퀵오스크

Project Information

프로젝트 명	퀵오스크
개발기간	2022. 11. 16 - 2022. 11. 18
참여인원	4명
담당업무	기획, 웹 설계, 디자인, 기능
담당기능	메뉴주문, 회원가입, 로그인, 재고관리

Used Technologies

운영체제	Windows 11 Pro
언어	Java, css, Java FX
툴	Java 8 (eclipse), JAVA FX(Scene Builder)
데이터베이스	Oracle DB(18CEx)

목차

| **Project Introduction**

프로젝트 주제

허들없는 디지털 단말기 퀵오스크

퀵오스크는 퀵(Quick)과 키오스크를 합친 말로 판매 및 주문을 빠르고 간편하게 할 수 있다는 의미를 가지고 있다.

< 핵심 기능 **>**

- 직관적인 인터페이스로 판매 및 구매를 손쉽게 할 수 있다.
- 멤버십 포인트 기능으로 회원관리가 가능하며 회원 가입을 편하게 할 수 있다.
- 재고가 소진된 경우, 알림 메시지가 나와 재고관리가 수월하다.

프로젝트 주제 선정 이유

<u>키오스크 현황</u>

키오스크는 햄버거, 치킨 등을 파는 패스트푸드 매장를 비롯해 카페, 편의점, 기차역, 공항 등에서까지 널리 활용되고 있으며 사업자와 소비자의 입장에 따른 현황은 아래와 같다.

[사업자 입장]

- 최저임금 상승으로 인한 약 1.5명, 약 평균 100만원 이상의 인건비 절감효과
- 월 15만원 수준에 쉽게 대여가 가능해 초기 부담비용 감소
- 정확한 매출 정산으로 부정행위 및 현금매출 누락 방지
- 진상 손님을 직접 응대하지 않아 매장 종사자 보호가능

[소비자 입장]

- 청각장애인은 대면일 때보다 비교적 빠르고 정확하게 주문 가능
- 사용자 입장을 전혀 고려하지 않은 인터페이스로 상대적 기술소외계층인 고령층 외에도 저연령층에서도 불편함을 호소
- 고연령층이 불편한 점으로는 상품 결제 시 복잡한 단계(51.4%), 다음 단계 버튼을 찾기 어려움(51.0%), 뒷사람 눈치가 보임(49.0%) 그림및 글씨가 안 보임(44.1%) 순으로 어려움을 호소

점차 무인화 시대로 나아가면서 카페나 음식점 등 일상에서 키오스크의 사용이 늘고 있다. 그런데 키오스크가 친숙하지 않은 사람들은 불가피하게 주문에 시간이 오래 걸리거나, 혹은 카운터에서 매장 직원을 호출한다. 이러한 상황은 주문하는 당사자에게는 뒤 차례에서 기다리고 있을 손님들에게 무언의 압박을 받기도 하며 피크 타임때에는 복잡한 매장을 더욱 복잡하게 만들기 때문에, 키오스크에 대한 부정적인 견해를 갖게 한다. 이러한 부분들을 해소하고자 사용하기 쉬운 키오스크를 주제로 선정하였다.

|요건 정의

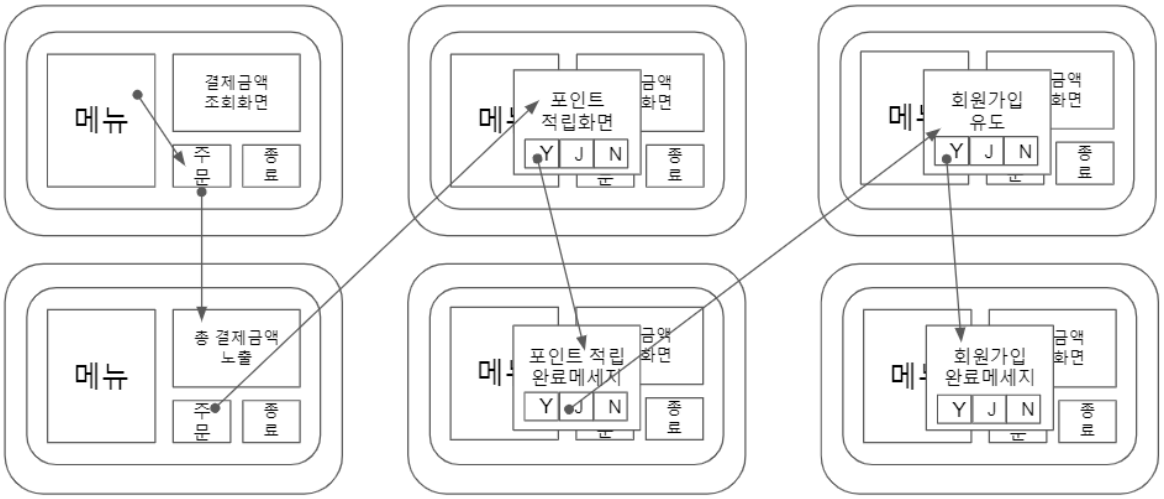
기능 요구 사항 정의

- 텍스트필드에 공백으로 입력 시 경고 메시지가 뜬다.
- 주문 수량 텍스트 필드는 모두 기본 0으로 세팅한다.
- 주문 수량 total 0개로 주문 시 경고 문구가 뜬다.
- 주문 수량에 문자 입력 시 경고 문구가 뜬다.
- 원료 재고가 부족한 메뉴 주문 시 주문 불가 메시지가 뜬다.
- 각 메뉴마다 menu_id를 갖고 있고, 이는 문자열값이다.
- 각 원료마다 ingredient_id를 갖고 있다.
- 각 주문마다 주문날짜 + 주문순서 + 주문메뉴로 구성된 order_id를 갖고 있다.
- 주문하지 않고 포인트 적립 시도하는 것을 막고 경고 메시지를 띄운다.
- 이름, 전화번호를 기입하지 않고 포인트 적립이나 조회, 회원 가입 시도 시 경고 메시지가 뜬다.
- 이름이 동일한 회원 가입이 가능하다.
- 이미 존재하는 전화번호로 회원가입 시도 시 경고문구가 뜬다.
- 존재하지 않는 전화번호로 포인트 적립이나 포인트 조회 시도 시 경고문구가 뜬다.
- DB의 회원정보 table은 보안을을 위해 view 이름이나 column 이름을 은닉하여 사용한다.
- DB는 새로운 사용자 계정을 생성한다.
- DB는 새로운 table을 생성한다.

