Term Project (KJ 쇼핑몰 구현)
1923729 김재영

1. 로그인 페이지

1-1. 로그인 기능

목적: 고객, 관리자는 로그인해야 쇼핑몰 시스템을 이용할 수 있게 만듭니다.
사용법: 로그인 화면에서 id와 pw를 입력하여 로그인을 한다.

#로그인 페이지 구성

```
<fieldset id="loginPage">
        <legend>로그인</legend>
        <p>
            <label for="user_lid">아이디</label>
            <input type="text" name="user_lid" id="user_lid" />
        </p>
        <p>
            <label for="user_lpw">패스워드</label>
            <input type="text" name="user_lpw" id="user_lpw" />
        </p>
        <p>
            <button onclick="login()">로그인</button>
            <button onclick="join()">회원가입</button>
            <button onclick="reInput()">다시입력</button>
        </p>
    </fieldset>
```

로그인 화면은 위와 같이 구성되며 아이디와 패스워드를 입력하여 로그인이 가능합니다. 회원이 아닌 경우 회원가입을 통해서 회원이 될 수 있습니다. 또한 아이디, 패스워드 중에 잘못된 값이 있다면 “다시 입력” 버튼을 통해서 값을 지울 수 있습니다.

1-1.[login()]

```
function login() {
            memberId = $('#user_lid').val(); // 입력한 아이디
            memberPw = $('#user_lpw').val(); // 입력한 비밀번호

            console.log(memberId)
            console.log(memberPw)
            var flag = memberList.filter((item) => {
                console.log(item.id)
                return item.id === memberId && item.pw === memberPw;
            });

            if (memberId === 'admin') {
                alert('환영합니다. 관리자님!');
                document.querySelector('#loginPage').style.display = "none";
                document.querySelector('#announcement').style.display = "none";
                document.querySelector('#adminMode').style.display = "block";
            } else if (flag.length > 0) { // 일치하는 아이디와 비밀번호가 있는지 확인
                alert('환영합니다. 회원님');
                document.querySelector('#loginPage').style.display = "none";
                document.querySelector('#announcement').style.display = "none";
                document.querySelector('#memberMode').style.display = "block";
            } else {
                alert('회원정보가 일치하지 않습니다.');
                document.getElementById('user_lid').value = "";
                document.getElementById('user_lpw').value = "";
            }
        }
```

#announcement는 로그인 페이지의 환영문장, #loginPage는 로그인 페이지, #adminMode는 관리자 페이지, #memberMode는 고객 페이지입니다.

로그인 화면에 있는 id값과 pw값을 받아와서 회원 배열에 존재한다면 flag에 값을 반환합니다.
아이디가 관리자이면 관리자를 환영하는 메시지를 출력하고 로그인 페이지와 로그인 페이지 환영문을 보이지 않게 설정하고 관리자 페이지만 보여줍니다. 관리자가 아닌 회원이 로그인에 성공하면 로그인 페이지와 로그인 페이지 환영문을 보이지 않게 설정하고 고객 페이지를 보여줍니다. 만약 일치하는 회원정보가 없다면 로그인 아이디 값과 패스워드 값을 비우고 회원정보가 일치하지 않다는 메시지를 띄웁니다.

1-2. 회원가입 기능

목적: 고객이 해당 쇼핑몰에 대해 가입하지 않았을 경우 가입하게 만듭니다.
사용법: 회원가입 버튼을 눌러 회원가입 페이지를 보여주며 이름, id, pw, 전화번호를 기록하여 회원정보를 생성합니다.

