByteString imgBytes = ByteString.copyFrom(data);

    List<AnnotateImageRequest> requests = new ArrayList<AnnotateImageRequest>();

    Image img = Image.newBuilder().setContent(imgBytes).build();

    Feature feat = Feature.newBuilder().setType(Type.LABEL_DETECTION).build();
⑤ AnnotateImageRequest requestres = AnnotateImageRequest.newBuilder().addFeatures(feat).setImage(img).build();
    requests.add(requestres);

⑥ BatchAnnotateImagesResponse responseres = vision.batchAnnotateImages(requests);
    List<AnnotateImageResponse> responses = responseres.getResponsesList();
    List<String> result = new ArrayList<String>();
    for (EntityAnnotation annotation : res.getLabelAnnotationsList()) {

        annotation.getAllFields().forEach((k, v) -> {
            System.out.println(k.toString());
            if (k.toString().equals("google.cloud.vision.v1.EntityAnnotation.description")) {
                result.add(v.toString());
            }
        });

    }
    System.out.println(result);
    JsonArray array = new Gson().toJsonTree(result).getAsJsonArray();
}
return result;
}
```

① AnnotateImageResponse: 이미지 요청에 대한 응답
② ImageAnnotatorClient : 이미지에서 감지된 엔티티 반환
③ Path : 파일을 찾는데 사용할 수 있는 개체
④ Files.readAllBytes : 파일의 모든 바이트 읽어 옴
⑤ ByteString.copyFrom(data) : 바이트 시퀀스의 집합
⑥ AnnotateImageRequest: : 구글 비전에 사용할 이미지의 정보

- 저장된 이미지 경로를 받아와, 이미지 정보를 읽어왔다
- 이후 Vision API에서 이미지 정보를 사용하여 이미지 속 객체 정보(음식 정보/이름)을 추출했다

마이페이지/식단 관리-칼로리 계산 API 코드

pstm_dailyinsert.jsp

```
function kcalcount(){

    var fileValue = $(".excelFile").val().split("\\");
    var filename = fileValue[fileValue.length-1];

    $.ajax({
          method: "POST",
          url: "daily.do?command=vision&filename="+filename,
          contentType : "application/text; charset=UTF-8 ",
          dataType : "json",
          success : function(data) {

             var obj = JSON.stringify(data);
             var foodlist = JSON.parse(obj);
             var foodnames = foodlist.toString().split(",");

        for(var i = 0; i < foodnames.length; i++){
                    $("#foodselect").append("<option value="+foodnames[i]+">"+foodnames[i]+"</option>");
                }
          },

          error:function(request,status,error){
                alert("code:"+request.status+"\n"+"message:"+request.responseText+"\n"+"error:"+error);
               }
        })

}
```

업로드된 파일이름을 ajax로 비동기화해서 컨트롤러로 보내주고 vision 메소드를 실행시켜 결과값을 json array 방식으로 받아온다.

받아온 값을 String으로 변환시키고 split을 이용해 ','로 잘라서 foodselect에 append 시켜준다.

- 업로드 된 이미지 정보를 ajax로 보내어 Vision으로 추출한 객체 정보들을 반환 받았고, 이를 select 태그에 연결했다

마이페이지/식단 관리-칼로리 계산 API 코드

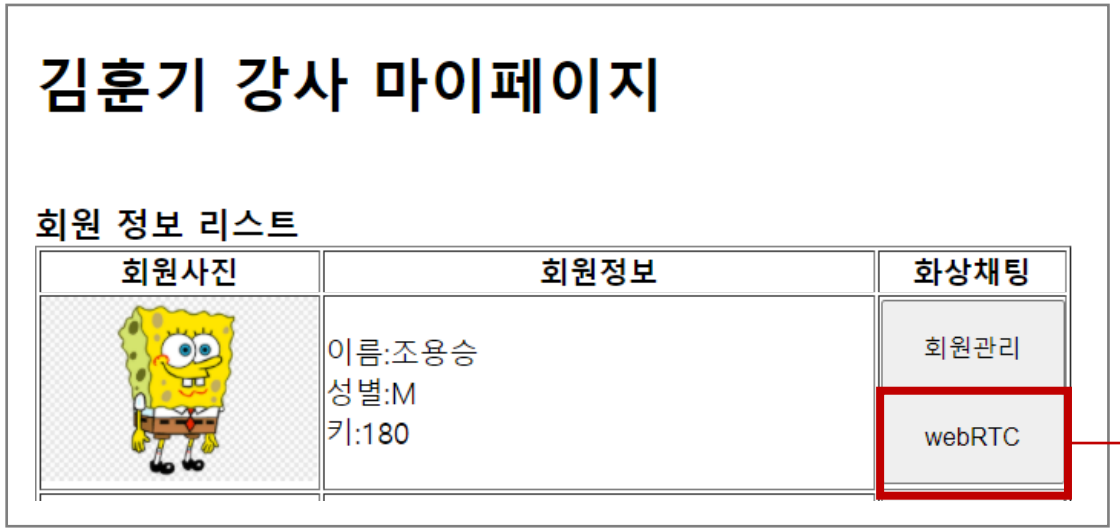
pstm_dailyinsert.jsp