개발 시 요구사항

- JAVA 8 (eclipse)
- DAO[data access object] pattern
- JDBC (ResultSet / PreparedStatement 사용)
- Oracle DB (11G OR 18CEx 0R Cloud)
- 패키지 2개 이상 사용
- JAVA FX (Scene Builder)

| 퀵오스크 와이어 프레임



| 퀵오스크 상세 기획

총 3가지의 *view point*로 기획을 진행하였으며 목록은 아래와 같다.

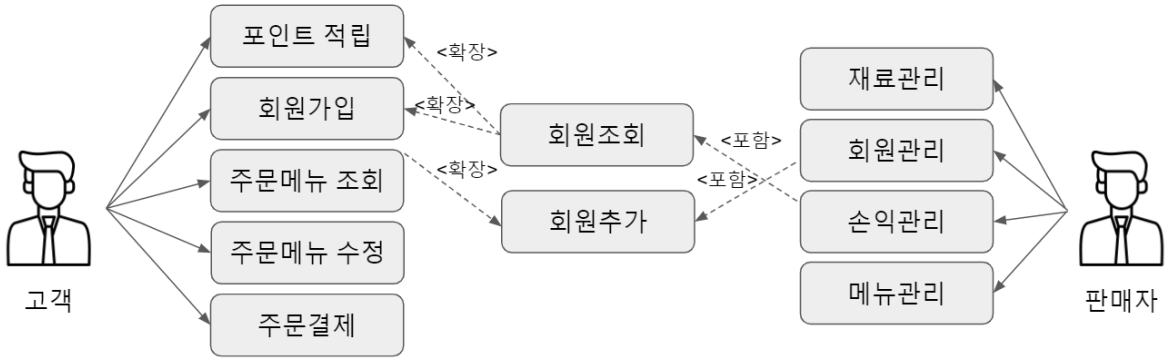
view point 1. 시스템과 사용자의 상호작용

view point 2. 관계형 DB모델링

view point 3. SW설계

view point 1. 시스템과 사용자의 상호작용에 관한 구도는 아래와 같다.

: ***Usecase Diagram***



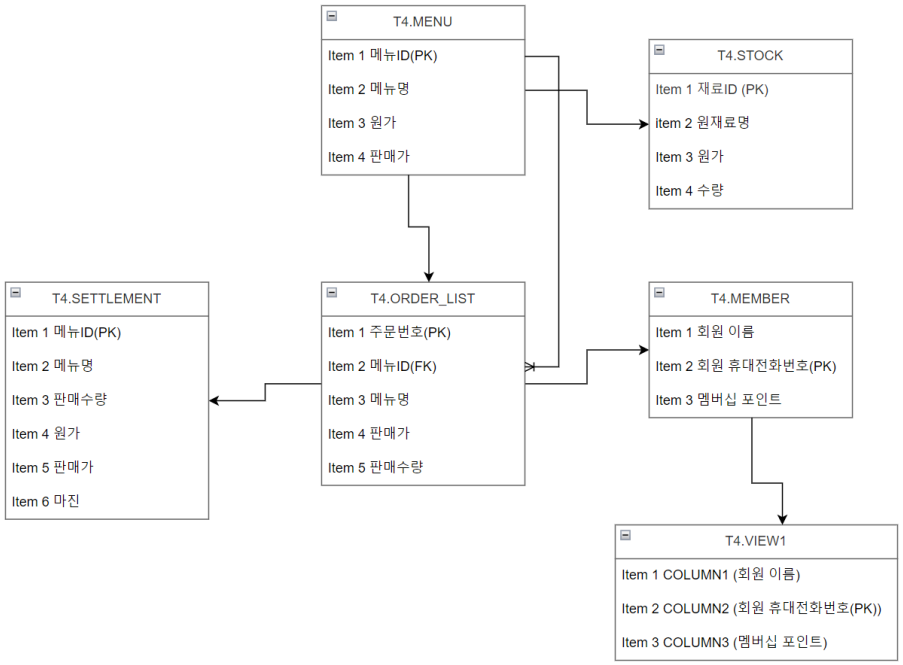
쿽오스크는 무인 주문 기계로서 상품의 판매나 구입을 간편하게 하기 위해 만들어진 디지털 단말기이다. 고객은 화면에 나와있는 메뉴와 옵션을 통해 간편하게 구매를 할 수 있으며, 판매자는 인건비를 절감하고 손님 응대시간의 효율성을 높이며,, 상품판매를 쉽게 진행할 수 있다. 쿽오스크를 사용하는 사용자는 일반 고객(비회원), 회원으로 나뉜다. 고객은 메뉴를 선택하거나, 주문을 결제하고 특정 판매점의 포인트를 적립하기 위해 회원으로 가입할 수 있다. 판매자는 메뉴 관리(등록)나 회원관리(조회, 추가), 재료관리, 손익관리 등과 같은 활동을 수행한다.

사용자 *To-do Action*

- 주문 수량은 모두 기입해야 한다. (디폴트값은 0으로 세팅)
- 주문 수량은 숫자로만 기입해야 한다. (문자값 입력할 경우 경고 메시지 노출)
- 주문 수량은 전체에서 1개 이상 주문해야 한다. (전부 0개일 경우 경고 메시지 노출)
- 이름, 전화번호는 모두 기입해야 한다. (누락 시 경고 메시지 노출)
- 회원 가입 시, 전화번호는 다른 사용자와 중복될 수 없다.
- 회원 가입 시, 회원가입창에 사용자는 이름, 전화번호를 입력한다.