1-2[join(), joinComplete()]

```
function join() { //회원가입 페이지 보여주기
        document.querySelector('#loginPage').style.display = "none";
        document.querySelector('#joinEx').style.display = "block";

    }

    function joinComplete() {
        memberName = $('#user_jname').val();
        memberId = $('#user_jid').val();
        memberPw = $('#user_jpw').val();
        memberNumber = $('#user_jnumber').val();

        var sameId = false;
        memberList.forEach(function (item, index) { // 중복된 아이디 확인 구문
            if (item.id === memberId) {
                sameId = true;
            }
        });

        if (memberId !== '' && memberPw !== '' && memberNumber !== '') {
            // 아이디, 비밀번호, 전화번호의 제약 조건 확인
            var idRegex = /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;
          var pwRegex = /^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{6,12}$/;
            var numberRegex = /^010\d{8}$/;

            if (idRegex.test(memberId)) {
                if (pwRegex.test(memberPw)) {
                    if (sameId) {
                        alert('중복된 아이디가 있습니다.');
                        document.getElementById('user_jid').value = '';
                        document.getElementById('user_jpw').value = '';
                    } else {
                        if (numberRegex.test(memberNumber)) {
                            var member = {
                                name: memberName,
                                id: memberId,
                                pw: memberPw,
                                number: memberNumber,
                                cartList: [], // 회원별로 cartList 추가
                                purchaseList: []// 회원별로 구매내역 추가
                            };
                            memberList.push(member);
```

```
                                alert('회원가입 성공, 다시 로그인 부탁드립니다.');
                                document.querySelector('#loginPage').style.display = 'block';
                                document.querySelector('#joinEx').style.display = 'none';
                                document.getElementById('user_jname').value = '';
                                document.getElementById('user_jid').value = '';
                                document.getElementById('user_jpw').value = '';
                                document.getElementById('user_jnumber').value = '';
                            } else {
                                alert('전화번호는 "010"으로 시작하고, 총 11자리여야 합니다.');
                                document.getElementById('user_jnumber').value = '';
                            }
                        }
                    } else {
                        alert('비밀번호는 6자 이상, 12자 이하로 입력해야하며, 영어, 숫자, 특수문자
(@$!%*#?&)를 1개 이상 포함해야합니다.');
                        document.getElementById('user_jpw').value = '';
                    }
                } else {
                    alert('아이디는 영어만 가능하고, 이메일 형식(@ 기호 포함)을 따라야합니다.');
                    document.getElementById('user_jid').value = '';
                }
            } else {
                alert('아이디와 패스워드, 전화번호를 모두 입력하세요.');
            }
        }
```

join()함수를 통해 로그인 페이지를 숨기고 회원가입 페이지를 보여줍니다.
joinComplete()함수를 통해 회원 정보를 생성하는데 회원 정보를 생성하기 위해서는
id는 영어만 사용 가능하며 이메일 형식을 따라야합니다. pw는 6~12글자로 영어, 숫자, 특수문자를 각각 1개이상 사용해야 됩니다. 마지막으로 전화번호는 010으로 시작하며 11개입니다.
처리방식은 아이디와 비밀번호를 먼저 문법에 맞는지 확인하고 중복된 아이디가 존재하는지 확인합니다. 마지막으로 전화번호의 문법을 확인하고 전부 올바르다면 name, id, pw, number, carList[], purchaseList[]를 가진 회원객체를 생성하고 회원정보 배열에 넣습니다.

1-3. 재입력 기능

목적: id, pw를 잘못입력하여 백스페이스를 누르기 귀찮을 때 id, pw값 한 번에 초기화 하기 위해 설계되었습니다.
기능: 버튼을 누르면 reInput()함수가 실행되어 로그인 페이지의 id, pw 값을 공백상태로 만든다.

1-3.[reInput()]

```
function reInput() { //입력정보 초기화
```

```
            alert('입력정보가 초기화 되었습니다.');
            document.getElementById('user_lid').value = "";
            document.getElementById('user_lpw').value = "";
        }
```

2. 고객 페이지 기능

```
<fieldset id="memberMode" style="display:none">
        <legend>KJ품목현황</legend>

        <div id="memberDiv"></div>
        <button onclick="changePrivacy()">회원정보변경</h1>
        <button id="shopping" onclick="shoping()">쇼핑하기</button>
        <button id="memberItem" onclick="showMemberItem()">장바구니보기</button>
        <button onclick="purchaseContent()">구매내역 확인</button>
        <button id="memberLogout" onclick="logout()">로그아웃</button>

    </fieldset>
```

고객 페이지는 div객체에 테이블 값을 출력하는 방식으로 고객의 기능들이 구현되어 있습니다. “회원정보를 변경하는 버튼”, 상품들을 보여주고 선택할 수 있는 “쇼핑하기 버튼”, 자신 선택한 상품들을 볼 수 있는 “장바구니 버튼”, 장바구니에 상품들에서 결제가 완료된 상품들을 보여주는 “구매내역 확인 버튼”, 마지막으로 “로그아웃 버튼”이 존재합니다.

