**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *EZOrder* |
| 팀 명 | *EZOrder - 32* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 4 |
| **Date** | 20 |

|  |  |
| --- | --- |
| **팀원** | 박민서 (팀장) |
| 문원기 |
|  |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “EZOrder”를 수행하는 팀 “32팀”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “32팀”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 수행결과보고서-EZOrder.doc |
| **원안작성자** | 박민서, 문원기 |
| **수정작업자** | 박민서, 문원기 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2024-05-19 | 박민서 | 1.0 | 최초작성 | 추진 배경 및 필요성  개발 목표  서비스 대상 분석 일부  연구/개발 내용 및 결과물  시스템 기능/비기능 요구사항  Thymeleaf의 적용 |
| 2024-05-19 | 문원기 | 1.1 | 내용 추가 | 프로젝트 개요  서비스 대상 분석 일부  시스템 아키텍쳐 설계  MVC 패턴 적용  웹소켓 기술의 활용  인증번호 시스템의 도입 |
| 2024-05-19 | 박민서 | 1.1 | 내용 추가 | 결과물 목록  기대 효과 및 활용 방안 |
| 2024-05-19 | 문원기 | 1.2 | 내용 추가 | 활용/개발된 기술 표 추가  참고문헌  자기평가 |
| 2024-05-20 | 박민서 | 1.3 | 내용 수정 | 연구/개발 내용 및 결과물  시스템 기능/비기능적 요구사항 |
| 2024-05-20 | 문원기 | 1.4 | 내용 추가 | 부록 작성 |
|  |  |  |  |  |

**목 차**

[1 개요 5](#_Toc167381073)

[1.1 프로젝트 개요 5](#_Toc167381074)

[1.2 추진 배경 및 필요성 6](#_Toc167381075)

[1.3 유사 서비스 분석 7](#_Toc167381076)

[1.4 서비스 대상 유저 분석 8](#_Toc167381077)

[2 개발 내용 및 결과물 10](#_Toc167381078)

[2.1 목표 10](#_Toc167381079)

[2.2 연구/개발 내용 및 결과물 10](#_Toc167381080)

[2.2.1 연구/개발 내용 10](#_Toc167381081)

[2.2.2 시스템 기능 요구사항 27](#_Toc167381082)

[2.2.3 시스템 비기능(품질) 요구사항 29](#_Toc167381083)

[2.2.4 시스템 구조 및 설계도 30](#_Toc167381084)

[2.2.5 활용/개발된 기술 31](#_Toc167381085)

[2.2.6 현실적 제한 요소 및 그 해결 방안 36](#_Toc167381086)

[2.2.7 결과물 목록 37](#_Toc167381087)

[2.3 기대효과 및 활용방안 37](#_Toc167381088)

[2.3.1 기대효과 37](#_Toc167381089)

[2.3.2 활용방안 37](#_Toc167381090)

[3 자기평가 38](#_Toc167381091)

[4 참고 문헌 39](#_Toc167381092)

[5 부록 40](#_Toc167381093)

[5.1 사용자 매뉴얼 40](#_Toc167381094)

[5.1.1 점주 매뉴얼 40](#_Toc167381095)

[5.1.2 손님 매뉴얼 45](#_Toc167381096)

[5.2 배포 가이드 48](#_Toc167381097)

[5.3 테스트 케이스 50](#_Toc167381098)

# 개요

## 프로젝트 개요

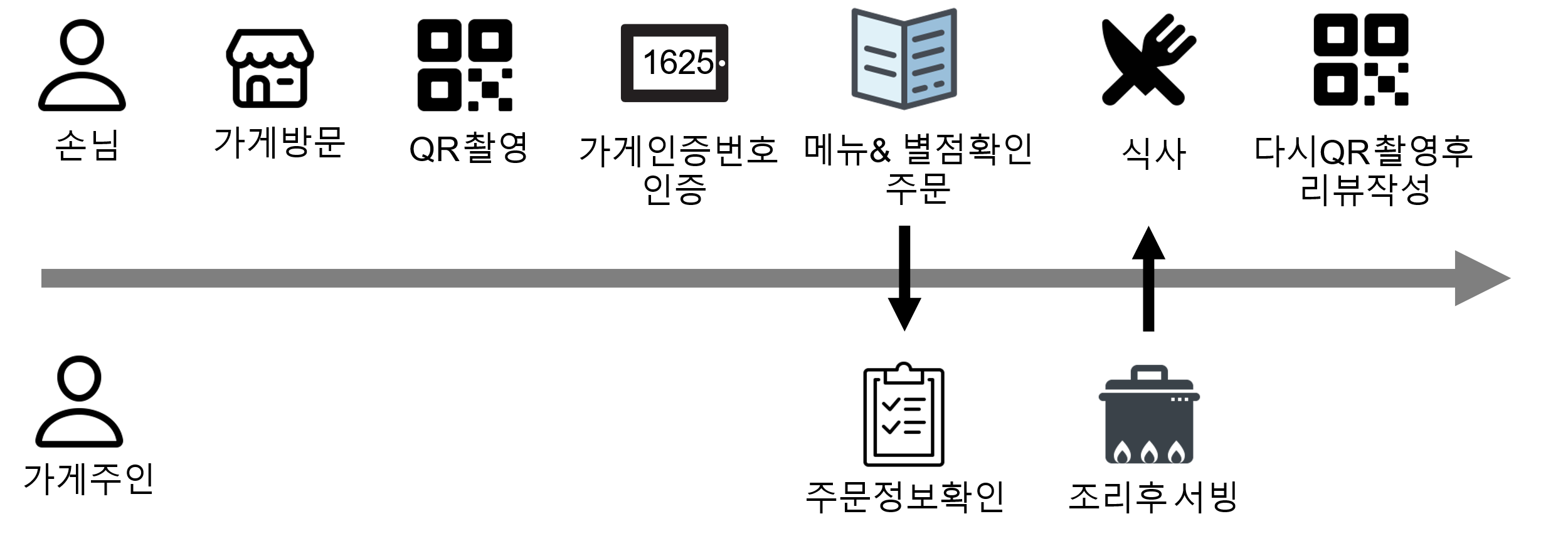
EZOrder는 음식점에서 테이블에 부착된 QR 코드를 통해 간편하게 주문을 할 수 있는 웹 서비스이다.

이 서비스의 목표는 사용자가 모바일을 통해 빠르고 편리하게 주문할 수 있는 시스템을 제공하는 것이다.

손님들은 테이블에 부착된 QR 코드와 가게에 부착된 인증 번호를 이용하여 자신의 스마트폰으로 메뉴에 바로 접근할 수 있다. 이를 위해 회원 가입이나 앱 다운로드와 같은 번거로운 절차는 필요하지 않다. 인증번호는 가게 내에 한 개만 존재하며 가게 내에 존재하는 모든 테이블에서 해당 인증번호로 주문할 수 있다.

메뉴를 확인하고 리뷰 및 별점을 살펴본 후, 손님들은 원하는 음식을 주문할 수 있다. 가게 주인은 이러한 주문을 ‘주문 대기열’ 페이지를 통해 즉각적으로 확인할 수 있다.

음식을 맛본 후에는 손님들은 자신의 경험을 리뷰로 남길 수 있다. 테이블에 부착된 QR 코드를 통해 리뷰 페이지에 접근할 수 있으며, 작성된 리뷰는 메뉴에 반영되어 다른 손님들이 확인할 수 있다.



## 추진 배경 및 필요성

2019년부터 시작된 코로나19 사태는 우리의 삶을 크게 변화시켰다. 비대면이 필수가 되는 시대로 바뀌면서 키오스크는 우리의 일상에 자리를 잡게 되었다.

자영업자들은 임대료와 최저임금 증가 등으로 어려움을 겪게 되면서, 편리하고 효율적인 수단을 찾게 되었다. 이로 인해 키오스크의 사용량은 계속해서 증가하고 있다. 이전에는 주로 패스트푸드점 등에 설치되어 있던 키오스크가 이제는 일반 음식점에서도 자주 볼 수 있게 되었다.

테이블 오더는 키오스크의 다른 형태로, 테이블에 앉아서 메뉴판 없이도 태블릿 PC를 통해 직접 주문할 수 있는 장점이 있다. 특히 매장이 넓고 테이블이 많은 업체에서는 테이블 오더를 점점 더 많이 사용하고 있다. 이는 고객 주문을 누락하지 않고 인건비를 절감하는 데에 도움이 된다.

하지만 키오스크와 테이블 오더의 기기 비용이 상당히 높은 편이다. 키오스크는 평균적으로 한 대당 200~500만원으로 형성되어 있으며, 대형 기기의 경우 1000만원을 넘기도 한다.

기기 당 300만원으로 가정할 때, 2대를 설치하면 종업원 1명의 월급 3개월치가 손익분기점이다. 테이블 오더의 경우에도 평균적으로 태블릿 한 대당 40~50만원 정도로 비용이 발생한다. 매장 내 테이블이 20개 있다면 설치 비용만 800~1000만원에 달할 것이다.

비용은 초기에만 부담할 수 있다면 괜찮을 것으로 생각할 수 있지만, 기기에 결함이 생기거나 유지 보수가 필요할 경우 비용이 더 많이 발생할 수 있다.

이러한 문제점과 경제적 부담을 고려하여 회원가입만으로도 사용 가능한 QR 스캔을 통한 주문 "웹 서비스"를 제공하고자 한다. 이를 통해 기기 설치 비용과 유지 보수 비용 등에 대한 문제를 해결하고자 한다.

## 유사 서비스 분석

비슷한 방식의 QR 주문을 제공하는 서비스들은 시중에 많이 나와있으나, QR에서 비롯된 문제점을 해결하는 방법에 있어서는 차이를 보였다.

QR 주문에서 마주할 수 있는 가장 큰 보안 문제로는 ‘QR 재접근’ 문제가 있다. 이는 악성 사용자가 특정한 가게의 QR 코드를 저장한 후에 매장 밖에서 악성 주문을 날리는 경우이다.

이렇게 되면 가게에서는 실제로는 존재하지 않는 손님을 위해 음식을 조리하며, 실제 손님의 주문이 밀리게 되는 등 운영에 큰 지장을 입게 된다. 위 예시처럼 QR 코드 기술의 취약점을 이용한 다양한 문제들이 존재하며, 시장에서는 이러한 문제를 해결하기 위해 다양한 솔루션을 도입하고 있었다.

그 중 가장 대표적인 ‘네이버 QR 주문’과 ‘핸드 오더’의 경우를 비교해보았다.

네이버 QR 주문의 경우, 네이버 계정 연동이 필수적이다. 네이버 계정이라는 단단한 보안 시스템을 통해 문제점을 해결하고, 또한 네이버 페이 시스템과 연계되어 결제까지 가능하다는 장점이 있다. 하지만 당장 네이버 페이를 사용하지 않는 사용자의 경우, 결국에는 다시 대면으로 주문을 진행해야 한다는 단점이 있다.