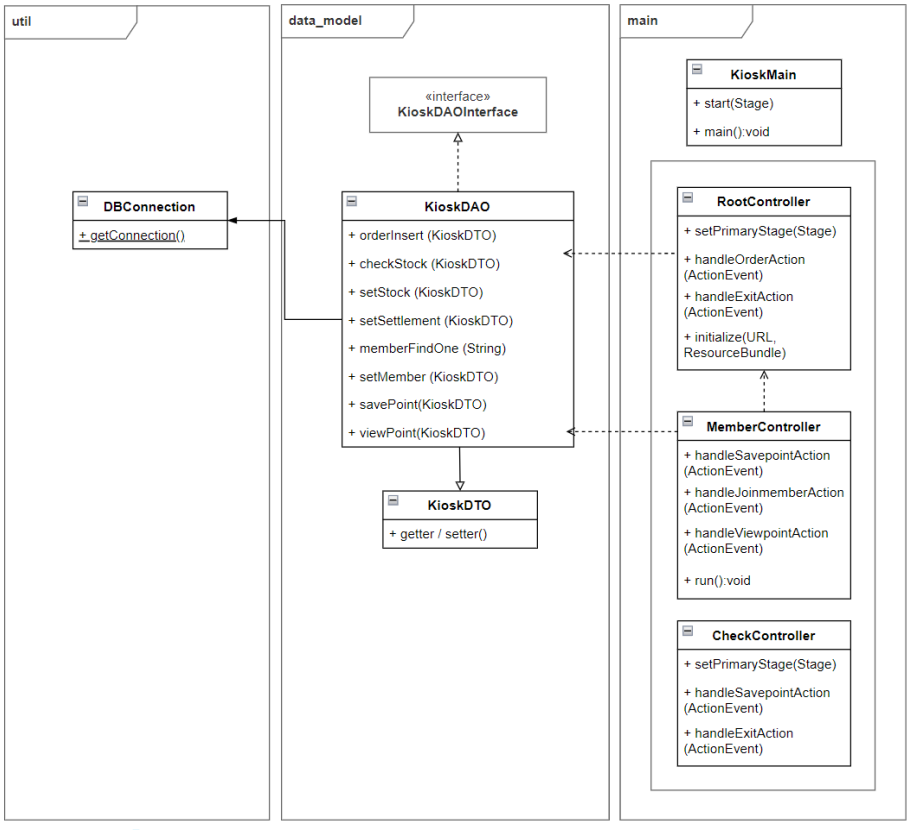
view point 2. DB모델링

: *Logical schema*



view point 3. SW설계

: Class Diagram



| 퀵오스크 상세 설계

: Class Diagram 상세

[package] data_model

<파일명*:* KioskDAO >

KioskDAO : DB Connection 객체를 통해 반환된 Connection 타입 객체 변수 conn을 통해 데이터베이스와 연결하여 데이터를 조회하거나 조작하는 기능을 전담하는 객체.
DAO에 포함된 메소드는 PreparedStatement class 객체와 resultSet class 객체를 사용하여 데이터베이스에 쿼리문을 보내 값을 저장하거나, 가져올 수 있음.

orderInsert : public void orderInsert(KioskDTO newKioskOrderDTO)
- 주문내역을 T4.order_list에 insert하는 메소드로 사용자가 입력한 주문내역을 T4.order_list에 쿼리문으로 보내는 메소드

checkStock : public int checkStock(KioskDTO checkStockDTO)
- 재고량을 T4.Stock에서 select하는 메소드로 T4.stock를 조회하고 그 값을 반환하는 메소드

setStock : public void setStock(KioskDTO setStockDTO)
- 재고량을 T4.Stock에 update하는 메소드로 T4.stock의 available_stock값을 사용자의 주문내역중 수량에 비례하여 차감하는 메소드.

setSettlement : public void setSettlement(KioskDTO setSettlementDTO)
- 판매수량과 마진을 T4.settlement에 update 하는 메소드

setMember : public void setMember(KioskDTO setMemberDTO)
- 회원가입을 통해 사용자가 입력한 이름과 휴대전화 번호를 T4.view1의 column1, column2에 각각 추가하여 insert하는 메소드

memberFindOne : public boolean memberFindOne(String guest_phone)

- 사용자가 멤버십 가입을 할때 입력한 휴대전화 번호가 T4.view1의 column3에 있는지 select count하고 그 여부를 boolean isExist값으로 반환하는 메소드

savePoint : public void savePoint(KioskDTO savePointDTO)
- 사용자가 적립한 멤버십 포인트를 T4.view의 column3에 update하는 메소드

viewPoint : public String viewPoint(KioskDTO savePointDTO)
- 멤버십 포인트를 조회하기 위해 사용자가 입력한 휴대전화 번호를 T4.view의 column2에서 조회하고, 값이 있을 경우 T4.view의 column3을 조회후 출력, 만약 저장된 값이 없으면 안내 메시지 "회원 정보가 없습니다" 출력하는 메소드

<파일명*:* KioskDTO>
KioskDTO : 데이터 교환을 위해서 각 테이블의 컬럼값을 필드로 정의한 클래스

<파일명*:* KioskDAOInterface>
KioskDAOInterface : KioskDAO에 대한 사용 규칙을 정의한 인터페이스

[package] main

<파일명*:* KioskMain>
KioskMain : 시작화면을 출력하는 클래스.
start : public void start(Stage primaryStage) throws Exception
- Parent root = FXMLLoader.load(getClass().getResource("Root.fxml"))코드로 FXMLLoader의 정적 메소드인 load(매개값으로 URL 객체가 옴.)를 이용해서 FXML파일을 읽어들여 선언된 내용을 객체화함. load() 메소드가 반환하는 실제 객체는 FXML 파일에서 루트 태그로 선언된 컨테이너이며, Scene을 생성할 때 매개값으로 사용됨.Scene에 root 객체를 심어서 primaryStage의 setScene() 메소드에 매개값으로 주어 화면을 세팅하고, show() 메소드로 화면 실행.
scene.getStylesheets().add(getClass().getResource("app.css").toString())코드를 통해 fxml 파일을 꾸미는 css 파일을 등록.
main : public static void main(String[] args) {

launch(args); }

- KioskMain 객체 생성 및 메인 윈도우를 생성하는 메소드

※***Controller*** *: fxml*파일에 대한 기능을 관리하는 클래스이며 *'@FXML private (*컨트롤러 이름*)* 태그명*'*형식으로 ***fxml***파일에 선언된 태그를 자바코드로 변환하여 실행한다.