2-1. 회원정보 변경

목적: 고객이 자신의 정보에 변동사항이 생기거나 id, pw값을 바꾸기 위해 설계 되었습니다.

기능: 버튼 클릭 시 회원정보를 테이블로 보여주고 값을 변경가능하게 input type =“text”로 주어서 변동된 값을 반영할 수 있습니다.

2-1[changePrivacy(), updateMember(index)]

```
function changePrivacy() {
            var loginMember = memberList.find(function (item) {
                return item.id === memberId;
```

```
            });
            console.log(loginMember)

            var index
            if (loginMember) {
                index = memberList.indexOf(loginMember);
            }

            var privacyTable = "<table>";

            privacyTable  +=  "<tr><th>이름</th><th>아이디</th><th>패스워드</th><th>
전화번호</th><th>조절</th></tr>";

            privacyTable += "<tr>";
            privacyTable += "<td><input type='text' value='" + loginMember.name +
"' id='name_" + index + "'></td>";
            privacyTable += "<td><input type='text' value='" + loginMember.id + "'
id='id_" + index + "'></td>";
            privacyTable += "<td><input type='text' value='" + loginMember.pw + "'
id='pw_" + index + "'></td>"; //value='" + item.price + "' 가격을 표시 // id='price_" +
index + "' 몇번째 품목의 가격인덱스인지 id에 저장
            privacyTable      +=      "<td><input      type='number'      value='"      +
loginMember.number + "' id='phoneNumber_" + index + "'></td>";
            privacyTable += "<td><button onclick='updateMember(" + index + ")'>저장
</button></td>";
            privacyTable += "</tr>";

            privacyTable += "</table>"
    document.getElementById("memberDiv").innerHTML = privacyTable;

        }
function updateMember(index) {
    var nameInput = document.getElementById("name_" + index).value;
        var idInput = document.getElementById("id_" + index).value;
        var pwInput = document.getElementById("pw_" + index).value;
 var phoneNumberInput = document.getElementById("phoneNumber_" + index).value;

            // 회원 정보 변경
            memberList[index].name = nameInput;
            memberList[index].id = idInput;
            memberList[index].pw = pwInput;
```

```
            memberList[index].phoneNumber = phoneNumberInput;

            alert("회원정보가 변경되었습니다.");
        }
```

현재 로그인된 고객 id값을 memberId 값에 저장하고 memberId값을 이용하여 회원배열 객체에서 회원의 객체를 찾은 뒤에 테이블 형식으로 보여주고 input type="text"를 이용해 변동가능하게 값을 만들고 맨 밑에 "저장 버튼"를 누르면 변동된 값들이 id값을 통해서 값을 가져온 후에 해당 고객 객체의 값을 변경합니다. 그리고 회원정보가 변경되었다는 메시지를 출력합니다.

2-2. 쇼핑하기

목적: 고객이 쇼핑몰에서 제공하는 상품들을 구매시키기 위해 설계되었습니다.
사용법: 쇼핑하기 버튼을 클릭하여 상품 테이블들을 보고 개수를 입력하여 "담기 버튼"을 이용하여 장바구니에 저장한다.