‘핸드 오더’의 경우, 위변조와 복사가 불가능한 디지털 QR 장비를 도입하여 이를 해결하였다. 다만 이는 QR 코드를 표시하기 위한 특수 디지털 장비가 필요하다는 단점이 존재한다.

텍스트, 만화 영화이(가) 표시된 사진

자동 생성된 설명

## 서비스 대상 유저 분석

'EZOrder의 서비스 대상은 가게를 운영하는 점주와 해당 가게를 이용하는 손님으로 나뉜다.

먼저 손님 측의 입장에서 이러한 디지털 주문 서비스에 어느 정도의 익숙함을 가지고 있는지가 매우 중요하기에, 이에 관하여 분석을 해보았다.'

EZOrder 서비스는 모바일을 통한 QR 주문을 핵심 기능으로 삼고 있다. 따라서 음식점 손님, 소비자들의 QR에 대한 인식과 현 시점의 스마트폰 보급률이 상당히 중요한 요소로 작용할 것으로 예측된다.

코로나 시대 때, 타인과의 접촉을 최소화하는 언택트 활동의 비율이 높아지면서 많은 사람들이 QR과 같은 디지털 정보 전달 매체에 대해 긍정적으로 인식하고 있는 것으로 보인다. 가까운 예시로 질병관리청에서 진행한 코로나19 예방접종 증명 방법 안내에서도 QR 코드를 통한 인증을 내세우고 있다.

코로나 시대 때 이루어진 경제 활동은 디지털 기기를 활용한 비대면 소비가 주를 이루었는데, 이 당시 서울연구원에서 2020년에 발표한 ‘‘2/4분기 서울시 소비자 체감경기와 비대면 경제’ 리포트에 따르면, 당시 이루어진 비대면 소비 경험 분야 중 54%가 음식 관련 분야인 것으로 드러났다.

EZOrder 서비스 역시 디지털 기기를 활용한 음식 주문 서비스로서, 코로나 시절에 디지털 기기를 활용한 음식 소비 활동에 주체적으로 참여한 소비자들이 거리감 없이 접근할 수 있는 환경을 제공할 것으로 기대된다.

또한 한국 갤럽이 발표한 ‘2012-2023 스마트폰 사용률 & 브랜드, 스마트워치, 무선이어폰에대한 조사’에 따르면 나이에 관계없이 많은 사람들이 스마트폰을 가지고 있는 것으로 나타났다. 해당 조사에 따르면 60대 노인층의 경우 98%, 70대 이상도 85%의 비율로 스마트폰을 사용하고 있다고 한다. 따라서 EZOrder 서비스는 넓은 연령층에게 접근 가능한 플랫폼을 제공할 수 있을 것으로 기대된다.

다음으로 ‘EZOrder’ 서비스의 도입을 결정하는 주체인 음식점을 운영하는 점주들을 기준으로 분석해보았다.

한국농촌경제연구원이 외식업체를 대상으로 실시한 ‘푸드테크 사용의향 조사’ 결과에 따르면 ‘인건비 절감(41.2%)’, ‘매장 운영 효율화(’29.4%)’ 등이 무인주문기 사용의향 이유로 선정됐다. 최저임금의 상승으로 인해 인건비 부담이 커지고 있고, 고된 일을 꺼리는 추세이다 보니 노동 시장에서 일할 인력을 구하기 어렵다고 한다. 점주들 입장에서는 인력 관리가 항상 고민 거리이기 때문에 인력을 줄이는 방법을 찾을 수밖에 없다고 한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

하지만 앞서 프로젝트 기획 배경에서 언급하였듯이 무인주문기를 도입하는 것은 기기 설치비, 유지 보수비, 여기에 추가적으로 서비스 이용료 및 수수료 등의 경제적 부담이 따르게 된다. 인건비를 줄이기 위해 이러한 무인주문기 서비스를 사용하려던 점주들의 우려가 생겨나고 있다.

‘EZOrder’는 QR 스캔을 통한 주문 방식과 회원가입 후 로그인만 하면 웹에서 바로 사용가능하다는 점을 통해 서비스 이용에 대한 점주들의 이러한 경제적인 부담을 줄이고자 한다.

# 개발 내용 및 결과물

## 목표

점주 입장에서 운영 비용 절감과 안전한 주문 처리를 누릴 수 있는 시스템을 개발한다.

손님 입장에서는 로그인이나 앱 설치가 필요 없는 QR 주문 서비스를 제공함으로써 더욱 편리하고 빠르게 주문할 수 있는 환경을 만든다.

## 연구/개발 내용 및 결과물

### 연구/개발 내용

1. 관리자 페이지

페이지의 동작을 서술하는 ‘화면’과 그리고 서버의 요청 처리 과정을 서술하는 ‘요청 처리’ 파트로 나누어 설명한다.

* 1. 로그인

텍스트, 스크린샷, 만화 영화, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 운영 체제, 로고이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

관리자가 로그인하기 위해 접속하는 최초의 화면이다. 아이디와 비밀번호를 입력하고 로그인 버튼을 클릭하면 해당 요청을 서버로 보내게 된다. 로그인에 성공할 경우, 실시간 테이블 현황 페이지(홈화면으로도 불린다)로 이동한다. 아이디 또는 비밀번호 오류로 인해 로그인에 실패했을 경우 로그인에 실패했음을 로그인 버튼 위에 표시한다.

* **요청 처리**

클라이언트에게서 로그인 요청이 들어왔을 때, DB에 존재하는 아이디인지를 우선적으로 판단한다. 아이디가 존재하고, 암호화된 비밀번호가 일치하면 로그인 성공 처리를 하여 클라이언트가 홈화면으로 리다이렉트하도록 했다. 아이디가 존재하지 않거나 비밀번호가 일치하지 않으면 오류 관련 parameter를 url에 추가하여 클라이언트가 해당 url로 리다이렉트하도록 했다.

* 1. 회원가입

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

회원가입 시에 입력해야 할 사항은 아이디, 비밀번호, 가게 이름이다. 다음 버튼을 클릭하면 전송하기 전에 비밀번호와 비밀번호 확인란이 일치하는지 js를 통해 검사하는데, 일치하지 않으면 css 애니메이션을 통해 일치하지 않음을 사용자에게 보여준다. 해당 검사를 통과하면 서버에 해당 내용들을 전송한다. 아이디가 중복이어서 회원가입에 실패하는 경우 그에 해당하는 메시지가 상단에 표시된다.

* **요청 처리**

클라이언트에게서 회원가입 요청이 들어오는 경우, 아이디 중복검사를 진행한다. 이미 존재하는 아이디면, 오류 관련 parameter를 url에 추가하여 클라이언트가 해당 url로 리다이렉트하도록 했다. 회원가입에 성공하면, 로그인 화면으로 리다이렉트하도록 했다.

* 1. 마이페이지

텍스트, 스크린샷, 소프트웨어, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

마이페이지에서 수정할 수 있는 것은 가게 이름, 가게 주소, 비밀번호, 가게 설명, 가게 대표 로고다. 기존의 프로필 내용이 반영된 폼이 사용자에게 제공된다. 수정이 발생하는 경우 기존 비밀번호는 필수로 입력해야 한다. 비밀번호가 일치하지 않으면 비밀번호가 틀렸다는 메시지를 표시한다. 비밀번호를 변경하는 경우에도 일치하지 않거나, 비밀번호의 길이가 기준에 맞지 않으면 해당 오류 메시지를 표시한다.

* **요청 처리**

클라이언트에게서 프로필 수정 요청이 들어오는 경우, 비밀번호가 일치하는지 확인한다. 비밀번호가 일치하면 수정된 폼의 내용에 따라 수정을 반영한다. 가게 대표 로고 사진의 경우, S3에 업로드하는 과정을 거치고 DB에는 사진이 아니라 실제로는 사진의 URL을 저장한다. 사진이 업로드되지 않은 상태로 요청이 들어오면 별다른 작업을 하지 않고, 해당 메뉴의 메뉴 사진 URL을 빈 값으로 DB에 저장한다. 반영 여부를 확인할 수 있도록 기존 url로 다시 리다이렉트(새로고침의 개념)되도록 한다. 비밀번호가 일치하지 않으면 오류 관련 parameter를 url에 추가하여 해당 url로 리다이렉트되도록 한다.

* 1. 메뉴 관리

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

네비게이션 바의 메뉴 관리 탭을 클릭하면 위 화면으로 이동하게 된다. 메뉴 관리 페이지에서는 카테고리별로 메뉴를 관리할 수 있다. 화면 상단에서는 별점이 가장 높은 메뉴, 가장 주문이 많은 메뉴를 확인할 수 있다. 새로운 카테고리를 추가, 삭제할 수 있으며, **새로운 메뉴를 추가, 수정, 삭제**할 수도 있다. 그 밑에서는 각 메뉴의 이름, 가격, 손님들의 평균 별점을 확인할 수 있다. **돋보기 버튼**을 통해 메뉴의 상세페이지에 들어가 메뉴의 상세 내용 및 손님이 남긴 리뷰들을 확인할 수도 있다.

* **요청 처리**

메뉴 관리 페이지로 이동하게 되면 서버는 요청을 보낸 사용자의 카테고리 리스트와 그에 해당되는 메뉴 리스트를 클라이언트에게 반환해준다. 메뉴 리스트에는 메뉴의 이름, 사진, 가격, 별점 내용이 포함되어 있다. 클라이언트의 브라우저는 이 정보를 가공하여 사용자에게 보여준다.

* 1. 메뉴 추가/수정

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

메뉴를 생성하는 경우에 카테고리 선택한 뒤, 메뉴 이름, 가격, 설명을 입력하고, 메뉴 사진을 업로드해야 한다. 메뉴 사진은 선택사항이지만, 카테고리, 메뉴 이름, 가격, 설명은 필수 항목으로 제한을 걸었다.

메뉴를 수정하는 경우에는 해당 메뉴의 정보들이 이미 폼에 반영되어 있다. 사진 또한 미리보기로 확인 가능하다. 원하는 부분만 수정하여 제출할 수 있다.

* **요청 처리**

클라이언트로부터 메뉴 생성/수정 요청이 들어오면 서버에서는 폼 내용을 기준으로 카테고리, 메뉴 이름, 가격, 설명, 사진을 DB에 반영을 한다. 메뉴 사진의 경우, S3에 업로드하는 과정을 거치고 DB에는 사진이 아니라 실제로는 사진의 URL을 저장한다. 사진이 업로드되지 않은 상태로 요청이 들어오면 별다른 작업을 하지 않고, 해당 메뉴의 메뉴 사진 URL을 빈 값으로 DB에 저장한다.