<파일명*:* RootController >

RootController : 메인 윈도우 주문 화면을 컨트롤하는 클래스이며, Root.fxml 파일의 기능을 자바코드로 구현하고 사용자가 입력한 메뉴의 수량에 따라 주문 및 계산을 할 수 있고, 멤버십 포인트 적립값을 저장하며 재고가 부족할 경우 주문이 불가하다고 알린다. 주문내역 및 재고수량값을 데이터베이스와 연동하여 추가 및 수정하는 기능을 보유한 클래스. 성공적으로 주문하기 버튼 클릭 시 다이얼로그로 Check.fxml파일을 띄우는 기능도 보유하고 있다.

setPrimaryStage : public void setPrimaryStage(Stage primaryStage)

- 커스텀 다이얼로그인 CheckController.fxml을 띄우기 위한 사전 작업으로 Stage 세팅 메소드 직접 생성. 컨트롤러에서 다이얼로그를 실행하기 위해서 primaryStage가 필요.

handleOrderAction : public void handleOrderAction(ActionEvent event)

- 사용자가 입력한 값을 주문수량으로 적용하여 T4.order_list로 값을 추가하고, if문

if(americanoCnt > americanoAvailable) {

Platform.runLater(new Runnable()을 이용하여 사용자의 주문 수량이 재고보다 많을 경우 안내메세지 "* 재고가 부족합니다" 출력하고, if문

if (americanoCnt > 0) {

KioskDTO newOrderDTO = new KioskDTO("101", "101", "아메리카노", 4000, americanoCnt);

kioskArrayList.add(newOrderDTO);

KioskDTO newStockDTO = new KioskDTO(2*americanoCnt, 201);

stockArrayList.add(newStockDTO) 을 이용하여 T4.order_list에 주문내역 입력 및 T4.stock의 available_stock 값을 수정하고,

(AnchorPane)FXMLLoader.load(getClass().getResource("Check.fxml") 구문을 이용하여 멤버십 적립 여부 확인을 하는 창인 Check.fxml을 실행하는, 전체적인 이벤트를 담당하는 메소드

handleExitAction : public void handleExitAction(ActionEvent event)
- 프로그램을 종료하는 이벤트 메소드

initialize : public void initialize(URL location, ResourceBundle resources)
- 컨트롤러는 Initializable 인터페이스를 반드시 상속받아야하므로 컨트롤러 class는 기본적으로 initialize 추상메소드를 오버라이딩 해야한다. initialize 추상메소드를 오버라이딩하는 메소드.

<파일명*:* MemberController>
MemberController : 멤버십 적립 및 조회, 회원가입 화면을 컨트롤하는 클래스.
사용자가 입력한 이름값과 번호값을 view1을 통해 대조하고, 계정이 있을 경우 멤버십 적립이나 포인트 조회를 하고, 계정이 없을 경우 회원가입을 할 수 있는 클래스.

handleSavepointAction : public void handleSavepointAction(ActionEvent event)
- 사용자가 휴대전화 번호를 입력하지 않고 적립 버튼을 누를 경우, 안내 메시지 "공백은 입력되지 않습니다."를 출력하고, DAO.memberFindOne을 사용하여 입력된 휴대전화번호가 기존 DB T4.view1의 column2의 값과 비교하여 있을 경우 멤버십포인트인 column3에 값을 추가하여 적립을 시행하는 이벤트 메소드.
휴대전화번호가 기존 DB에 없을 경우, "회원을 조회할 수 없습니다. 회원가입을 먼저 해주세요." 안내 메시지 출력.

handleJoinmemberAction : public void handleJoinmemberAction(ActionEvent event)
- 사용자가 휴대전화 번호를 입력하지 않고 적립 버튼을 누를 경우, 안내 메시지 "공백은 입력되지 않습니다."를 출력하고, DAO.memberFindOne을 사용하여 입력된 휴대전화번호가 기존 DB T4.view1의 column2의 값과 비교하여 없을 경우 insert문 실행한 후 "(회원이름)님, 회원가입되었습니다." 안내 메시지 출력하는 이벤트 메소드.
휴대전화번호가 기존 DB에 있을 경우 "이미 가입된 회원입니다." 안내 메시지 출력.

handleViewpointAction : public void handleViewpointAction(ActionEvent event)
- 사용자가 입력한 휴대전화 번호를 DAO.memberFindOne을 사용하여 T4.view1의 column2의 값과 비교하여 있을 경우 멤버십 포인트인 column3값을 KioskDAO.viewpoint를 사용하여 안내메세지를 출력하는 이벤트 메소드. 휴대전화번호가 기존 DB에 없을 경우, "회원을 조회할 수 없습니다." 메시지 출력.

initialize : public void initialize(URL arg0, ResourceBundle arg1)
- 컨트롤러는 Initializable 인터페이스를 반드시 상속받아야하므로 컨트롤러 class는 기본적으로 initialize 추상메소드를 오버라이딩 해야한다.

<파일명*:* CheckController >
CheckController : 멤버십 포인트 적립 여부를 확인하는 기능을 가진 클래스.

serPrimaryStage : public void setPrimaryStage(Stage primaryStage)
- 커스텀 다이얼로그인 CheckController.fxml을 띄우기 위한 사전 작업으로 Stage 세팅 메소드 직접 생성. 컨트롤러에서 다이얼로그를 실행할 때는 소유자 윈도우가 될 primaryStage가 필요.

handleSavepointAction : public void handleSavepointAction(ActionEvent event) throws Exception
- anchorPane= (AnchorPane)FXMLLoader.load(getClass().getResource("Membership.fxml")) 코드를 이용하여 멤버십포인트 적립 및 가입창인 Membership.fxml을 실행하는 이벤트 메소드

handleExitAction : public void handleExitAction(ActionEvent event)
- 프로그램을 종료하는 이벤트 메소드

initialize : public void initialize(URL arg0, ResourceBundle arg1)
- 컨트롤러는 Initializable 인터페이스를 반드시 상속받아야하므로 컨트롤러 class는 기본적으로 initialize 추상메서드를 오버라이딩 해야한다. initialize 추상메소드를 오버라이딩하는 메소드.

<FXML & CSS>

FXML : 자바에서 사용할 수 있는 HTML로 씬빌더를 통해서 디자인을 작업할 수 있음

Root.fxml : 기본 창으로, 메뉴 수량 선택, 주문내역, 주문하기, 종료하기 버튼을 구현한 화면파일

Membership.fxml : 포인트 내역 확인 창으로 이름과 휴대전화번호 입력창 및 이를 통한 멤버십포인트 적립과 조회하기 버튼을 구현한 화면파일

Check.fxml : 포인트 적립 여부를 묻는 창으로 적립하기 버튼과 종료하기 버튼을 구현한 화면파일

app.css : fxml을 꾸며주는 스타일 시트 언어파일

[package] util

DBConnection : 자바에서 데이터베이스에 연결하는 클래스.