```
function foodnamesend(){

    var foodname = $("#foodselect option:selected").val();

    $.ajax({
        method : "POST",
        url : "http://koreanfood.rda.go.kr/kfi/openapi/service",
        data : {
            apiKey : "20200917103617HS0PNAS5YSH17TMJEO",
            serviceType : "AA002",
            //fdGrupp :"I",
            fdEngNm : ""+foodname+""
        },
    }).done(function(msg) {

        var kcal = msg.service.list[0].irdnt[0].irdnttcket[0].contInfo;
        $("#kcalall").append("음식이름 :" +foodname+"<br> 칼로리 : "+kcal);

    })
}
```

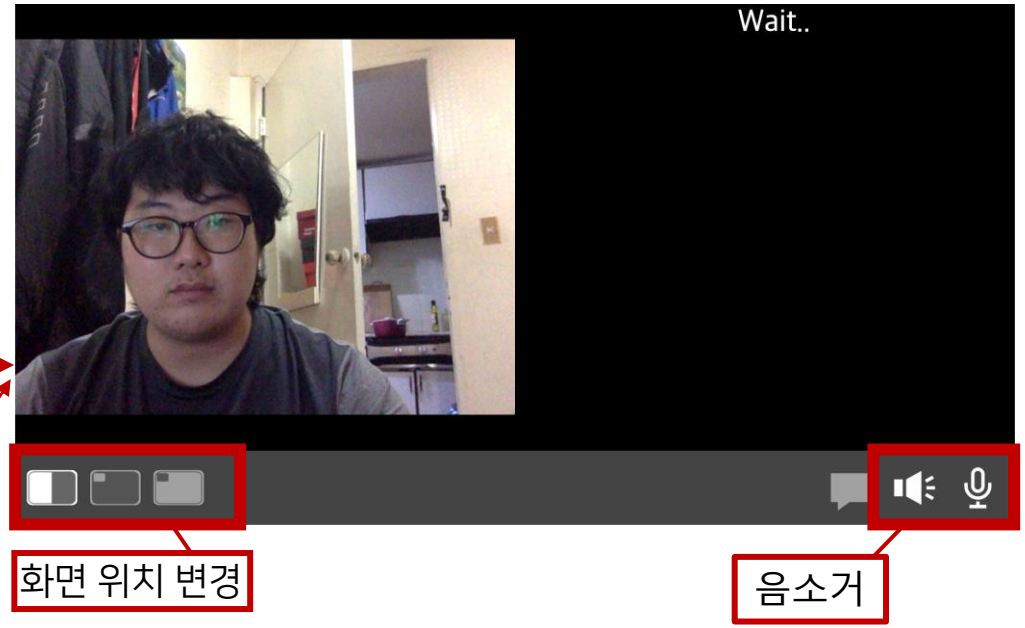
선택한 값을 ajax로
칼로리 API에 전송

받아온 값을 kcalall에 append 시켜준다.

- Select 박스에서 선택된 음식 이름을 국가 표준 식품 성분 API에 전송해, 칼로리 값을 반환받았다
- 이를 화면에 출력시켰다



강사 마이페이지/일반 회원 마이페이지에서
webRTC/강의 버튼을 누르면
1:1 강의 기능 페이지 로 이동한다



- 1:1 강의 기능의 이용 절차 및 화면 흐름은 위와 같다
- Node.js/React.js 기반으로 구현을 진행했다
- 클라이언트에서는 React-redux, socket.io-client 패키지를 이용했다
- 서버에서는 express, socket.io, https, fs 패키지를 이용했다

1:1 강의-소켓 통신 코드

node_main.js(서버)