* 1. 메뉴 상세 관리

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

메뉴 관리 페이지에서 각 메뉴의 돋보기 버튼을 누르면 이동하는 페이지다.

* **화면(View)**

서버에서 보내준 메뉴 정보를 토대로 메뉴 사진, 메뉴 이름, 메뉴 설명, 가격, 평균 별점, 주문횟수를 보여준다. 또한, 해당 메뉴에 사람들이 작성한 리뷰들과 별점을 확인할 수 있다. 해당 메뉴와 관련된 리뷰들이 10개씩 제공되며, 페이지네이션 바(화살표 모양)가 리뷰 목록 아래에 위치하여, 전체 리뷰 목록을 순회하며 조회할 수 있다. **메뉴 삭제**와 **메뉴 수정** 버튼 또한 존재하여 해당 페이지에서 메뉴 관리 또한 가능하다.

* **요청 처리**

클라이언트로부터 GET 요청이 들어오면 서버에서 메뉴 정보를 반환해준다. 메뉴와 연결되어 있는 리뷰 정보 또한 같이 반환해준다. 이때, URL에 있는 page 파라미터 정보를 통해 리뷰 목록의 페이지를 파악하고 해당하는 리뷰 정보를 반환해준다.

* 1. 테이블 현황

텍스트, 스크린샷, 소프트웨어, 도표이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

전체적인 매장 내 테이블 현황을 확인 할 수 있다. 가장 손님이 많이 앉았던 인기 있는 테이블과 인기 없는 테이블을 확인하여 매장 테이블 배치에 도움이 될 수 있다. 추가하고 싶은 테이블 이름을 입력하여 추가할 수 있으며, 테이블 이름을 수정하거나 테이블을 삭제할 수도 있다. 테이블에서 주문한 내역을 확인할 수 있으며, 테이블을 비우거나 결제하는 단계로 넘어갈 수도 있다.

**새로고침** 버튼을 누르면, 테이블 현황을 최신화하여 확인할 수 있다.

**결제하기** 버튼을 누르면 해당 테이블의 ‘**테이블 결제**’ 페이지로 넘어간다.

**비우기** 버튼을 누르면 해당 테이블에 올려져 있는 주문 내역이 전부 취소된다.

* **요청 처리**

사용자가 추가한 테이블 리스트와 해당 테이블과 연결되어 있는 주문 내역 리스트를 반환해준다.

화면 상단에 위치한 테이블에 대한 통계값 관련 정보도 같이 반환해준다. 테이블 이름 수정 요청이 오면 이를 반영하여 다시 반환을 해준다. 테이블 삭제 요청이 들어오면 테이블을 삭제 처리한 후 리다이렉션을 반환해준다.

* 1. 테이블 결제

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

각 테이블에서 결제하기 버튼을 눌렀을 때 넘어오는 화면이다. 해당 테이블에서 주문한 내역과 총 가격을 확인할 수 있다. 상세보기를 클릭 시 JSON 형태의 주문 내역 정보를 출력하여 준다. 결제하기 버튼을 누를 시, 서버로 결제처리 요청을 하며, 다시 테이블 현황 페이지로 넘어가게 된다.

* **요청 처리**

클라이언트에서 요청한 테이블을 기준으로 그 테이블의 주문내역을 반환한다. 결제하기 버튼을 통해 결제처리 요청이 들어오면 DB 상으로 해당 주문이 BILLED 처리가 된다. 상세보기 클릭시 해당 테이블의 주문 내역 정보를 Json 형태로 리턴한다.

기존 계획은 결제 처리시 타사의 결제 시스템을 연동을 통해 실제로 결제가 진행되거나, 또는 영수증을 출력하는 것이었다.

그러나 구현 시간이 부족하고 법률적 지식이 동반되어야 하는 부분임을 알게 되어, 부득이하게 Json 형태의 주문 내역 정보를 출력하도록 변경하였다.

이 결과가 해당 페이지의 ‘상세보기’ 버튼이다.

추후에 해당 제약 사항이 해결되어 구현이 본격화 될 때, 확장이 용이하도록 하기 위해 해당 부분을 삭제하지 않고 남겨두었다.

* 1. 주문 현황

텍스트, 스크린샷, 번호, 도표이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

전체적인 매장 내 주문 현황을 확인 할 수 있다. 웹소켓을 통해 실시간으로 손님이 주문한 주문내역이 화면에 업데이트되고 Queue의 형태로 대기열을 확인할 수 있다. 취소, 접수, 완료 상태를 선택할 수 있으며, 이 상태는 서버에 전달되어 DB에 반영된다.

* **요청 처리**

웹소켓을 통해 손님이 주문할 때마다 업데이트된 주문 리스트를 반환해준다. 클라이언트에서 주문 상태의 업데이트 요청이 들어오면 해당 요청을 검증하여 DB에 반영한다. 주문 대기열에 변경사항이 생길 경우 웹소켓을 통해 현재 연결된 Socket Connection 정보를 기반으로 해당 주문 대기열을 봐야 하는 클라이언트를 식별하여 주문 리스트를 전송한다.

* 1. 인증번호

폰트, 텍스트, 그래픽, 로고이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

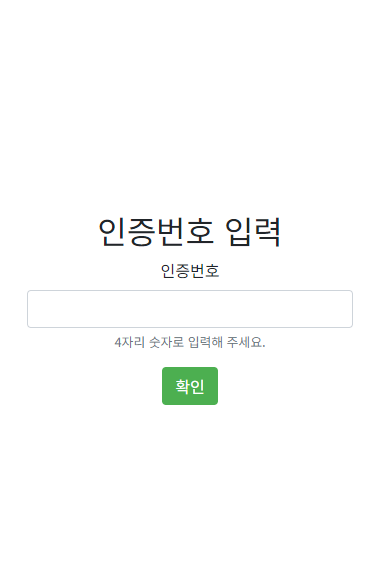
현 매장의 주문 인증번호를 확인할 수 있다. 웹소켓을 통해 연결되어서 인증번호가 변경될 시, 변경된 인증번호가 즉시 반영되어 화면에 출력된다.

**매장 내의 모든 손님들은 이 페이지에 표시된 인증번호를 사용하여 주문 페이지에 접근할 수 있다.**

* **요청 처리**

현 매장의 주문 인증번호를 반환한다. 인증번호는 30분마다 랜덤으로 변경되며, brute-force 공격을 방지하기 위해 한번 틀릴 경우에도 즉시 변경된다. 이때 웹소켓을 통해 연결된 클라이언트 html에 실시간으로 인증번호가 반영된다.

1. 손님용 페이지
   1. 초기화면



* **화면(View)**

**테이블에 있는 QR를 통해 접근할 수 있는 페이지**이다.

주문하기 버튼과 리뷰 작성 버튼이 존재한다. 주문하기 버튼을 누르면 인증번호를 입력하는 화면으로 이동한다.

리뷰 작성 버튼을 누르면 ‘리뷰 작성’ 페이지로 이동한다.

* **요청 처리**

클라이언트에서 인증번호 입력이 들어오면 인증번호 일치 여부를 판단하고, 일치하면 주문페이지로 리다이렉트시킨다. 이때, 쿠키에 인증 정보를 삽입하여 클라이언트에게 보낸다. 이 쿠키에 담긴 인증 정보가 주문 페이지에서 검증을 위해 사용된다. 인증번호가 틀려서 인증에 실패한다면 해당 매장의 인증번호를 랜덤한 난수로 변경한다. 이는 Brute Force 공격을 막기 위한 조치이다.

* 1. 주문하기

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

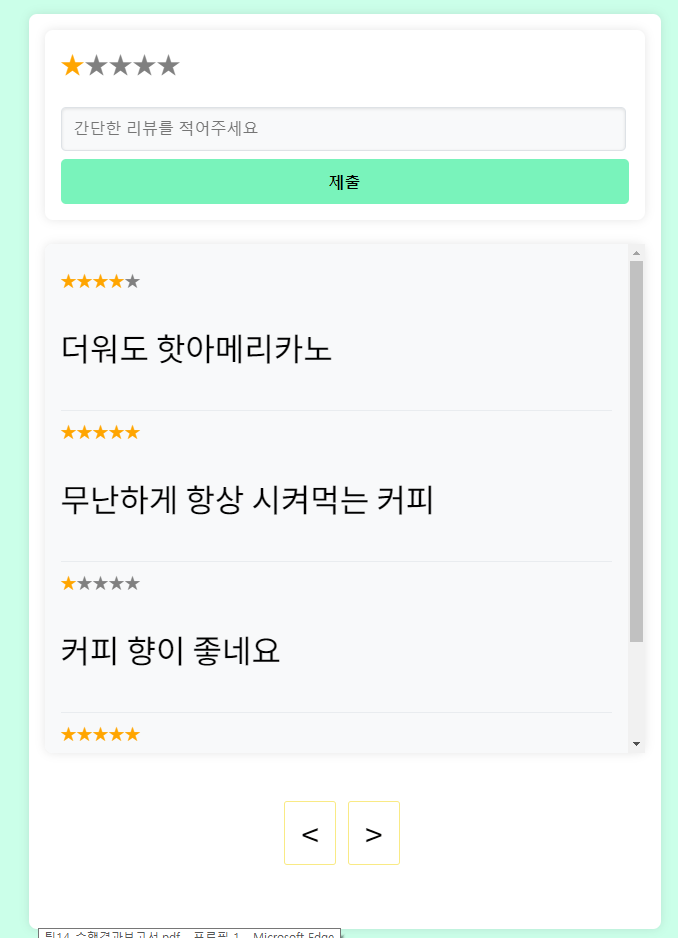
매장의 가게 로고 사진을 볼 수 있으며, 매장의 메뉴가 카테고리 기준으로 정렬되어 있다. 서버에서 반환 받은 가게 프로필과 카테고리, 메뉴 리스트가 통해 렌더링되어 보이게 된다. “담기” 버튼을 누르면 장바구니 탭이 열리며 장바구니에 해당 메뉴가 담겼음을 확인할 수 있다. 장바구니에서는 메뉴 이름, 주문할 메뉴 수량, 해당 메뉴의 금액을 확인할 수 있다. 주문 수량을 선택할 수 있으며, X 버튼을 통해 지울 수도 있다. 장바구니 버튼을 누르면 장바구니를 열고 닫을 수 있다. 최종적으로 주문하기 버튼을 누르면 서버로 주문 요청이 JSON의 형태로 넘어간다.