Reader reader = new FileReader("lib/oracle.properties") 코드을 통해 DB와의 연결을 위해 필요한 driver, URL, username, password 정보가 담긴 properties 파일을 읽어오고 Class.forName코드를 통해 드라이버를 클래스를 불러와 자바에서 데이터베이스에 연결하는 기능 수행.

getConnection : public static Connection getConnection()

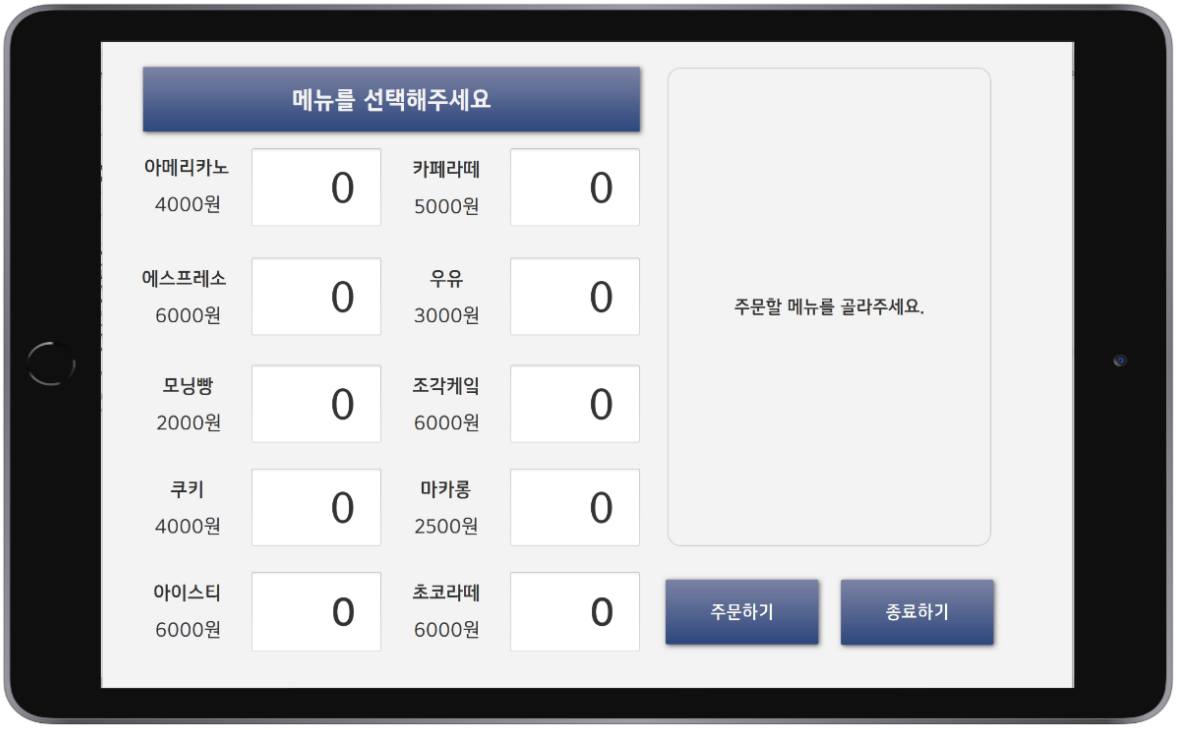
- DBConnection으로 생성되어 필드값 Connection conn에 대입된 Connection 객체를 반환. DAO에서 DBConnection 이용 시 따로 작업할 필요 없이 getConnection() 메소드로 conn 객체만 받아와 DB에 연결 가능함.

oracle.properties : DBConnection에 사용되는 driver, URL, username, password 정보가 담긴 properties 파일.