```
socket.on('sendMessage', (res) => {
    console.log(res);
    for(var i = 0; i < users.length; i++) {
        if(users[i].id === res.id) {
            if(res.command === 'offer') {
                users[i].socket.emit('receiveOffer', {sdp:res.sdp});
                break;
            } else if(res.command === 'answer') {
                users[i].socket.emit('receiveAnswer', {sdp:res.sdp});
                break;
            } else if(res.command === 'candidate') {
                users[i].socket.emit('receiveCandidate', {candidate:res.candidate});
                console.log('receiveCandidate');
                break;
            } else if(res.command === 'speakerMute'){
                users[i].socket.emit('opponentSpeakerMuted', {mute:res.mute});
                break;
            } else if(res.command === 'micMute') {
                users[i].socket.emit('opponentMicMuted', {mute:res.mute});
                break;
            } else if(res.command === 'sendMessage') {
                users[i].socket.emit('recieveMessage', {message:res.message});
            }
        }
    }
});
```

클라이언트에서 데이터 수신

상대방에게 데이터 전송

MainPage.js(클라이언트)

클라이언트 데이터 수신

```
initSocket() {
    this.socket.on('start', () => {
        this.createOffer();
    });

    this.socket.on('receiveOffer', (res) => {
        console.log('receiveOffer');
        console.log(res.sdp);
        this.setRemoteDesciprtion(res.sdp, true);
    });

    this.socket.on('receiveAnswer', (res) => {
        console.log('receive answer');
        this.setRemoteDesciprtion(res.sdp, false);
    });

    this.socket.on('receiveCandidate', (res) => {
        console.log(res.candidate);
        this.createCandidate(res.candidate);
    });

    this.socket.on('opponentSpeakerMuted', (res) => {
        console.log(res.mute);
        this.props.setOpponectSpeakerMute(res.mute);
    });

    this.socket.on('opponentMicMuted', (res) => {
        console.log(res.mute);
        this.props.setOpponectMicMute(res.mute);
    });

    this.socket.on('opponentDisconnect', () => {
        this.pc.close();
        this.pc = new RTCPeerConnection(pc_config);
        this.remoteVideo.reset();
    });
}
```

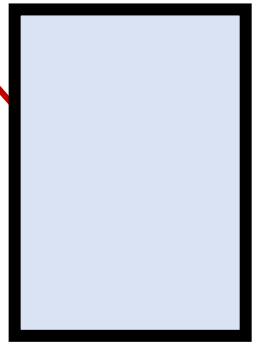
- socket.io와 socket.io-client 패키지를 사용하여 소켓 통신 구현
- 서버 데이터 수신 및 상대방에게 송신

1:1 강의-반응형 웹 코드

VideoPlayer.js

```
if(this.props.displayType === this.props.displayTypeEnum.half) {
    if(this.state.width > this.state.height) {
        videoStyle.width = '50%';
        videoStyle.height = '100%';


        if(this.props.openChat) {
            videoStyle.width = '35%';
        }

        if(this.props.remote) {
            videoStyle.left = '50%';

            if(this.props.openChat) {
                videoStyle.left = '35%';
            }
        }
    } else {
        videoStyle.width = '100%';
        videoStyle.height = '50%';

        if(this.props.openChat) {
            videoStyle.height = '35%';
        }

        if(this.props.remote) {
            videoStyle.top = '50%';

            if(this.props.openChat) {
                videoStyle.top = '35%';
            }
        }
    }
```

화면의 가로가 세로보다 클 때



화면의 세로가 가로보다 클 때

- 웹의 크기 등의 데이터를 받아와 보여주는 크기를 결정해서 보여 줌
- 웹의 가로 세로을 확인하여 보여주는 방식을 결정하기도 함.
- 비율에 맞춰 보여주는 방식 변환

MainPage.js