* **요청 처리**

해당 페이지에 접근시, 쿠키에 적힌 인증정보를 비교하여 검증한 뒤, 검증에 통과한다면 가게 프로필, 카테고리 리스트, 메뉴 리스트를 클라이언트에게 반환한다. 주문 요청이 들어오면 JSON 형태로 받은 주문 내역을 가공하여 어떤 테이블에서 어떤 메뉴들을 몇 개씩 주문했는지 DB에 저장한 후, 클라이언트에게 주문 내역 정보를 담은 쿠키를 전송한다. 이는 클라이언트가 리뷰를 작성하고자 할 때 활용된다. 그 후에는 웹소켓을 통해 주문 대기열 페이지의 주문 내역들을 실시간으로 업데이트한다.

* 1. 리뷰 작성

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

* **화면(View)**

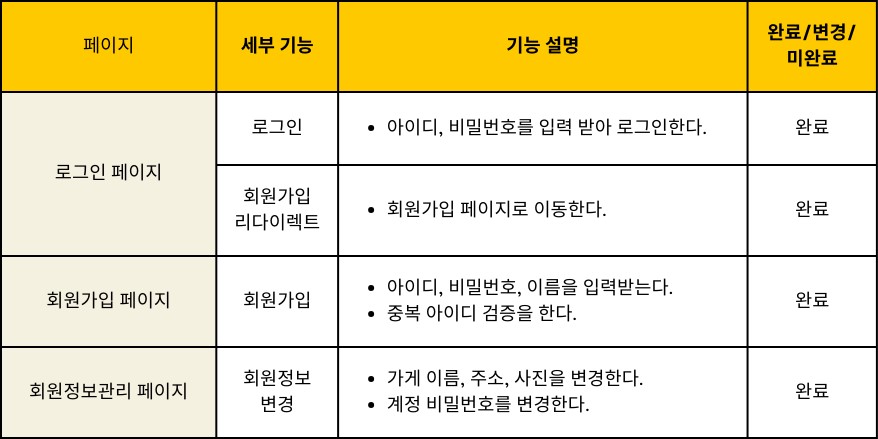
자신이 주문한 메뉴의 내역을 확인할 수 있다. 특정 메뉴를 클릭시 해당 메뉴의 리뷰 작성 페이지로 넘어간다. 자신이 주고자 하는 별점을 클릭하고, 짧은 리뷰를 작성한 뒤, 버튼을 누르면 해당 리뷰가 작성된다.

* **요청 처리**

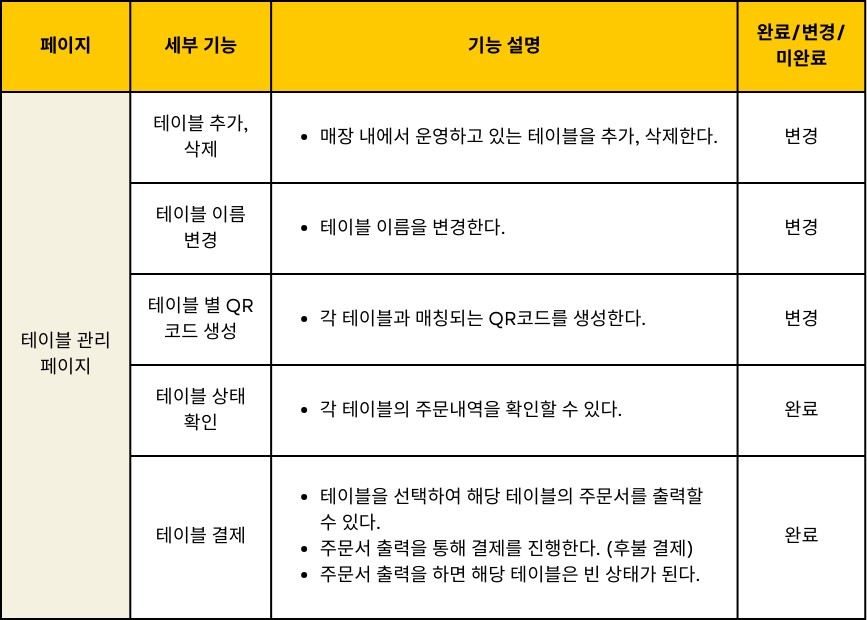
해당 페이지에 접근시, 쿠키에 적힌 주문 내역 정보를 확인하여 해당 메뉴와 요청 URL의 이상 사항은 없는지 검증한다. 검증이 통과된다면 해당 주문 내역에 포함되어 있는 메뉴만을 목록화하여 클라이언트에게 리턴한다. 클라이언트가 해당 메뉴 목록 중에서 특정 메뉴의 리뷰를 작성하여 요청을 보낸다면 해당 내용을 DB에 반영한다.

### 시스템 기능 요구사항

* 회원 관리

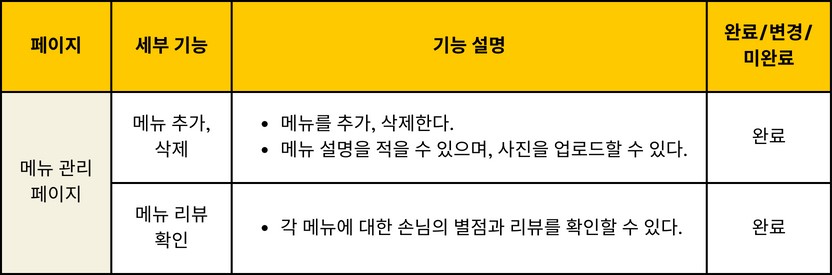


* 테이블 관리

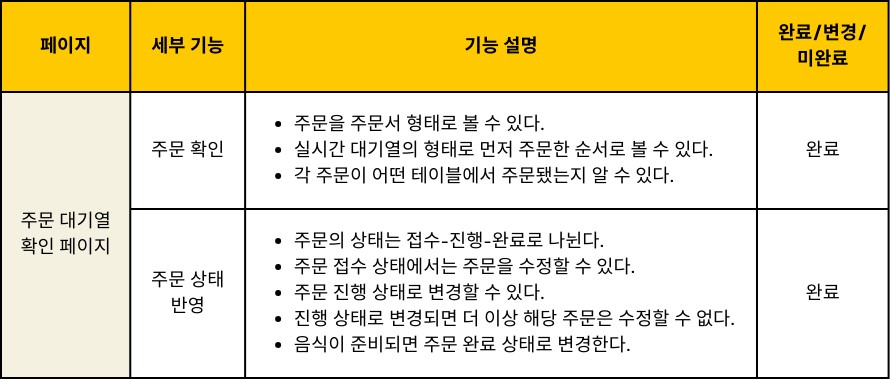


**테이블 괸리 페이지와 테이블 현황 페이지로 나뉘던 것을 하나로 병합하였다.**

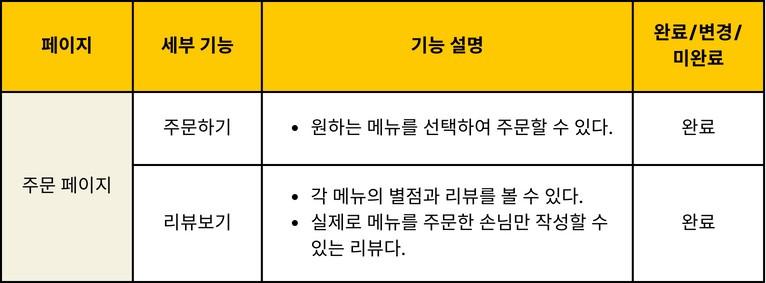
* 메뉴 관리



* 주문 시스템 (점주 기준)



* 주문 시스템 (손님 기준)



### 시스템 비기능(품질) 요구사항

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

### 시스템 구조 및 설계도

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

EZOrder의 시스템은 위와 같이 구성되어 있다.

AWS의 EC2 서버에 Docker를 통해 2개의 컨테이너가 올라간다. 백엔드 프레임워크로는 Spring Boot를 사용되었으며, 다른 컨테이너에 올려진 MySQL을 DB로 사용하고 있다.

Spring Boot 서버와 DB를 연결하기 위해 ORM 기술인 Spring Data JPA를 활용하였으며, 이를 통해 서버에서 이루어지는 데이터 수정 작업이 DB에 반영된다.

프론트엔드 서버는 따로 존재하지 않으며 Thymeleaf 템플릿 엔진을 통해 Spring Boot 서버에서 직접 화면(Html)을 전달한다. Thymeleaf의 자세한 설명은 2.2.5 1)에 자세히 설명되어 있다.

주문 대기열 페이지와 같이 실시간으로 데이터를 반영할 필요가 있는 부분에서는 Web Socket을 활용하였다. Spring WebSocket과 SockJS를 활용하였으며 2.2.5 4)에 관련 내용이 자세히 설명되어 있다.

### 활용/개발된 기술

|  |  |
| --- | --- |
| 사용 기술 | 설명 |
| Spring Framework | Java 애플리케이션 개발을 편하게 할 수 있게 해주는 오픈소스 경량급 애플리케이션 프레임워크 |
| Spring Boot | 스프링을 기반으로 하여 애플리케이션을 쉽게 생성하고 배포하기 위해 개발된 모듈. 기본적으로 의존성 관리와 라이브러리의 설정을 해주며, 웹 서버가 하나의 파일에 내장된 상태로 빌드되기 때문에 배포(Deploy)가 매우 간편해진다. |
| Thymeleaf | JSP, Freemarker와 같은 템플릿 엔진의 일종으로 자바 기반이다. 본 서비스에서는 사용자에게 HTML을 동적으로 렌더링하여 전달하기 위해 사용되었다. |
| Spring WebSocket | Spring 기반의 애플리케이션에서 웹소켓 관련 기능을 지원하는 스프링 하위 API. 본 서비스에서는 주문 대기열 페이지와 인증번호 페이지의 웹소켓 시스템을 지원하기 위해 사용되었다. |
| Sock.js | 웹소켓 프로토콜의 대안으로 실시간 통신을 구현하기 위한 JavaScript 라이브러리이다. 본 서비스에서는 주문 대기열 페이지와 인증번호 페이지의 웹소켓 시스템을 지원하기 위해 사용되었다. |
| STOMP.js | WebSocket 위에서 동작하는 프로토콜로써 클라이언트와 서버가 전송할 메세지의 유형, 형식, 내용들을 정의하는 매커니즘을 STOMP라고 한다. Stomp.js는 Javascript에서 STOMP 프로토콜을 사용하여 WebSocket 기반의 통신에 도움을 주는 라이브러리이다. 본 서비스에서는 주문 대기열 페이지와 인증번호 페이지의 웹소켓 시스템을 지원하기 위해 사용되었다. |
| Spring Scheduler | Spring Framework에서 기본적으로 제공하는 Scheduler로, 특정한 시간에 특정한 작업을 자동으로 실행시키는 역할을 한다. 본 서비스에서는 인증번호를 30분마다 교체하는데 사용되었다. |
| Spring Security | Spring 기반의 애플리케이션의 보안(인증과 권한, 인가 등)을 담당하는 스프링 하위 프레임워크. |
| Spring Data JPA | Hibernate 와 JPA 를 기반으로한 데이터 엑세스 기술을 보다 간편하게 사용할 수 있도록 지원하는 Spring Framework 의 프로젝트 중 하나. 본 서비스에서는 ORM 기법으로 DB와 연계되는 작업을 수행하기 위해 사용되었다. |
| MySQL | 오픈소스형 관계형 데이터베이스. |
| AWS EC2 | AWS 클라우드에서 온디맨드 확장 가능 컴퓨팅 용량을 제공하는 서비스. 본 서비스를 구동시키기 위한 가상 클라우드로 활용되었다. |
| Docker | 커널 컨테이너 기술을 이용하여 만든 컨테이너 기술. EC2 환경에서 MySQL과 Spring Application을 배치하기 위해 사용되었다. |
| AWS S3 | AWS에서 제공하는 객체 스토리지 서비스. 사진을 업로드하고 전달하기 위해 사용되었다. |
| Zxing 라이브러리 | 구글에서 제공하는 오픈소스. QR를 생성하는데 사용되었다. |