| 퀵오스크 실행 화면

- 초기화면
- 주문화면
- 포인트 적립화면
- 포인트 조회 화면
- 회원 관련 화면

퀵오스크 초기 화면



퀵오스크 주문화면

Action point1. 사용자가 메뉴에 주문 개수를 입력 후 주문하기를 눌렀을 때

Result. 3번화면에서 총 금액 확인가능

Action point2. 주문을 하지 않고 주문하기 버튼을 눌렀을 때

Result. 3번 화면에서 “주문할 메뉴를 골라주세요” 메시지 노출

Action point3. 가지고 있는 재고 이상의 주문 건수를 입력했을 때

Result. 2번화면에서 관련 메뉴의 재고 수량이 부족하다는 메세지 노출

Action point1. 사용자가 메뉴에 주문 개수를 입력후 주문하기를 눌렀을 때

Result. 3번화면에서 총 금액 확인가능

Action point2. 주문을 하지 않고 주문하기 버튼을 눌렀을 때

Result. 3번 화면에서 “주문할 메뉴를 골라주세요” 메시지 노출

Action point3. 가지고 있는 재고 이상의 주문 건수를 입력했을 때

Result. 2번화면에서 관련 메뉴의 재고 수량이 부족하다는 메세지 노출

퀵오스크 포인트 적립화면

Action point. 적립하기 버튼 선택

Result. 2번화면에서 “포인트 적립되었습니다” 메세지 노출

Action point. 적립하기 완료 후 다시 적립하기 버튼을 눌렀을 때

Result. 2번화면에서 “주문을 먼저 해주세요” 메세지 노출

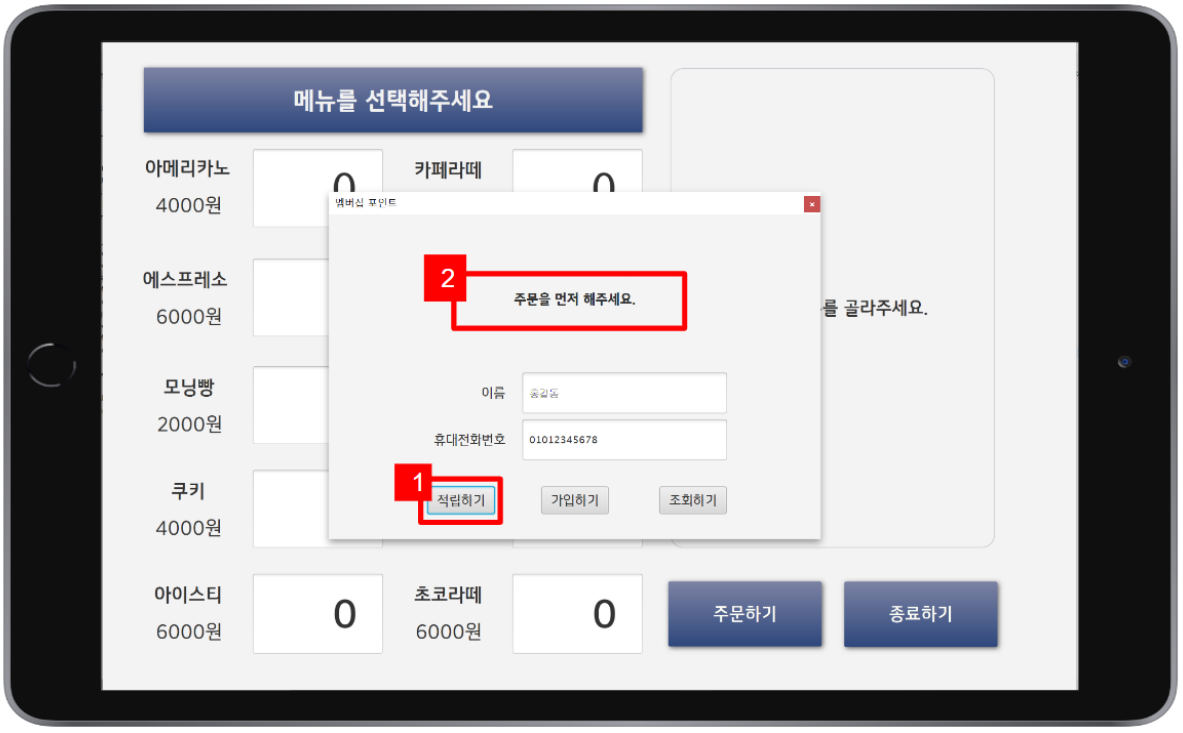
퀵오스크 포인트 적립화면

Action point. 주문하기 버튼 클릭 후 포인트 팝업에서 적립하기 버튼 눌렀을 때

Result. 2번화면에서 “포인트 적립되었습니다” 메세지 노출