2-2.[shoping(), addToCart(index, memberId)]

```
function shoping() {
            memberId = $('#user_lid').val(); // 입력한 아이디
            console.log(memberId)

            var table = "<table>";

           table    +=    "<tr><th>상품종류</th><th>품목이름</th><th>가격</th><th>수량</th><th>장바구니</th></tr>";
            inventoryList.forEach(function (item, index) {
                table += "<tr>";
                table += "<td>" + item.type + "</td>";
                table += "<td>" + item.name + "</td>";
                table += "<td>" + item.price + "</td>";
                table += "<td>" + item.num + "</td>";
                table  +=  "<td><input  type='number'  id='quantity_" + index + "' min='0' max='" + item.num + "'><button onclick='addToCart(" + index + ", \"" + memberId + "\")'>담기</button></td>";
                table += "</tr>";
            });

            table += "</table>";
```

```
    document.getElementById("memberDiv").innerHTML = table;
}

function addToCart(index, memberId) {
   var quantityInput = document.getElementById("quantity_" + index);
    var quantity = parseInt(quantityInput.value);

    if (isNaN(quantity) || quantity <= 0) {
        alert("유효한 수량을 입력해주세요.");
        return;
    }

    var item = inventoryList[index];

    if (quantity > item.num) {
        alert("재고 수량보다 많이 담을 수 없습니다.");
        return;
    }



    for (var i = 0; i < memberList.length; i++) {
        if (memberList[i].id === memberId) {
            member = memberList[i];
            break;
        }
    }

    if (!member) {
        alert("회원을 찾을 수 없습니다.");
        return;
    }

    var cartItem = {
        type: item.type,
        productId: item.name,
        price: item.price,
        num: quantity
    };

    console.log(cartItem)
    member.cartList.push(cartItem);
```

```
            document.getElementById('quantity_' + index).value = '';
            alert("장바구니에 상품이 추가되었습니다.");
        }
```

쇼핑하기 버튼을 클릭하여 shoping()함수를 호출하여 쇼핑몰에서 제공하는 상품들을 테이블 형식으로 보여줍니다. 테이블 표 마지막 열에는 장바구니라는 이름이 있는데 장바구니의 해당하는 열들은 수량을 입력하는 input type= number를 통해 값을 입력하고 “담기 버튼”을 이용하여 고객이 구매하고 싶은 상품들을 장바구니 담을 수 있습니다. “담기 버튼”을 클릭하면 addToCart() 함수가 호출되면서 index, memberId 값을 통해 해당 고객의 장바구니인 cartList에 주문한 상품 기록이 저장됩니다. 또한 고객이 상품의 수량보다 초과, 미만인 수로 입력한 경우 상품의 개수를 벗어난다는 의미에 메시지를 출력합니다.

2-2 장바구니

목적: 고객이 선택한 상품들을 결제하기 전 단계이고 장바구니 페이지에서 결제를 진행합니다. 또한 자신이 주문한 내역을 취소할 수 있습니다.

사용법: 장바구니 버튼을 클릭하여 고객이 장바구니에 담은 상품 목록들을 보여준다.

```
    <fieldset id="memberBasket" style="display:none">
        <legend>장바구니 현황</legend>

        <div id="basketTable"></div>

        <button onclick="purchase()">구매확정</button>
        <button onclick="cancel()">구매보류</button>
        <button id="shopping" onclick="reShoping()">쇼핑하기</button>

    </fieldset>
```