**하위 내용은 위 기술의 채택의 이유에 대한 상술이다.**

1. **Thymeleaf의 적용**

Thymeleaf란 JSP, Freemarker와 같은 템플릿 엔진의 일종으로 서버 사이드 렌더링을 가능하게 해준다. 서버 사이드 렌더링(SSR)이란 백엔드 서버에서 HTML을 동적으로 렌더링하여 사용자에게 전달하는 것으로 브라우저에서 렌더링을 처리하는 클라이언트 사이드 렌더링(CSR)과는 다르다.

CSR은 렌더링을 브라우저에게 맡김으로써 서버 트래픽을 감소시키고 더 빠른 인터렉션을 제공할 수 있다는 장점이 있다. 하지만 첫 페이지 로딩 속도가 SSR에 비해 다소 느리다는 점과 검색 엔진 최적화를 위한 추가 작업이 필요하다는 단점이 존재한다. EZOrder 서비스에서는 여러 요소를 고민한 끝에 최종적으로 Thymeleaf를 활용하는 SSR 방식을 채택하였다.

Thymeleaf를 선택한 첫번째 이유는 다음과 같다. 본 팀은 주로 백엔드 기술 스택을 중심으로 학습해왔으며, CSR 기반 프론트엔드 프레임워크에 대한 지식이 부족하다. 만약 CSR를 선택할 경우, 해당 기술 스택을 익히고 능숙하게 사용하는 데에는 상당한 시간이 소요될 것으로 예상했다. 따라서 새로운 프론트엔드 프레임워크를 학습하여 도입하는 대신 팀원 모두가 활용해 본 경험이 있는 Thymeleaf 템플릿 엔진을 선택했다.

Thymeleaf를 선택한 두번째 이유는 스프링과의 호환성이다. Thymeleaf는 스프링 프레임워크와의 통합이 용이하며, HTML을 템플릿으로 사용하고 서버 측에서 데이터를 동적으로 렌더링할 수 있는 기능을 제공한다. 이를 통해 스프링 애플리케이션의 백엔드와 프론트엔드를 간편하게 연결할 수 있다.

또한 Thymeleaf는 HTML과 자바 코드를 혼합하여 사용할 수 있는 템플릿 엔진으로, HTML 파일을 그대로 유지하면서도 서버 측에서 데이터를 쉽게 삽입하고 제어할 수 있다.

이러한 이유들로 Thymeleaf를 선택했으며, 스프링 프레임워크에 익숙한 우리 팀에게는 좋은 선택이라고 판단하였다.

결과적으로 Thymeleaf를 사용함으로써 팀원들이 프론트엔드 개발에 필요한 추가적인 학습과 익숙해질 필요가 없었으며, 프로젝트 진행에 더욱 집중할 수 있었다.

1. **MVC 패턴의 적용**

Spring Boot를 백엔드 프레임워크로 선정하면서 해당 프레임워크가 제시하는 MVC 패턴을 적극적으로 활용하였다.

MVC(모델-뷰-컨트롤러) 패턴은 소프트웨어 디자인 패턴 중 하나로, 소프트웨어의 구성 요소를 모델, 뷰, 컨트롤러 세 가지 역할로 나누어 구현하는 것이다. 각 구성 요소는 역할에 따라 서로 다른 책임을 가지며, 애플리케이션의 구조를 분리하여 유지보수성과 확장성을 향상시킨다.

모델(Model)은 애플리케이션의 데이터와 비즈니스 로직을 담당한다. 데이터의 상태를 표현하고, 데이터를 조작하거나 처리하는 동작을 수행한다. EZOrder 서비스에서는 ORM 방식 중 JPA를 활용하여 자바 코드로 작성된 Entity 클래스를 데이터베이스와 연동시킨다.

뷰(view)는 사용자가 보는 부분으로, 데이터를 시각적으로 표현하고 사용자와의 상호작용을 담당한다. HTML, XML, JSON의 형식으로 데이터를 표시하거나 사용자 입력을 받으며, 서비스 안에서 일어나는 변화를 업데이트하여 사용자에게 보여준다. EZOrder 서비스에서는 Thymeleaf 템플릿 엔진을 활용하여 사용자에게 완성된 HTML과 JS, CSS를 제공한다.

컨트롤러(Controller)는 모델과 뷰 사이의 상호 작용을 조정하고 사용자의 입력을 처리한다. 사용자의 요청을 받아 데이터를 처리하고, 그 결과를 적절한 뷰에 전달하여 화면을 갱신한다. 컨트롤러는 사용자의 요청을 해석하고 그에 따른 작업을 수행하는데, EZOrder 서비스에서는 Service라는 추가적인 단계를 적용하여 이를 더 세밀하게 분리하여, 컨트롤러 단계에서는 입력과 출력에 관련된 부분만 남기고자 했다.

모델에 직접적으로 영향이 가해지는 로직은 전부 Service 단계로 분리하였다. 이는 모델이 DB에 직접적으로 JPA를 통해 연결되어 있는 상태에서, 모델을 변경하는 로직이 상대적으로 수정이 잦은 Controller에 있으면 위험하다는 판단 하에 이루어졌다. 또한 모델에 이루어지는 작업은 CRUD와 같이 공통적인 부분이 많기 때문에, 코드 재사용을 높여서 생산성을 향상시키기 위해 Service를 분리하였다.

1. **웹소켓 기술의 활용**

주문 대기열 페이지와 인증번호 페이지는 끊임없이 변경되는 모델을 실시간으로 반영할 필요가 있다. 그 이유는 다음과 같다.

주문 대기열 페이지는 가게 주인이 로그인하여 확인하는 페이지로, 현재 처리되지 않은 주문들을 보여준다. 이 페이지가 실시간으로 업데이트되지 않으면, 가게 주인은 페이지를 새로고침하지 않는 이상 새로운 주문이 들어온 것을 확인할 수 없다.

인증번호 페이지는 가게 주인이 로그인하여 접근하는 페이지로, 가게 내에 배치된 장비를 통해 손님들에게 제공된다. 손님들은 이 페이지에 표시된 인증번호를 사용하여 주문 페이지에 접근할 수 있다. 인증번호는 30분마다 랜덤으로 변경되며, brute-force 공격을 방지하기 위해 한번 틀릴 경우에도 즉시 변경된다. 이때 실시간으로 인증번호가 반영되지 않으면, 손님들은 주문 페이지에 접근할 수 없다.

위 두 문제들을 해결하기 위해 EZOrder 서비스에서는 웹소켓 기술을 활용하였다. 웹소켓이란 웹 브라우저와 웹 서버 간의 Socket Connetion을 유지해서 통신을 가능하게 하는 프로토콜이다. 이를 활용하면 클라이언트와 서버가 지속적인 연결을 유지하면서 양방향으로 데이터를 실시간으로 주고받을 수 있다. 만약 Socket Connection이 없다면, 서버 쪽 데이터가 업데이트 되더라도 클라이언트 쪽에는 화면은 Refresh하지 않는 한 변경 이전 데이터가 화면에 유지되는 문제가 발생한다.

EZOrder 서비스에서는 Spring WebSocket과 SockJS를 활용하여 웹소켓을 구현하였다. 웹소켓을 적용한 부분과 적용 전, 그리고 적용 후 결과는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 상황 | Web Socket 적용 전 | Web Socket 적용 후 |
| 손님이 인증에 실패하여 인증번호가 교체됨 | 교체 전 인증번호를 계속 띄움. | 실시간으로 인증번호가 교체됨 |
| 스케쥴러에 의해 인증번호가 자동으로 교체됨 | *위와 동일* | *위와 동일* |
| 손님이 음식을 주문함 | 주문 대기열에 주문 정보가 반영되지 않음. | 실시간으로 주문 정보가 주문 대기열에 반영됨. |
| 가게 주인이 주문의 상태를 변경함 (취소, 접수, 완료) | 주문 대기열에 변경된 주문의 상태가 반영되지 않음. | 주문 상태 변경이 주문 대기열에 실시간으로 표시됨 |
| 가게 주인이 특정 테이블의 주문 내역을 비움  (가게 현황 페이지의 ‘비우기’ 버튼) | 해당 테이블에 걸려있던 주문 내역이 여전히 주문 대기열에 남아 있음 | 테이블에 걸려있던 모든 주문이 자동으로 취소 처리되어 주문 대기열에 실시간으로 반영됨. |
| 손님이 자신이 먹은 내역을 결제함  (가게 주인이 결제를 진행하며 가게 현황 페이지의 ‘결제하기’ 버튼을 누름) | 해당 테이블에 걸려있던 주문 내역이 여전히 남아있음. | 테이블에 걸려있던 모든 주문 내역이 자동으로 결제처리되며 이 내용이 주문 대기열에 실시간으로 반영됨. |

1. **인증번호 시스템의 도입**

로그인이나 앱 설치 없이 주문을 할 수 있다는 점은 손님들의 편리함을 증가시킨다는 큰 이점이 존재했지만, 그와 반대로 보안 위험성 또한 증가하였다. 우리가 식별한 가장 큰 문제는 **‘주문 링크가 손님에게 공개’**된다는 점이다. 이 때문에 QR 사진을 저장하여 가게를 벗어나 다른 장소에서 악의적으로 주문을 넣을 가능성이 제기되었다. 따라서 우리는 QR 뿐만 아니라 가게 현장에서만 확인할 수 있는 실시간 인증번호를 도입하였다.