Action point. 적립하기 완료 후 다시 적립하기 버튼을 눌렀을 때

Result. 2번화면에서 “주문을 먼저 해주세요” 메세지 노출



퀵오스크 회원 관련 화면 상세

회원가입

Action point1. 포인트 적립화면에서 회원정보 입력 후 가입하기 버튼 클릭

Result. 회원 정보가 DB에 없으면 가입 가능

Result. 회원 정보가 DB에 있으면 가입 불가능

Action point2. 회원 정보 아무것도 입력하지 않고 가입하기 눌렀을 때

Result. 오류 발생

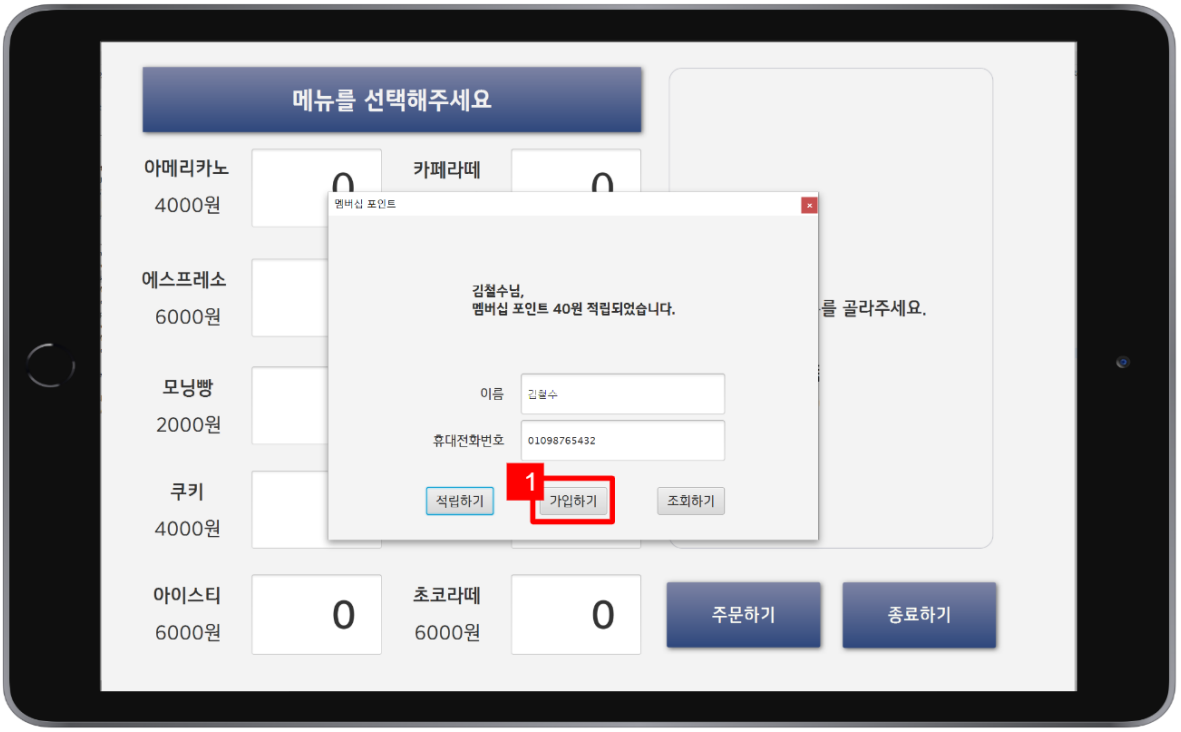
회원조회

Action point1. 회원 정보 입력 후 조회하기 버튼 클릭

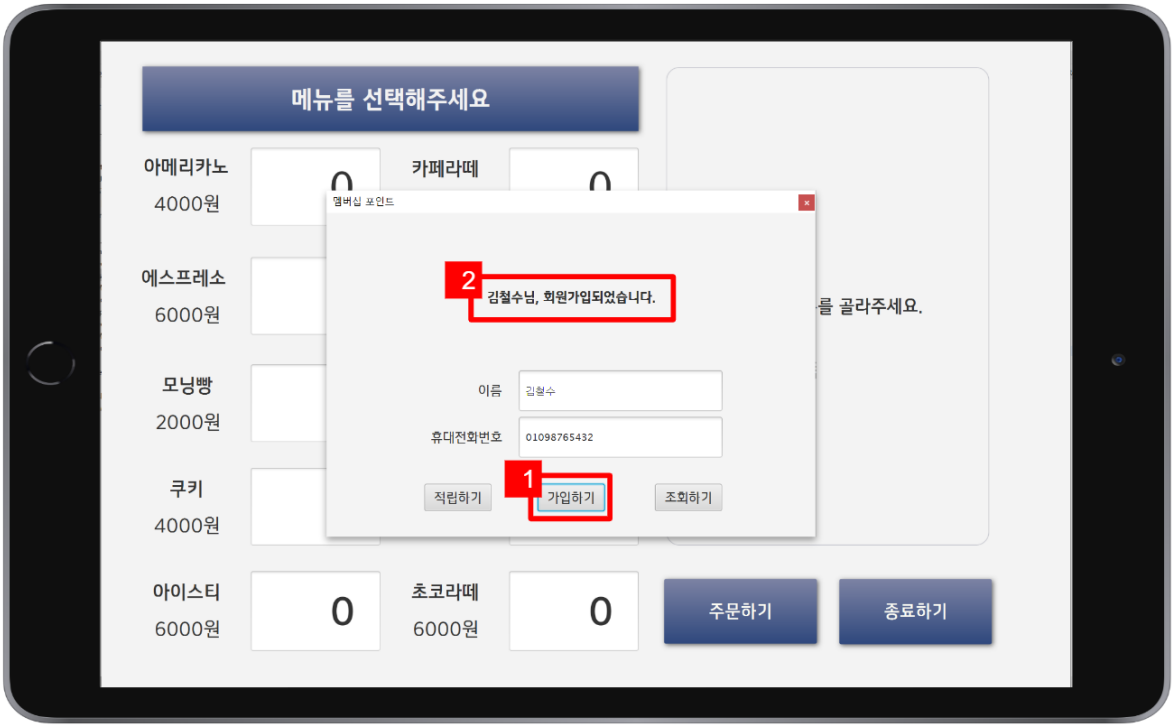
Result. 회원 정보가 DB에 없으면 조회 불가능

Result. 회원 정보가 DB에 있으면 적립 포인트 내역 조회

Action point1. 포인트 적립화면에서 회원정보 입력 후 가입하기 버튼 클릭



Result. 회원 정보가 DB에 없으면 가입 가능

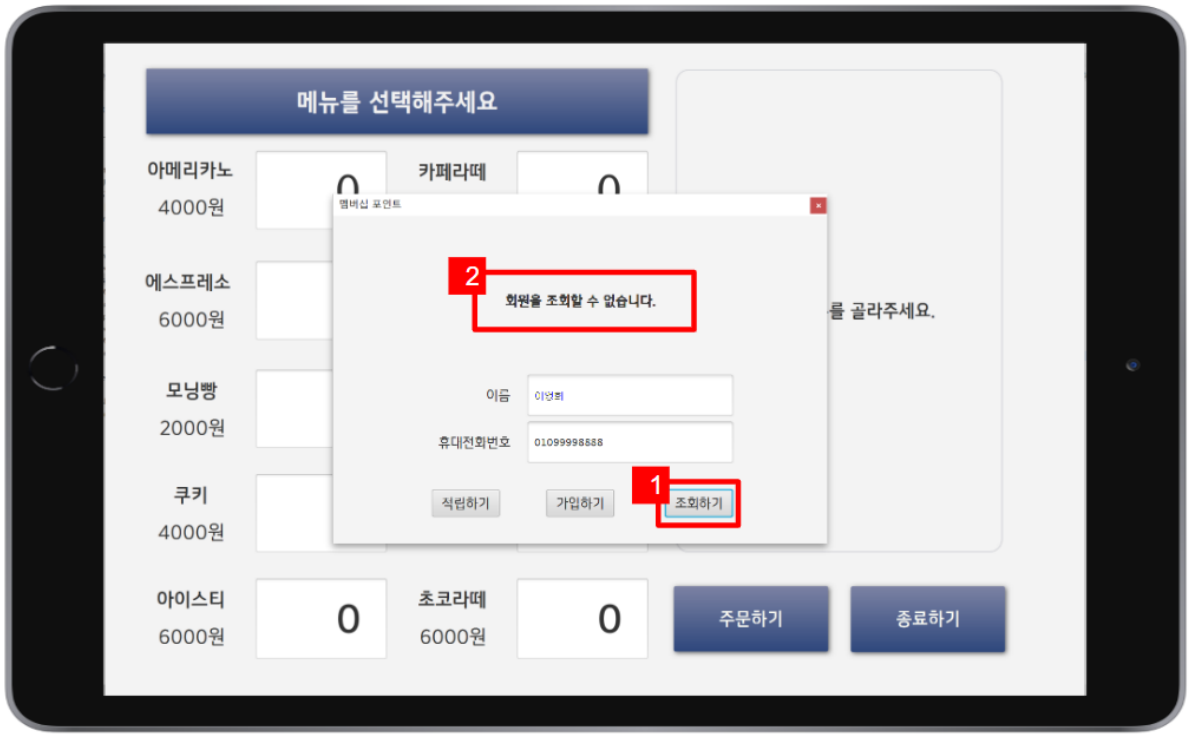


Result. 회원 정보가 DB에 있으면 가입 불가능

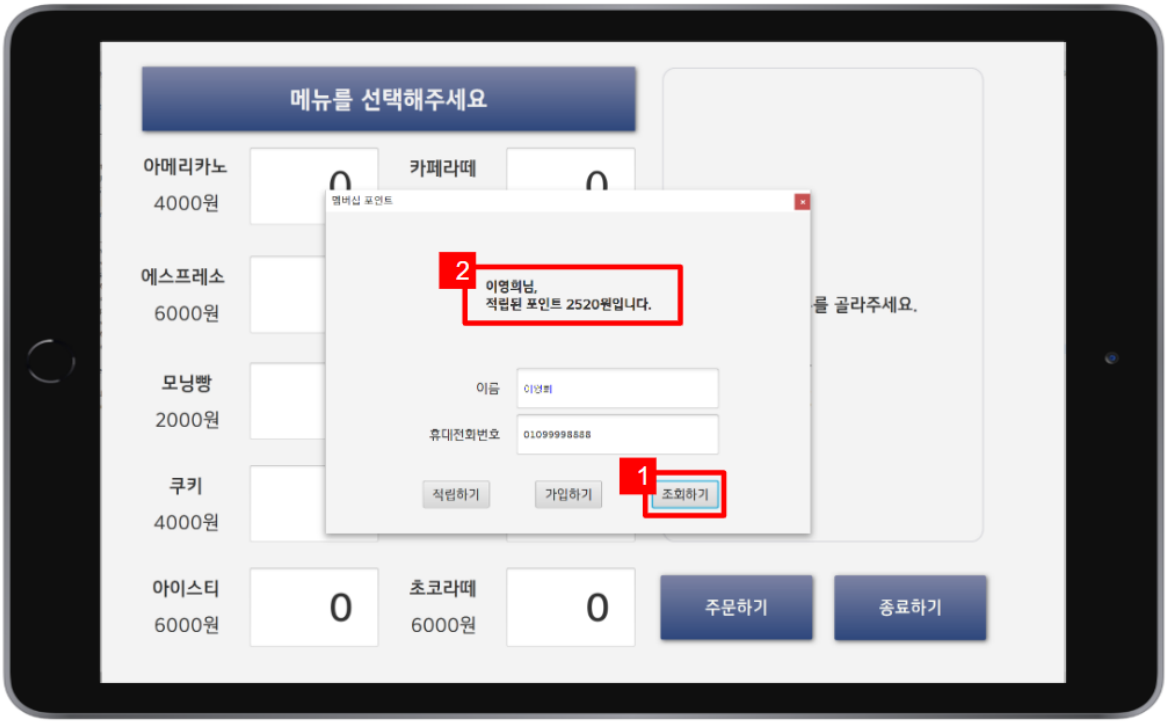
Result. 회원 정보 아무것도 입력하지 않고 가입하기 눌렀을 때 오류 발생

Action point1. 회원 정보 입력 후 조회하기 버튼 클릭

Result. 회원 정보가 DB에 없으면 조회 불가능



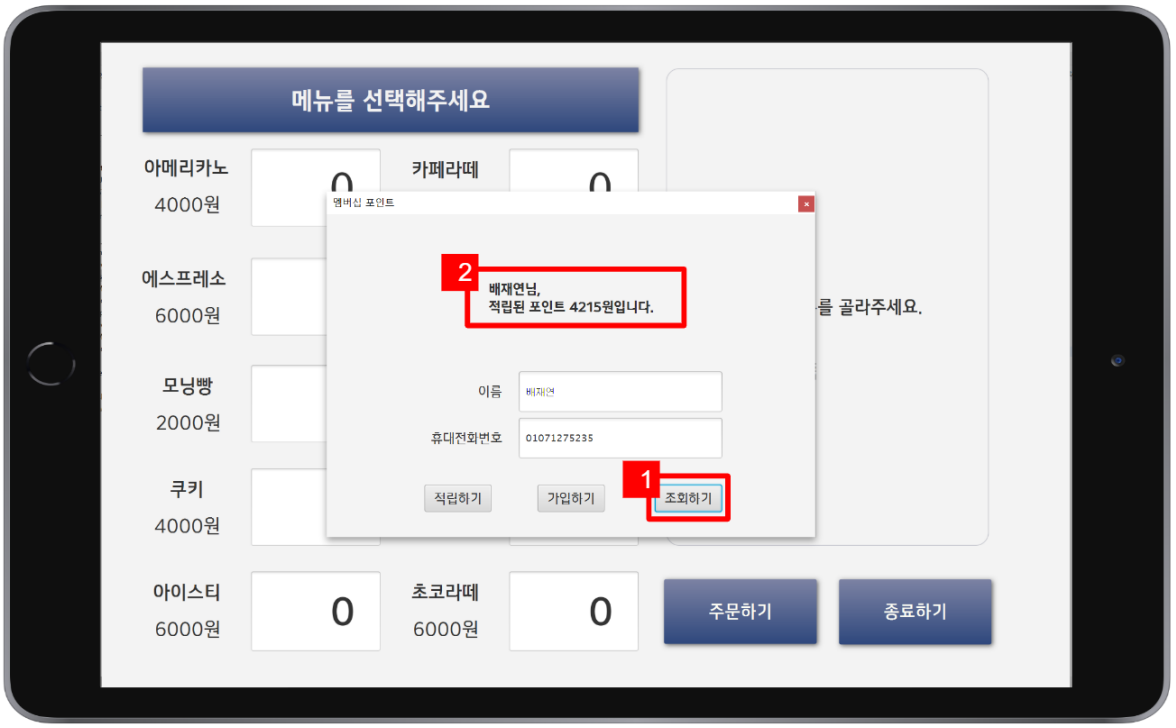
Result. 회원 정보가 DB에 있으면 적립 포인트 내역 조회



퀵오스크 포인트 조회