2-2[showMemberItem(), cancelOreder(index, memberId), purchase(), reshoping(), cancel()]

```
function showMemberItem() { //장바구니 보여주기
            memberId = $('#user_lid').val(); // 입력한 아이디

            for (var i = 0; i < memberList.length; i++) {
                if (memberList[i].id === memberId) {
                    member = memberList[i];
                    break;
                }
```

```
            }

            if (!member) {
                alert("멤버를 찾을 수 없습니다.");
                return;
            }

            var table = "<table>";

            table += "<tr><th>상품종류</th><th>품목이름</th><th>가격</th><th>수량
</th><th>주문 취소</th></tr>";
            member.cartList.forEach(function (item, index) {
                table += "<tr>";
                table += "<td>" + item.type + "</td>";
                table += "<td>" + item.productId + "</td>";
                table += "<td>" + item.price + "</td>";
                table += "<td>" + item.num + "</td>";
                table += "<td><button onclick='cancelOrder(" + index + ", \"" +
memberId + "\")'>주문 취소</button></td>";
                table += "</tr>";
            });

            table += "</table>";

            document.querySelector('#memberBasket').style.display = "block";
            document.querySelector('#memberMode').style.display = "none";
            document.getElementById("basketTable").innerHTML = table;
        }

        function cancelOrder(index, memberId) {
            var member;
            var item;

            for (var i = 0; i < memberList.length; i++) {
                if (memberList[i].id === memberId) {
                    member = memberList[i];
                    break;
                }
            }

            if (!member) {
```

```
        alert("멤버를 찾을 수 없습니다.");
        return;
    }

    if (index >= 0 && index < member.cartList.length) {
        item = member.cartList[index];
        member.cartList.splice(index, 1); // 주문 취소
        alert(item.productId + " 상품이 주문 취소되었습니다.");
    } else {
        alert("잘못된 주문 취소 인덱스입니다.");
    }

    showMemberItem();
}


function purchase() {
    memberId = $('#user_lid').val(); // 입력한 아이디

    for (var i = 0; i < memberList.length; i++) {
        if (memberList[i].id === memberId) {
            member = memberList[i];
            break;
        }
    }

    if (!member) {
        alert("멤버를 찾을 수 없습니다.");
        return;
    }

    var newCartList = []; // 새로운 주문 내역 배열

    // cartList에서 각 상품 정보를 가져와 처리
    for (var i = 0; i < member.cartList.length; i++) {
        var cartItem = member.cartList[i];

        // inventoryList에서 상품 찾기
        var itemIndex = inventoryList.findIndex(function (item) {
            return item.name === cartItem.productId;
        });
```

```
            if (itemIndex !== -1) {
                var basketItem = inventoryList[itemIndex];

                if (basketItem.num >= cartItem.num) {
                    // 상품 수량 감소
                    basketItem.num -= cartItem.num;

                    // 판매량 증가
                    basketItem.sales += cartItem.num;

                    //구매내역 추가하기
                    member.purchaseList.push(cartItem);
                } else {
                    alert("상품 수량이 부족합니다: " + basketItem.name);
                    // 주문 내역 유지
                    newCartList.push(cartItem);
                }
            } else {
                alert("상품을 찾을 수 없습니다: " + cartItem.productId);
                // 주문 내역 유지
                newCartList.push(cartItem);
            }
        }

        member.cartList = newCartList; // 새로운 주문 내역으로 업데이트
        alert('구매가 완료되었습니다.');
        showMemberItem();
    }
```

showMemberItem(), cancelOreder(index, memberId), purchase()]
showMemberItem()함수를 통해서 고객의 개인 장바구니를 테이블 형태로 보여주고
모든 상품 주문목록옆에 “주문 취소” 버튼을 통해 언제든지 상품 주문을 취소할 수 있습니다.
“주문 취소”버튼을 누르면 cancelOreder(index, memberId)함수를 실행하여 상품주문목록을 index값을 통해 지정하고 memberId값을 통해 해당 고객의 상품주문목록을 제거합니다.
그리고 다시 showMemberItem()을 통해 주문상품 목록들을 최신화합니다.
만약 고객이 “결제하기 버튼”를 누르면 purchase()함수를 실행하여 상품목록이 들어있는 inventoryList 객체의 상품들을 최신화 시킵니다. 또한 고객의 결제내역을 저장하는 purcahseList 객체에 결제내역이 저장됩니다.

```
function reShoping() {
            document.querySelector('#memberBasket').style.display = "none";
            document.querySelector('#memberMode').style.display = "block";
            shoping();
        }

  function cancel() { //장바구니 페이지 나오기


            document.querySelector('#memberBasket').style.display = "none";
            document.querySelector('#memberMode').style.display = "block";
        }
```

두 개의 코드들은 각각 장바구니 페이지에 존재하는 “다시 쇼핑하기”, “결제 보류” 버튼입니다. “다시 쇼핑하기 버튼은” 상품목록 테이블을 보여주면서 최신화하기 위해 shoping()함수를 호출하고 결제보류는 아직 결제하지 않았기 때문에 상품목록 테이블을 그대로 보여줍니다.