인증번호는 **가게 내에 한 개만 존재**하며, **모든 테이블에서 해당 인증번호로 주문을 넣을 수 있다.** 가게 주인은 손님들이 잘 확인할 수 있는 벽면이나 표지판에 해당 인증번호를 보여줄 수 있는 디스플레이를 부착하기만 하면 된다. 마치 시계처럼 보기 쉬운 곳에 부착해 놓으면, 손님들이 주문 페이지에서 QR을 요구할 때, 그것을 쉽게 확인하고 주문을 넣을 수 있다. QR를 통해 주문 페이지를 접근하더라도, 인증번호를 맞추지 못한다면 주문을 넣을 수 없도록 설계하였다. 따라서 앞서 말한 악의적인 재주문 시도는 인증번호를 확인할 수 없기 때문에 차단된다.

우리는 이 인증번호 시스템의 견고함을 보강하기 위해 더 나은 보안 기능을 부착하였다. 먼저 Brute-Force 공격을 막기 위해 한번 실패 시 해당 인증번호가 자동으로 교체되며, 또한 30분마다 주기적으로 교체되도록 구현하였다.

### 현실적 제한 요소 및 그 해결 방안

1. 결제 시스템의 부재

가장 아쉬운 부분 중 하나로써 실제 결제 시스템과의 연동이 존재하지 않는다. 본래 기획 단계에서의 계획은 결제 처리시 타사의 결제 시스템을 연동을 통해 실제로 결제가 진행되거나, 또는 영수증을 출력하는 것이었다.

그러나 구현 시간이 부족하고 법률적 지식이 동반되어야 하는 부분임을 알게 되어, 해당하는 부분을 구현하지 못했다.

다만 해당 제약 사항이 해결되어 구현이 본격화될 때, 확장이 용이하도록 하기 위해 Json 형태의 주문 내역 정보를 출력하는 기능을 추가하였다.

이 결과가 결제 페이지의 ‘상세보기’ 버튼이다.

1. SSR 방식의 한계

Thymeleaf와 JS를 적극적으로 활용하여 UI의 사용성과 가시성을 높이는데 성공했지만, 이 과정에서 HTML과 Controller의 강결합이 형성된 곳이 일부 존재한다. 이로 인해 추후 웹이 아닌 앱을 통한 서비스 제공을 목표로 서비스가 확장되거나 프론트엔드 서버가 도입될 경우, 해당하는 부분의 강결합을 제거하고 Controller를 분리해야 한다.

### 결과물 목록

* 사용자 매뉴얼
* 배포 가이드
* 테스트 케이스

## 기대효과 및 활용방안

### 기대효과

EZOrder를 도입함으로써 다음과 같은 긍정적인 효과를 기대할 수 있다.

1. **경제적 부담 절감**

* 키오스크나 태블릿 설치가 필요 없어 점주는 초기 투자 비용 및 유지 관리 비용을 절감할 수 있다.

1. **높은 접근성**

* EZOrder는 인터넷을 통해 접속할 수 있으며, 간단한 회원가입 절차만으로 사용이 가능해 누구나 쉽게 접근할 수 있다.

1. **효율적인 관리**

* 메뉴와 주문 관리, 테이블 현황 파악이 용이해져 운영 효율성을 크게 향상시킬 수 있다.

1. **신속한 피드백**

* 손님이 실제로 메뉴를 주문한 후 남긴 리뷰를 통해 메뉴에 대한 정확하고 빠른 피드백을 받을 수 있다. 이는 메뉴 개선 및 서비스 질 향상에 큰 도움이 될 것이다.

### 활용방안

EZOrder의 효과적인 활용을 위해 다음과 같은 방안을 고려할 수 있다:

1. **홍보 및 교육**

* 손님들에게 QR 코드를 활용한 주문 방법을 적극적으로 안내하고, 이를 홍보하는 마케팅 활동을 전개한다.

1. **데이터 분석**

* 주문 데이터를 분석하여 인기 메뉴 파악에 활용합니다. 이를 통해 더 나은 메뉴 구성을 할 수 있다.

1. **고객 만족도 조사**

* 손님이 남긴 리뷰를 분석하여 서비스 개선 방안을 마련합니다. 이를 통해 고객 만족도를 지속적으로 향상시킬 수 있다.

1. **프로모션 연계**

* 손님이 QR 주문을 통해 참여할 수 있는 이벤트를 기획하여 매출 증대를 도모한다.

# 자기평가

QR 코드를 활용한 유효한 주문 시스템을 개발하면서, 간편하면서도 안전한 서비스를 제공하기 위해 특별한 보안 정책인 '인증번호' 시스템을 도입했다. 이를 통해 사용자 친화적이면서도 안전한 주문 환경을 구축했다.

단순한 '주문' 시스템에서 멈추지 않고, 리뷰 시스템과 메뉴 관리 시스템과 같이 가게 주인과 손님 모두가 이득을 얻을 수 있는 다양한 편의 기능을 제공한다.

프론트엔드 서버를 도입하지 않고도 SSR 방식을 통해 수려한 UI를 구성하는 데 성공했다. 결제 부분에서는 현재 실제 결제 시스템과의 연동을 통한 선불 결제가 불가능하다는 점이 아쉽지만, 신용카드 사용이 보편화된 대한민국에서는 후불 결제의 영향력이 커 사용감에 큰 불편함이 없다고 판단된다.

이처럼, 해당 웹 서비스는 안전하고 편리한 주문 시스템과 다양한 부가 기능을 통해 사용자 경험을 향상시키는 데 성공했으며, 따라서 해당 시스템의 사용 가능성은 몹시 긍정적이라고 판단된다.

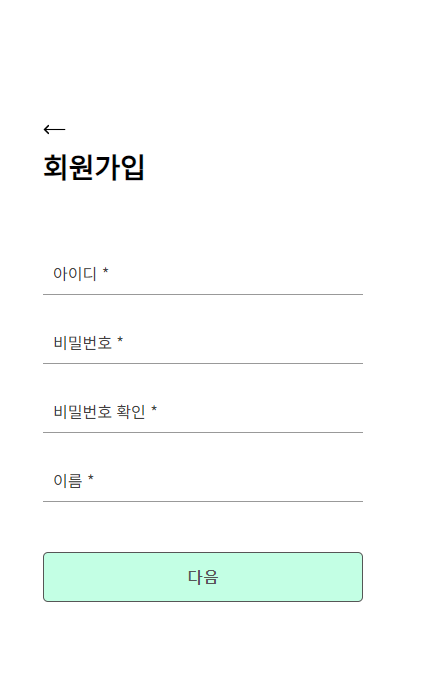
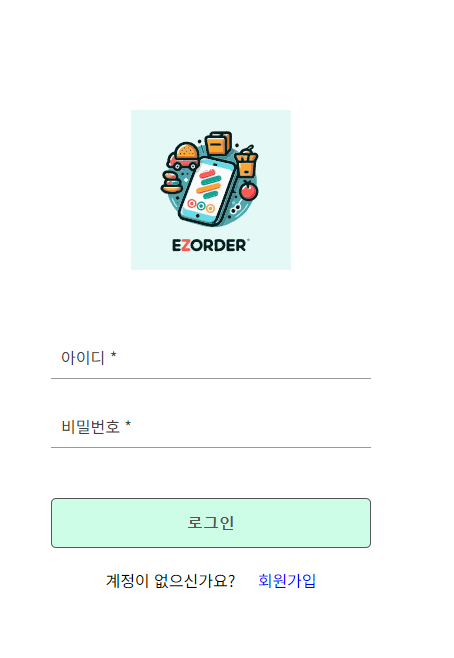
# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 웹페이지 | 2/4분기 서울시 소비자 체감경기와 비대면 경제 | 서울연구원 | 2020 |  |  |
| 2 | 웹페이지 | 2012-2023 스마트폰 사용률 & 브랜드, 스마트워치, 무선이어폰에대한 조사 | 한국갤럽 | 2023 |  |  |
| 3 | 웹페이지 | 푸드테크 사용의향 조사 | 한국농촌경제연구원 |  |  |  |
| 4 | 기사 | 태블릿이냐 QR이냐…테이블오더 시장 주도권 경쟁 치열 | 서울경제 | 2023 | 이덕연 기자 |  |