```
createOffer() {
    console.log('Offer');

    this.pc.createOffer({offerToReceiveVideo:1, offerToReceiveAudio:1}).then(sdp => {
        console.log(JSON.stringify(sdp));
        this.pc.setLocalDescription(sdp);
        this.socket.emit('sendMessage', {command:'offer', sdp:JSON.stringify(sdp), id:this.opid});
        console.log('send Offer');
    }, e => {})
}

createAnswer() {
    console.log('Answer');
    this.pc.createAnswer({offerToReceiveVideo: 1, offerToReceiveAudio:1}).then(sdp => {
        console.log(JSON.stringify(sdp));
        this.pc.setLocalDescription(sdp);
        console.log(this.opid);
        this.socket.emit('sendMessage', {command:'answer', sdp:JSON.stringify(sdp), id:this.opid});
        console.log('sendAnswer');
    }, e => {});
}
```

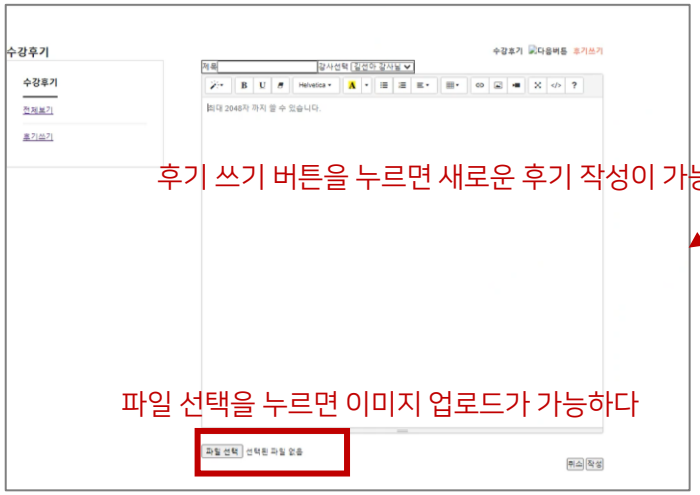
WebRTC를 초기화 할 때 필요한 데이터 생성

소켓을 이용하여 상대방에게 데이터를 전송

- WebRTC를 이용하여 비디오, 오디오를 공유
- 필요한 데이터들(비디오/오디오 데이터)은 소켓 통신으로 공유

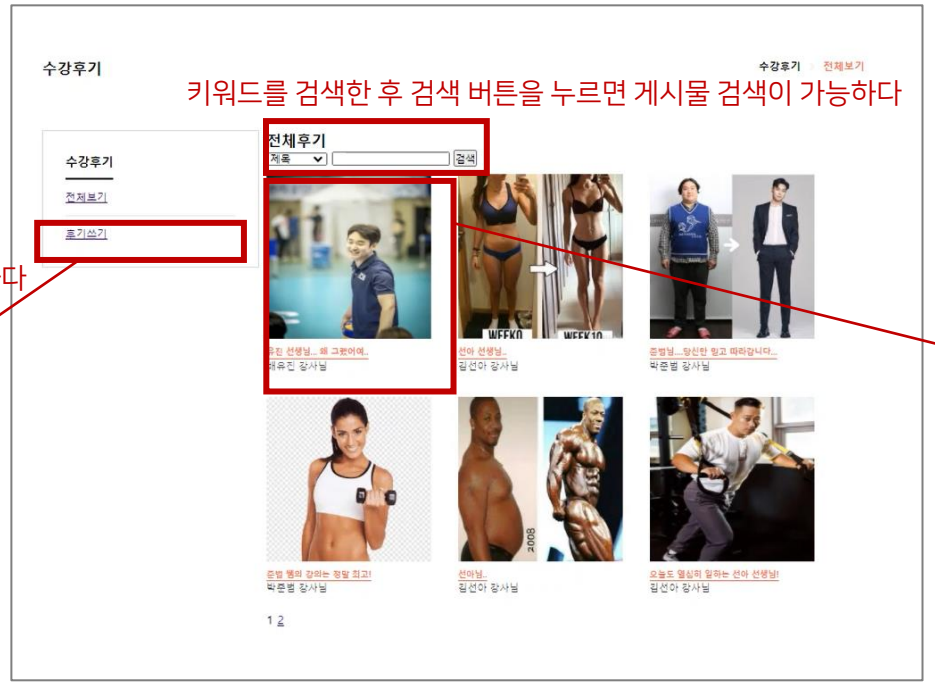
후기 게시판 -화면 흐름

-후기 게시판 이용 절차 및 화면 흐름은 위와 같다



후기 쓰기 버튼을 누르면 새로운 후기 작성이 가능하다

파일 선택을 누르면 이미지 업로드가 가능하다

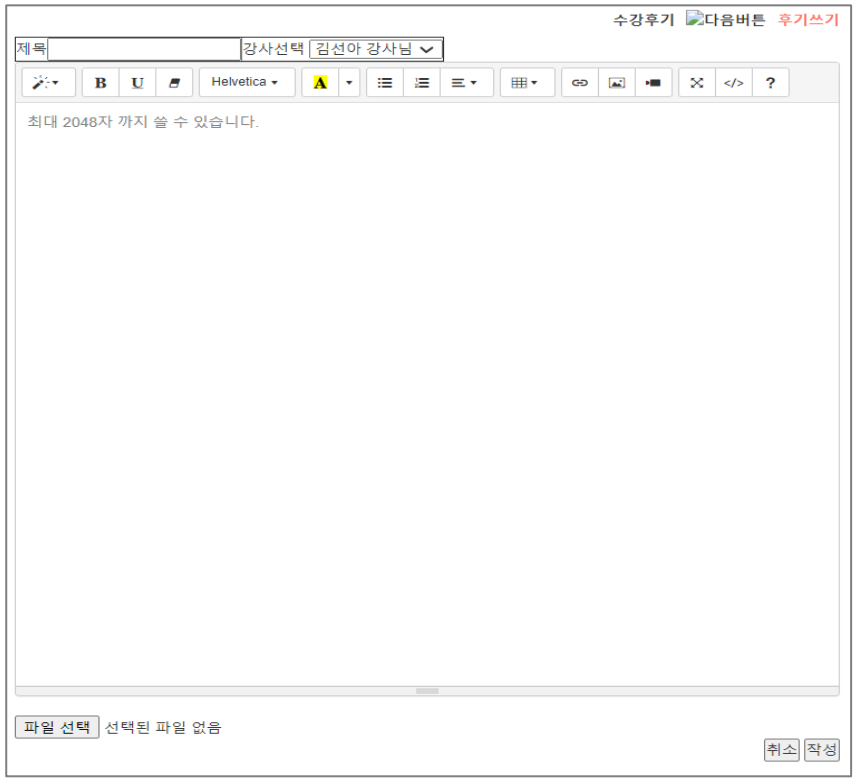


키워드를 검색한 후 검색 버튼을 누르면 게시물 검색이 가능하다

게시물을 클릭하면 상세보기가 가능하다

상세 페이지에서 게시물 수정/삭제가 가능하다

썸머노트 API



- 글을 작성할 때 폰트, 색깔, 밑줄 등을 사용해 꾸밀 수 있다.
- 썸머노트 API를 이용해 글쓰기 업로드를 진행했다