2-3. 구매내역 확인

목적: 고객이 자신의 결제내역들을 확인하기 위해 설계되었습니다.

사용법: “구매내역 확인” 버튼을 누르면 자신이 결제했던 상품내역이 테이블의 형태로 보여줍니다.

2-3[purchaseContent()]

```
function purchaseContent() {
            memberId = $('#user_lid').val(); // 입력한 아이디

            var purchaseTable = "<table>";
            purchaseTable += "<tr><th>종류</th><th>상품이름</th><th>가격</th><th>구입한 개수</th></tr>";

            member.purchaseList.forEach(function (item, index) {
                purchaseTable += "<tr>";
                purchaseTable += "<td>" + item.type + "</td>";
                purchaseTable += "<td>" + item.productId + "</td>";
                purchaseTable += "<td>" + item.price + "</td>";
                purchaseTable += "<td>" + item.num + "</td>";
                purchaseTable += "</tr>";
            });

            purchaseTable += "</table>";
            var message = member.name + "회원님의 구매내역입니다.";
          document.getElementById("memberDiv").innerHTML = "<p>" + message + "</p>" + purchaseTable;
        }
```

purchaseContent() 함수를 호출하여 고객의 로그인 아이디값을 받아온 뒤에 그 값을 통해 고객의 구매내역인 purchaseList 객체 배열을 이용하여 자신이 주문했던 상품기록을 볼 수 있게 만듭니다.

3. 관리자 페이지

기능: 관리자는 상품의 매출현황, 상품의 값과 개수변동 및 상품 추가, 제거, 회원 정보 조회를 통해 쇼핑몰을 관리한다.
사용법: 로그인 페이지에서 관리자의 id,pw를 입력하여 로그인한다.

```
<fieldset id="adminMode" style="display:none">
        <legend>관리자 기능</legend>
        <h1 onclick="showSales()">품목현황</h1> <!-- 각 품목마다 판매가격을 통해 수익량 확인, 판매량 확인-->
        <h1 onclick="showItem()">재고관리</h1> <!-- 육류, 과일, 과자 3개로 분할하고 수량및 가격변동-->
        <h1 onclick="showMember()">회원정보보기</h1> <!-- 회원정보 확인-->


        <div id="itemTable"></div>
        <p></p>
        <button id="adminLogout" onclick="logout()">로그아웃</button>
    </fieldset>
```

3-1. 상품의 매출현황

기능: 각 상품의 팔린 개수와 매출량을 테이블을 통해 한눈에 알아보기 위해 설계 되었습니다.
사용법: “품목현황 버튼”을 클릭하여 showSales()함수를 실행합니다.

```
function showSales() { //매출현황 보기
            var salesTable = "<table>";

            salesTable += "<tr><th>상품종류</th><th>품목이름</th><th>가격</th><th>판매수량</th><th>매출액</th></tr>";
            inventoryList.forEach(function (item, index) {
                salesTable += "<tr>";
                salesTable += "<td>" + item.type + "</td>";
                salesTable += "<td>" + item.name + "</td>";
                salesTable += "<td>" + item.price + "</td>";
                salesTable += "<td>" + item.sales + "</td>";
                salesTable += "<td>" + item.price * item.sales + "</td>";
                salesTable += "<td></td>"
```

```
            salesTable += "</tr>";
        });

        salesTable += "</table>";
        document.getElementById("itemTable").innerHTML = salesTable;

    }
```

모든 상품들의 정보가 담긴 inventoryList 상품배열객체를 통해 각 상품의 판매량과 매출액을 테이블을 통해 보여줍니다.