Action point1. 멤버십 포인트 화면에서 회원정보 입력 후 조회하기 버튼을 눌렀을 때

Result. 2번화면에서 포인트 적립내역 조회 가능



| 기대효과 및 활용방안

현재 키오스크의 도입은 급성장하는 추세지만, 인터페이스에 있어 전연령층에서 불편함을 호소하는 사례 또한 늘고 있다. 본 프로젝트는 보다 직관적인 UI로 키오스크가 낯선 이들도 손쉽게 사용할수 있기에 바쁜 현대사회에서 주문대기줄이 늘어나는 것을 미연에 방지하고, 매장 회전율을 높여 판매자 입장에서는 같은 시간 내 매출 증가 효과를, 손님 입장에서는 구매시간 단축 효과를 볼 수 있다.

| 개발 일정

구분	22' 11월							
	16일	17일	18일	21일	22일	23일	24일	25일
분석/설계								
-요구사항 재분석	■							
-전체 시스템 기능설계		■						
개발								
-DB 설계			■	■				
-메인 시스템 개발				■	■			
-회원가입 시스템 개발				■	■			
-포인트 적립 시스템 개발				■	■			
디자인								
-화면 설계					■			
테스트 & 검수								
-통합테스트 & 디버깅						■	■	■

| 역할 및 담당 업무

페어 프로그래밍으로 프로젝트 진행

배재연

Team Leader

SW설계

DB모델링

김미리

UI설계&디자인

DB모델링

프로젝트 보고서

김학수

SW설계

ppt 제작

윤희진

UI설계&디자인

ppt 제작

| 아쉬운 점

- 프로젝트 기간이 다소 짧아 구현하지 못한 기능이 발생하였다.
- 해당 프로젝트를 구현할 때 필요한 언어나 라이브러리에 대해 학습하는 시간이 많이 필요했다.
- 판매자 입장에서 매출내역이나 회원정보를 일괄적으로 관리하는 프로그램도 있었으면 좋았을텐데 시간이 부족했다.