```
    $(document).ready(

            function() {
                $('#summernote').summernote(
                        {
                            lang : 'ko-KR',
                            height : 600,
                            minHeight : null,
                            maxHeight : null,
                            focus : true,
                            fontNames : [ '맑은고딕', 'Arial', 'Arial Black',
                                    'Comic Sans MS', 'Courier New', ],
                            placeholder : '최대 2048자 까지 쓸 수 있습니다.'
                        });
            })

    function updateSummernoteImageFile(file, editor) {
        data = new FormData();
        data.append("file", file);
        $.ajax({
            data : data,
            type : "POST",
            url : "/uploadSummernoteImageFile",
            processData : false,
            contentType : false,
            success : function(data) {
                $(editor).summernote('insertImage', data.url);
            }
        });
    }
</script>
```

화면 시작 시 썸머노트 실행

이미지 업로드

후기 게시판- 페이징/검색 기능 코드

review_mapper.xml

```
<select id="selectSearchCount" resultType="int"
    parameterType="ReviewDto">
    SELECT COUNT(*) FROM
    PSTM_REVIEWINFO

    <where>
        <if test="reviewtitle!=null">
            REVIEWTITLE LIKE '%' || #{reviewtitle}|| '%'
        </if>
        <if test="trainer!=null">
            TRAINER LIKE '%' || #{trainer} || '%'
        </if>
    </where>

</select>
```

로우 개수 추출(Count)

검색어가 포함 되어있는 리스트 추출

review_mapper.xml

```
<select id="selectSearchReviewPaging" resultType="ReviewDto"
    parameterType="map">


        SELECT REVIEWID, USERID, REVIEWTITLE, REVIEWCONTENT, UPLOADIMG, REGDATE, TRAINERID, TRAINER
        FROM (SELECT SEQ, REVIEWID, USERID, REVIEWTITLE, REVIEWCONTENT, UPLOADIMG, REGDATE, TRAINERID, TRAINER
        FROM (SELECT ROWNUM AS SEQ, REVIEWID, USERID, REVIEWTITLE, REVIEWCONTENT, UPLOADIMG, REGDATE, TRAINERID, TRAINER
        FROM (SELECT * FROM PSTM_REVIEWINFO
        <where>
        <if test="reviewTitle!=null">
            REVIEWTITLE LIKE '%' || #{reviewTitle}|| '%'
        </if>
        <if test="trainerName!=null">
            TRAINER LIKE '%' || #{trainerName} || '%'
        </if>
        </where>
        ))
        WHERE SEQ <![CDATA[ > ]]> #{offset})
        WHERE ROWNUM <![CDATA[<= ]]> #{count}