3-2. 상품 변동,추가,제거

기능: 각 상품의 수량과 가격을 조정하거나, 기존의 상품을 제거, 새로운 상품을 추가

사용법: “재고관리 버튼”을 통해 showItem()호출하여 테이블을 출력한다.

```
function showItem() {
        loginId = $('#user_lid').val(); // 입력한 아이디
        console.log(loginId)
        var table = "<table>";

        table += "<tr><th>상품종류</th><th>품목이름</th><th>가격</th><th>수량</th><th>조절</th></tr>";
        inventoryList.forEach(function (item, index) {
            table += "<tr>";
            table += "<td>" + item.type + "</td>";
            table += "<td>" + item.name + "</td>";
            table += "<td><input type='number' value='" + item.price + "' id='price_" + index + "'></td>"; //value='" +
item.price + "' 가격을 표시 // id='price_" + index + "' 몇번째 품목의 가격인덱스인지 id에 저장
            table += "<td><input type='number' value='" + item.num + "' id='num_" + index + "'></td>";
            table += "<td><button onclick='updateItem(" + index + ")'>저장</button></td>";
            table += "<td><button onclick='removeItem(" + index + ")'>제거</button></td>"; // 제거 버튼 추가
            table += "</tr>";
        });

        // inventoryList에 품목을 추가할 수 있게만드는 행을 추가
        table += "<tr>";
        table += "<td><input type='text' id='newType'></td>";
        table += "<td><input type='text' id='newName'></td>";
        table += "<td><input type='number' id='newPrice'></td>";
        table += "<td><input type='number' id='newNum'></td>";

        table += "<td><button onclick='addItem()'>추가</button></td>";
        table += "</tr>";

        table += "</table>";

        document.getElementById("itemTable").innerHTML = table;
    }
    function removeItem(index) { // 인덱스 번호를 받아 inventoryList의 상품을 제거
        inventoryList.splice(index, 1); // 해당 인덱스의 상품을 제거
        alert("상품이 제거되었습니다.");
        showItem(); // 변경된 상품 목록을 다시 표시
    }
```

```
function updateItem(index) { //인덱스 번호를 받아 inventoryList품목을 고침
    var priceInput = document.getElementById("price_" + index); //테이블에서 아이디가 가격_n인 항목을 가져옴
    var numInput = document.getElementById("num_" + index);

    var newPrice = parseInt(priceInput.value);
    var newNum = parseInt(numInput.value);

    if (isNaN(newPrice) || isNaN(newNum)) { //둘 중 하나라도 숫자가 아니면 오류
        alert("올바른 가격과 수량을 입력해주세요.");
        return;
    }

    inventoryList[index].price = newPrice; //n번째 항목의 값을 변경
    inventoryList[index].num = newNum;


    alert("상품 정보가 업데이트되었습니다.");
}

function addItem() {
    var newType = document.getElementById("newType").value;
    var newName = document.getElementById("newName").value;
    var newPrice = document.getElementById("newPrice").value;
    var newNum = document.getElementById("newNum").value;


    if (newType === '' || newName === '' || isNaN(newPrice) || isNaN(newNum)) { //값이 존재해야하고 문자면 안됨
        alert("올바른 상품 정보를 입력해주세요.");
        return;
    }

    var newItem = {
        type: newType,
        name: newName,
        price: parseInt(newPrice),
        num: parseInt(newNum),
        sales: 0
    };

    inventoryList.push(newItem);

    alert("새로운 상품이 추가되었습니다.");
    showItem(); // 추가한 상품을 포함한 품목 테이블을 다시 표시
}
```

showItem()함수를 호출하여 쇼핑몰의 모든 상품목록을 보여주고 각 상품의 num, price값을 변동가능한 input type= number로 설정하였다. 각 상품목록마다 “저장 버튼”을 통해서 updateItem(index) 함수를 실행시켜 각 상품의 인덱스, id값을 참고하여 값을 변동시킵니다. 각 상품목록마다 “제거 버튼”을 통해서 removeItem(index) 함수를 실행시켜 인덱스 번호를 이용하여 상품목록을 제거한 후에 테이블을 최신화 시키기 위해 showItem()함수를 실행합니다. 마지막으로 테이블 마지막 행에는 모든 값이 공란으로 주어져 있으며 그 값들을 전부 채워서 “추가 버튼”을 누르면 addItem() 함수를 실행시켜 새로운 상품목록이 inventoryList 객체배열에 추가됩니다.