# 부록

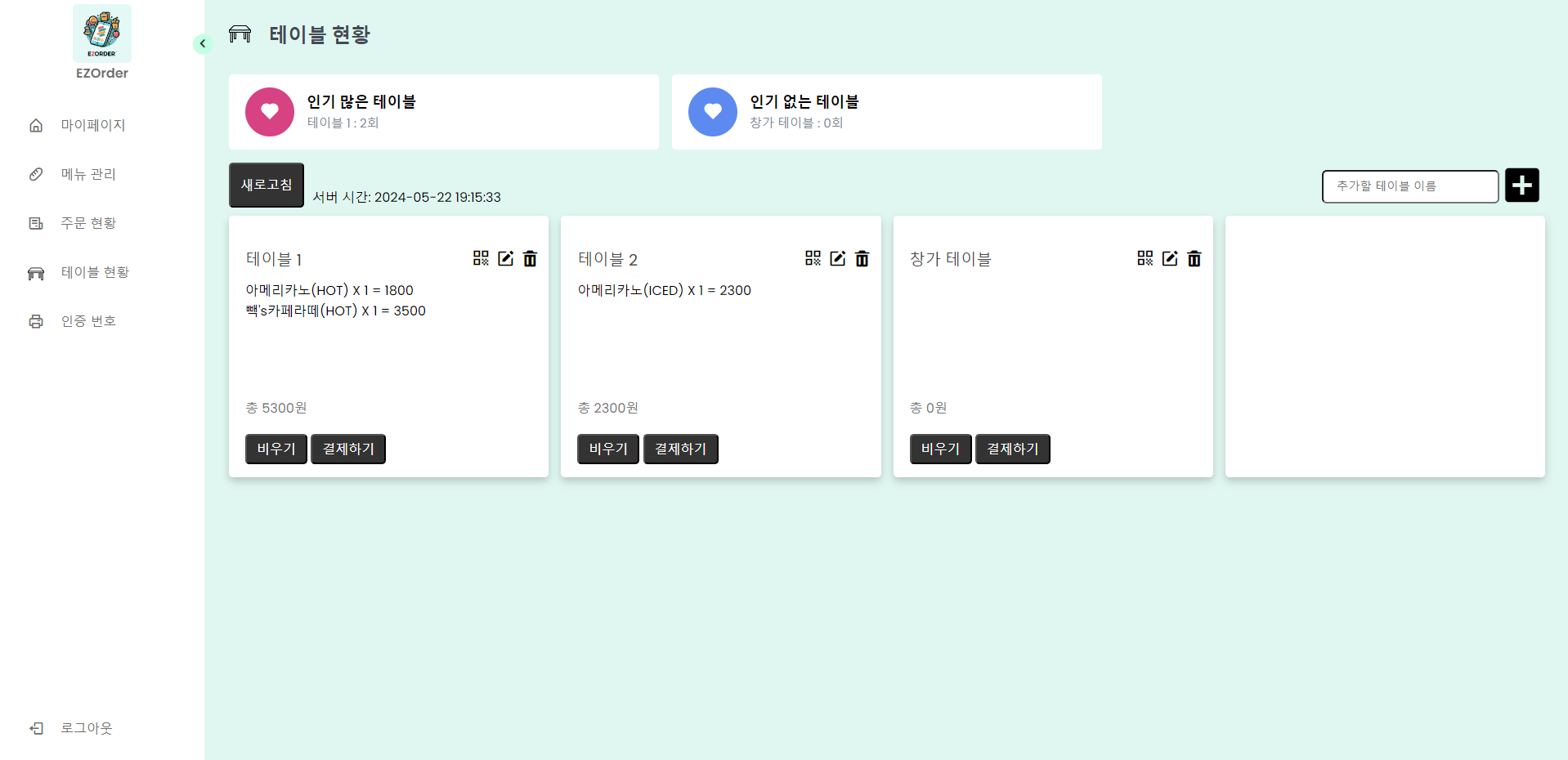
## 사용자 매뉴얼

### 점주 매뉴얼



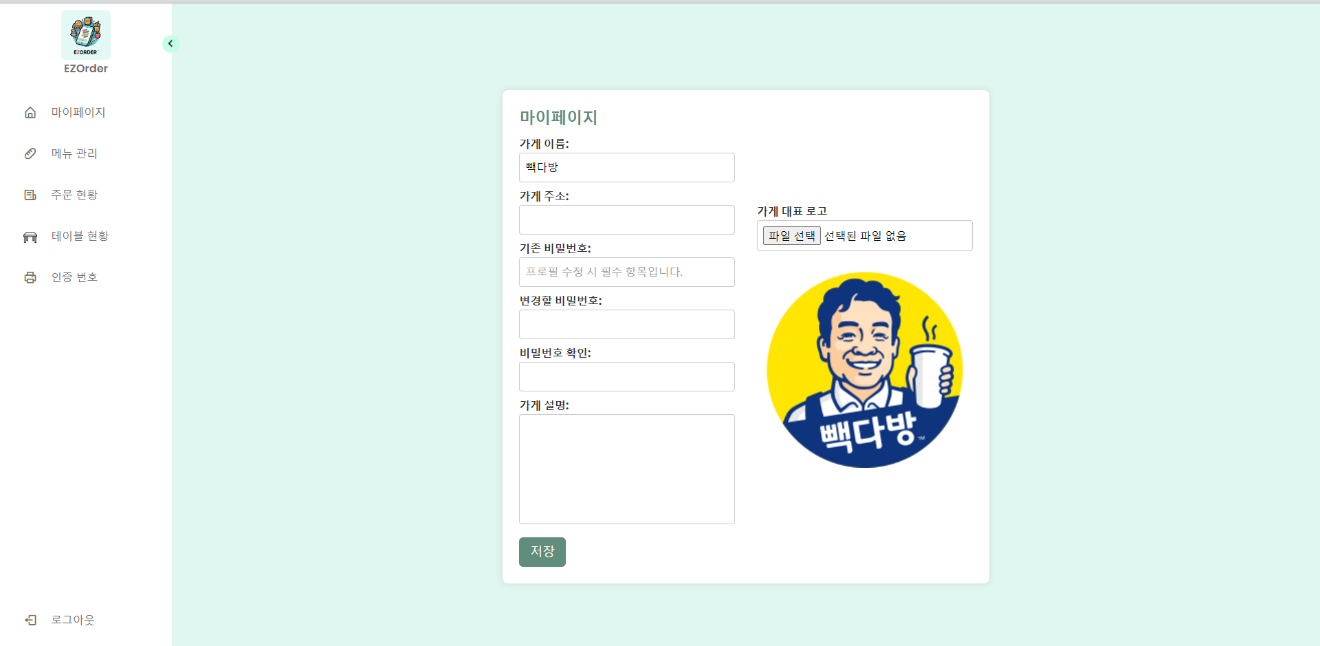
점주는 /login 주소를 통해 로그인 페이지에 접근할 수 있다.

회원가입 페이지로 이동하여 회원가입을 진행할 수 있다.

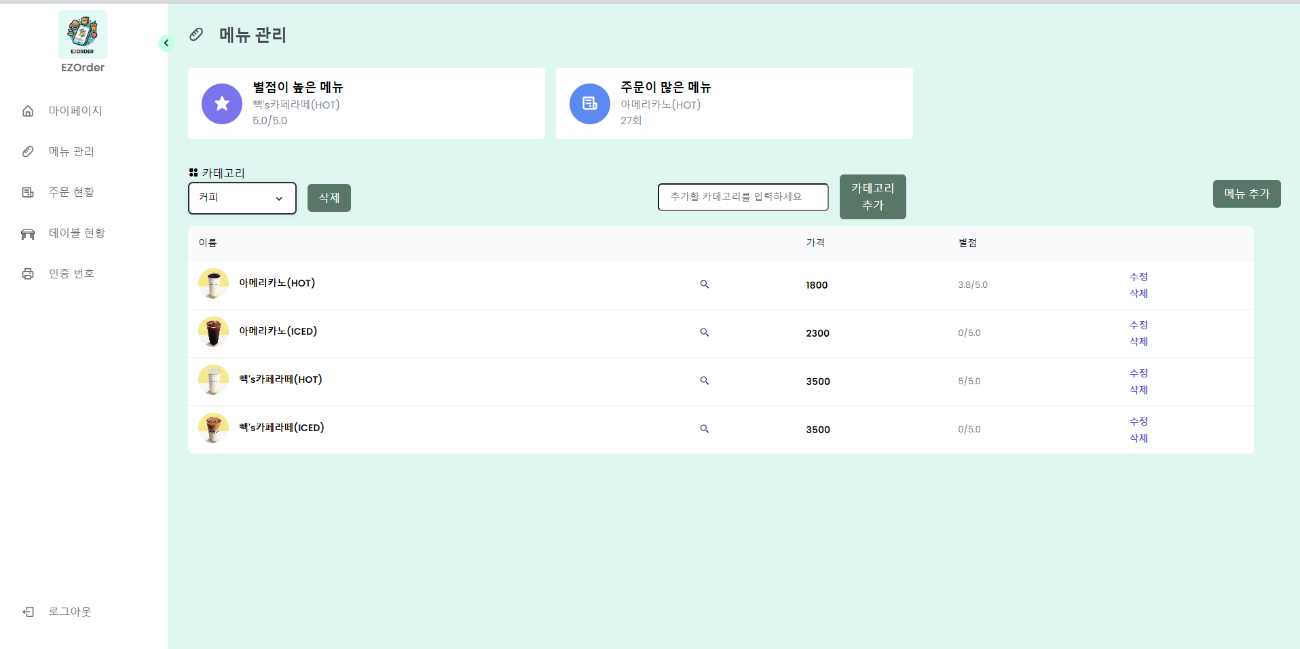


테이블 현황 페이지에서 테이블을 추가할 수 있다.

테이블을 생성한 뒤에는 테이블 이름 우측에 위치한 버튼을 통해 해당 테이블의 QR를 생성하거나, 이름을 변경하고, 삭제할 수 있다.



마이페이지에서는 자신의 계정 정보를 관리하고 가게 로고를 업로드 할 수 있다.



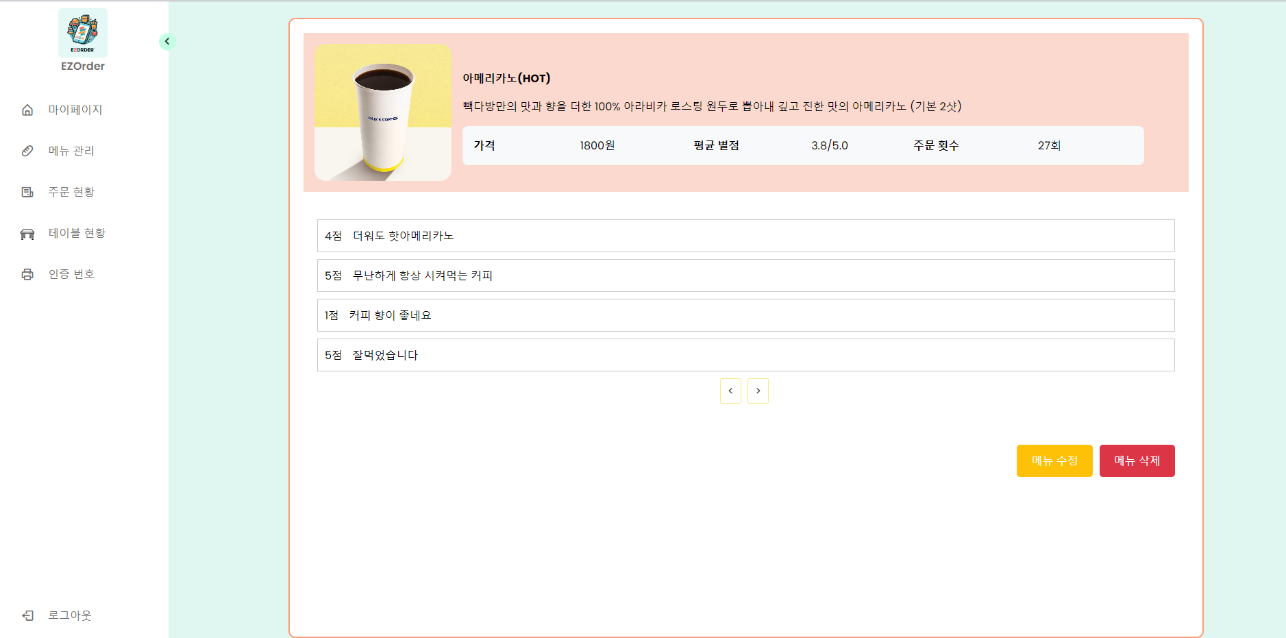
메뉴 관리 페이지에서는 메뉴를 추가하거나 수정, 삭제할 수 있다.