</select>
```

검색어가 포함 되어있는 리스트 추출

Offset = 현재 페이지 시작 번호

count = 한 페이지당 출력돼야 할 Dto 개수

<검색 기능>
- 마이바티스의 <where> 구문을 이용해 키워드를 가진 값들을 추출했다

<페이징>
- Offset과 count를 입력하여 페이지 번호에 따라 리스트를 잘라서 추출했다
- 이후 컨트롤러에서 페이징 객체와 리스트를 jsp에 전달 했다
- 페이지 번호를 클릭할 시 페이지 번호를 컨트롤러로 전달 한다(초기값 = 1)

3. 시연(영상으로 대체)

https://www.youtube.com/watch?v=TFLQwBOJcLE

4. 소감 및 느낀 점

문제점 및 느낀 점-김선아(후기 게시판, 썸머노트 API 담당)

문제점

1. 글을 작성할 때 영문에서만 폰트를 바꿀수 있었다. 폰트를 추가했지만 input select에 출력이 되지 않았고 그 이유는 섬머노트의 api에 있는 jsp를 수정해야 할 것 같았으나 찾지 못하였다.

2. 글과 사진, 그 외 파일들을 textarea에 한번에 넣고 select 했을 때 같이 출력되게 만들고 싶었지만 그 기능는 스프링을 사용해야 하는것 같았다.
그래서 파일을 넣을수 있는 공간을 따로 만들어 이미지를 업로드 할 수 있도록 대체하였다.

3. 사진을 넣지 않았을 때 사용자의 프로필 사진이나 기본사진이 리뷰 리스트에 출력되도록 만들고 싶었으나 하지 못했다.

느낀점

리뷰 게시판만 했는데 게시판은 수업시간에도 기본적인 기능을 배웠고 다른 페이지를 가면 예시도 많이 볼 수 있으니 쉬울줄 알았다.
근데 페이징은 아직도 모르겠고 이미지 업로드는 내가 이게 왜 되는지 이해가 안된다.....
어찌어찌 성공한건 기적입니다.하느님 감사합니다...
페이징을 도와준 유진이, 이미지 업로드를 도와준 지훈이, 멘탈이 많이 갈려서 이제 스무디라고 불러달라던 민석이, 밤 늦은시간까지 함께 고생한 용승오빠와 잘한다고 다독여준 자바 귀요미 모두모두 감사합니다

문제점 및 느낀 점- 김지훈(마이페이지, Vision API, 파일 업로드 담당)

문제점

1 . 객체감지를 쓸 때 하나의 데이터를 가져오고 싶었는데 못 가져와서 다음에는 원하는 데이터를 가져와 보고 싶다.

2. 이미지 파일을 선택할 때 미리보기를 해보고 싶었는데 객체 감지와 연결하는 부분에서 어떻게 손봐야 될지 모르겠어서 아쉬웠다

느낀점

처음 시작할 때 어떤 것부터 시작해야 될 지 막막했는데 조금씩 기능들이 완성되니 희열을 느꼈다. 내가 무엇이 부족한지 어디서부터 공부해야 될 지 몰랐지만 이번 프로젝트 때 내가 부족한점이 무엇인지 확실히 깨달아서 공부의 방향을 잡을 수 있어 좋았다

문제점 및 느낀 점- 박민석(1:1 화상 강의 담당)

문제점

코드 분할을 너무 못해서 정리도 안되고 어지러웠던 것 같다

느낀점

팀워크나 이야기 할 것들에 대해 잘 생각하고 더 공부해야 할 것 같다