3-3.회원정보보기

기능: 모든 회원들의 이름, id, pw, 전화번호를 볼 수 있습니다.

사용법: “회원정보보기”버튼을 누르면 showMember()함수가 호출되어 회원정보가 테이블로 보여집니다.

3-3[showMember(), changePrintNumber(phoneNumber)]

```
function showMember() { //회원정보 보여주기
        var memberTable = "<table>";

        memberTable += "<tr><th>이름</th><th>아이디</th><th>비밀번호</th><th>전화번호</th></tr>";
        memberList.forEach(function (item, index) {
            if (item.id == 'admin') return;
            memberTable += "<tr>";
            memberTable += "<td> " + item.name + " </td>";
            memberTable += "<td> " + item.id + " </td>";
            memberTable += "<td> " + item.pw + " </td>";
            memberTable += "<td> " + changePrintNumber(item.number) + " </td>";
            memberTable += "</tr>";
        });

        memberTable += "</table>";

        document.getElementById("itemTable").innerHTML = memberTable;

    }

    function changePrintNumber(phoneNumber) { //폰번호 출력형식바꾸기
        var changePhoneNumber = phoneNumber.slice(0, 3) + "-" + phoneNumber.slice(3, 7) + "-" +
phoneNumber.slice(7);
        return changePhoneNumber;
    }
```

showMember()을 통하여 회원의 정보를 테이블로 보여주고 changePrintNumber(phoneNumber)을 통하여 테이블의 마지막 요소인 회원의 전화번호를 ###-####-####형태로 ‘-’를 붙여서 예쁘게 폰번호를 출력합니다.

4. 팝업창 기능

목적: 고객님께 감사의 인사말씀을 인사드리기 위해 설계되었습니다.

사용법: 창의 위치상태에 따라 창의 왼쪽 끝에 자동으로 팝업창이 생깁니다.

```
<body>
    <script src="js/popups.js"></script>
      // 수많은 기능들.....
</body>
```

#popups.js

```
function openPop() {

    var newWin = window.open("popup.html", "", "width=400,height=400");
    var currentScreenX = window.screenX;
```

```
    var currentScreenY = window.screenY;

    window.localStorage.setItem('screenX', currentScreenX);
    window.localStorage.setItem('screenY', currentScreenY);

    if (newWin == null) {
      alert('팝업이 차단');
    } else {

      var storedScreenX = window.localStorage.getItem('screenX');
      var storedScreenY = window.localStorage.getItem('screenY');


      newWin.moveTo(storedScreenX, storedScreenY);
    }
  }

  window.onload = openPop;
```

#popup.html

```
<!DOCTYPE html>
<html lang="en">
<style>
    #content {
        max-width: 600px;
        margin: 0 auto;
        padding: 20px;
        background-color: #fff;
        border: 1px solid #ccc;
        border-radius: 5px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        margin-top: 50px;
    }

    h1 {
        color: #333;
        text-align: center;
    }

    p {
```

```
        color: #666;
        line-height: 1.5;
        margin-bottom: 15px;
    }

    #close {
        text-align: center;
        margin-top: 30px;
    }

    #close a {
        color: #999;
        text-decoration: none;
    }

    #close a:hover {
        text-decoration: underline;
    }
</style>

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>

    <div id="content">
        <h1>KJ쇼핑몰 안내사항</h1>

        <p>오늘도 저희 매장을 찾아오신 손님께 심심한 감사인사를 드립니다.</p>
        <p>더 좋은 서비스를 줄 수 있게 더욱 더 발전하겠습니다.</p>
        <p id="close"><a href="javascript:window.close();">창닫기</a></p>
    </div>

</body>

</html>
```

js의 폴더에서 popups.js를 실행하여 popup.html를 통해 로그인 페이지에서 팝업창을 출력합니다.

이상으로 KJ쇼핑몰의 설계명세서 였습니다.

1923729 김재영