문제점 및 느낀 점-박준범(지도 API, 칼로리 API 담당)

문제점

1. 객체 감지와 api에서의 정확도가 떨어지다보니 생각보다 품질적으로 떨어져서 매우 아쉬운 바이다.

2. api에서 내가 원하는 정보를 딱딱 알맞게 가져오지 못하여서 좀더 확실하게 만들지 못한점이 아쉽다.

3. 칼로리를 계산 시 이름으로만 값을 가져오다 보니 정확도 면에서 많이 떨어진 것 같다

4. 크롤링을 하여서 필요한 정보들을 긁어오는 페이지를 만들고 싶었는데 기술이 부족하여 못한 것이 아쉽다.

느낀점

프로젝트를 진행하면서 생각지 못한곳에서 발생하는 에러나 문제점이 많다고 느끼게 되었다.

그러한 점들을 좀더 신경 쓰고 나의 기술적인 부족한 점을 더 채워 나가야 할 것 같다.

문제점 및 느낀 점-배유진(결제, 채팅 담당)

문제점

1. 보안 측면에서 신경을 잘 쓰지 못한 것 같다. Get방식으로 정보를 보내거나 github에 api 키가 그대로 노출된 경우 등이 있어서 아쉬웠다.

2. Ajax를 이용해 팝업을 띄우는 코드를 작성했는데, 비동기 통신이기 때문에 팝업창에 정보가 제대로 전달되지 않는 경우가 있었다

느낀점

1. 팀원 간 업무 분배를 제대로 하지 못했던 점에 대해서 아쉬움이 있다.

2. 도메인 하나를 공유해서 프로젝트를 진행했다면, 더 수월하게 프로젝트를 진행할 수 있었을 듯 하다

3. 비록 사용해보지 못한 기술이라도, 구글링을 통해 정보를 얻을 수 있기 때문에, 결국 중요한 것은 지금까지 무엇을 배웠느냐가 아니라 원하는 정보를 얻을 때까지 서칭하는 인내심인 것 같았다

문제점 및 느낀 점- 조용승(로그인 기능 담당)

문제점

프로젝트를 마감하면서 내가 하지 못했던 것들 그리고 고쳐야 할 점이 있다면 일단 아이디 비밀번호 찾기, 회원의 정보수정 그리고 이메일을 이용하지 않았다는 점이다

이 밖에도 자바스크립트의 수련부족 등 많은 부분이 부족하다고 느낀 세미 프로젝트였던 것 같다.

느낀점

로그인이 다른 기능에 비해 쉬워 보였지만 하다 보니 로그인이 생각보다 많이 중요 하다는 것을 느꼈다.

프로젝트를 진행하면서 구현해야 했던 기능들이 대부분 로그인하는 것을 전제로 하지 않으면 안되는 것들 이었다

아이디 비밀번호를 입력하면 되는 그런 간단한 문제라고 생각했던 내 생각이 너무 짧았다고 생각했고 신경 써주어야 하는 것들이 많았다.

5. 팀 소개 및 프로젝트 결과

코더차야란?

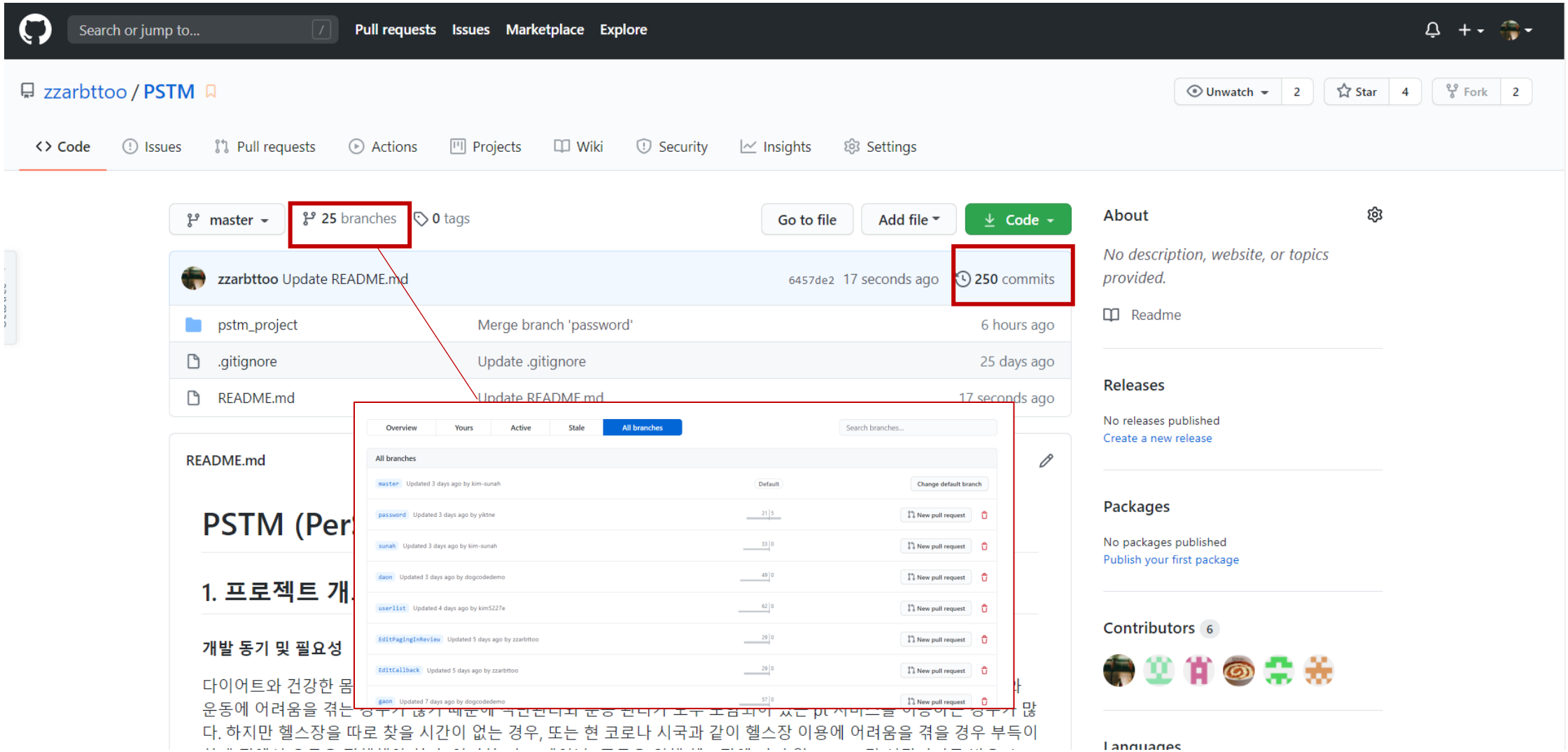
코더차야는 “ 직접 가서 여러 음식을 골라서 주문해서 먹는 포장마차인 코다차야처럼 코더들끼리 인터넷에서 조사하고 의견을 모아 하나의 프로젝트를 만드는 가상의 공간” 이라는 의미를 가지고 있습니다!

코더차야 멤버는?

코더차야는 멤버는 다음과 같이 구성되어 있습니다!

Contributors 6

팀소개 및 프로젝트 결과



https://github.com/zzarbttoo/PSTM

Q & A

감사합니다