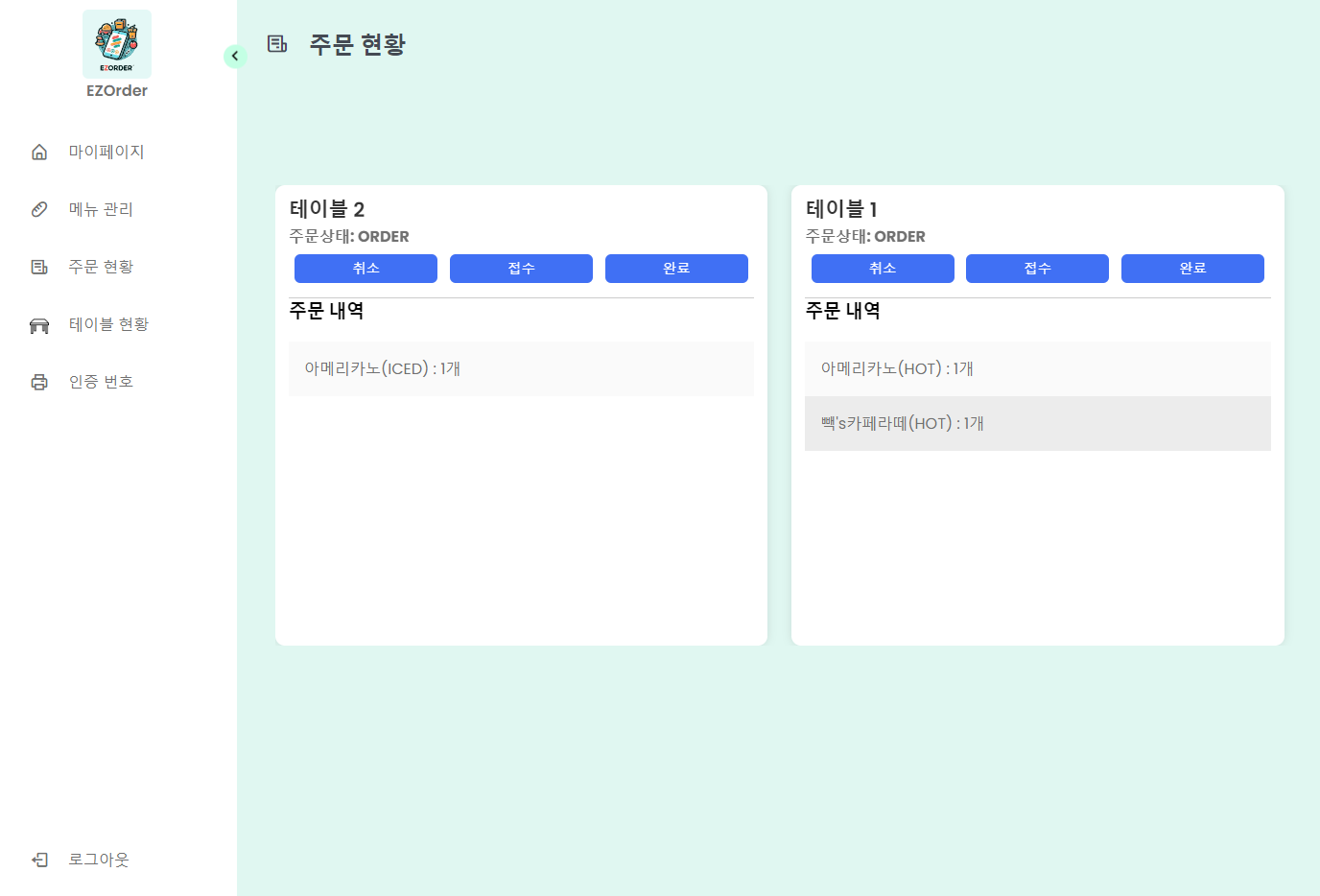
메뉴를 추가하기 위해서는 먼저 해당 메뉴의 카테고리를 지정해야 한다.

카테고리 입력란에 카테고리 이름 (ex: 커피, 샐러드, 고기 등등)을 지정한 뒤, 카테고리 추가 버튼을 클릭하면 카테고리가 생성된다. 이후 메뉴 추가 버튼을 눌러 해당 카테고리를 선택한 후에 메뉴 정보를 입력한다.

메뉴가 생성되었으면, 해당 메뉴의 카테고리를 선택하여 제대로 입력되었는지 확인할 수 있다. 해당 메뉴의 돋보기 버튼을 클릭하여 메뉴 상세보기 페이지에 접근할 수 있다.

이 페이지에서 해당 메뉴에 작성된 리뷰들을 확인할 수 있다.





주문 현황 페이지에서 실시간으로 들어오는 주문을 확인할 수 있다.

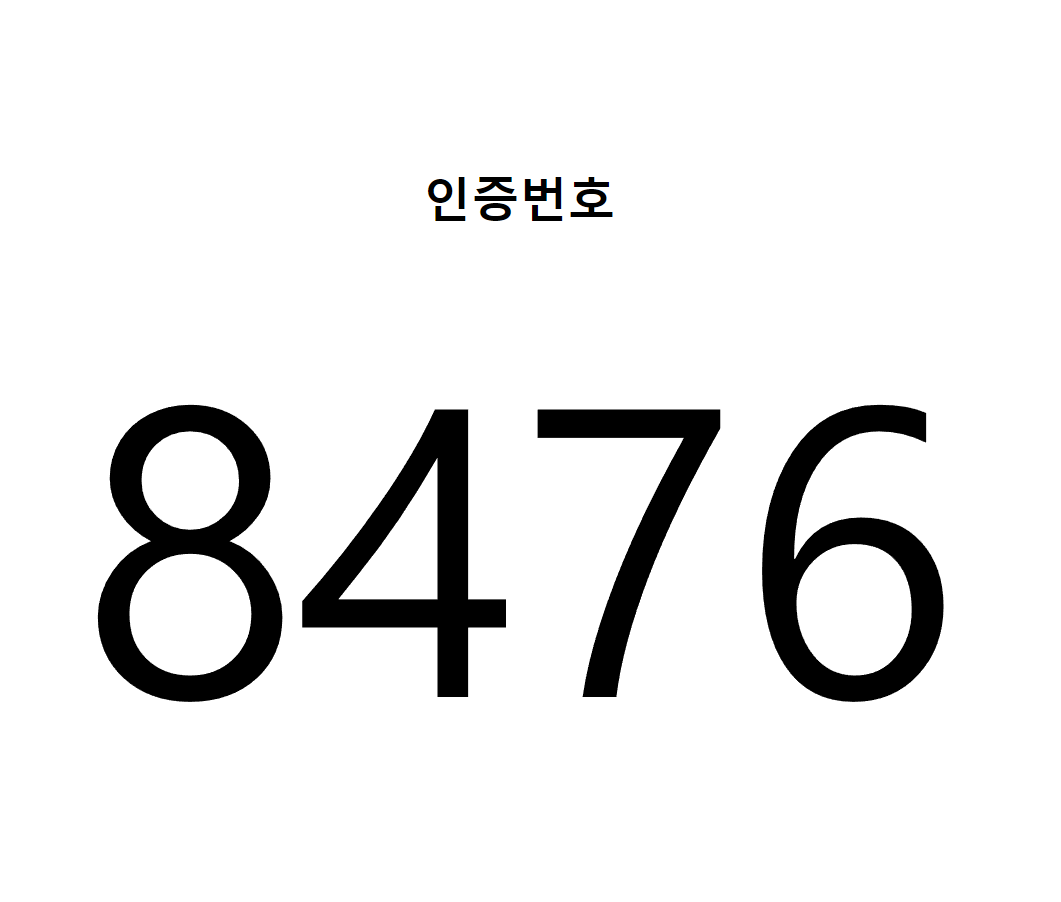
주문에는 3가지 상태 변경 버튼이 존재한다.

기본적으로 주문이 들어온 상태에서 아무런 조작을 가하지 않았다면 ‘ORDER’상태이다.

손님이 부득이한 이유 (주문 실수, 가게 이탈 등등)로 주문 취소를 요구하면 취소 버튼을 누르면 된다. 이러면 대기열에서 주문이 사라진다.

주문을 맡아서 진행중이라면 ‘접수’ 버튼을 누른다. 이경우, 주문상태가 ‘PROGRESS’로 변경된다. 이는 ‘현재 누군가가 해당 주문을 맡아서 조리중’임을 의미한다.

조리가 완료되고 주문을 처리했다면 ‘완료’ 버튼을 누르면 된다. 이 경우 주문이 주문 대기열에서 처리되어 사라진다.



손님들에게 주문을 받고자 할 때, 손님들에게 **인증번호**를 보여줘야 한다. 가게 안 모든 테이블이 하나의 인증번호를 공유한다. 따라서 가게 내에 잘 보이는 곳에 해당 인증번호 창을 띄워둔 디스플레이 (모바일, 태블릿 PC, 모니터 등등)을 배치하는 것이 좋다. 마치 시계처럼 잘 보이는 곳에 배치한다면, 손님들이 쉽게 인증번호를 찾아서 입력하여 주문을 진행할 수 있다. 해당 가게의 모든 테이블이 해당 인증번호로 주문을 진행할 수 있다.



테이블 현황 페이지에서 특정 테이블의 주문을 처리하기 위해서는 결제하기 버튼을 누르면 된다. 그렇게 하면 위와 같은 테이블 결제 페이지가 등장한다. 결제 페이지에서 ‘결제하기’를 누르거나 현황 페이지에서 ‘비우기’를 클릭할 시 해당 테이블의 주문 내역이 전부 사라지므로 주의해야 한다!

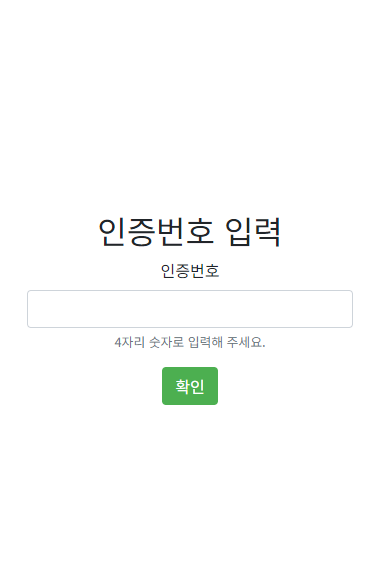
### 손님 매뉴얼

가게에 도착한 손님은 테이블에 착석한 뒤, 해당 테이블의 QR를 타고 페이지에 접속한다.

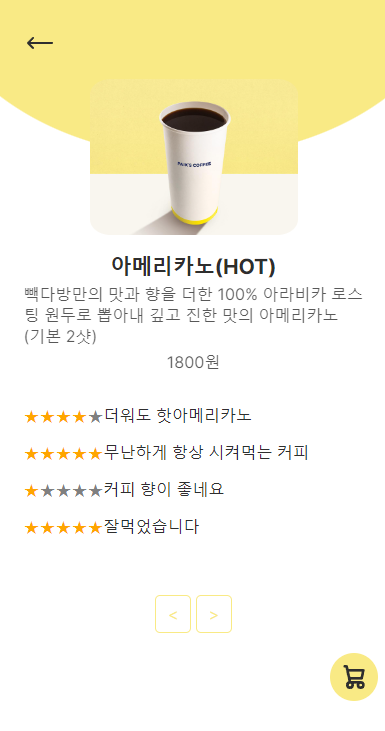


가장 처음 등장한 페이지에서 ‘주문하기’를 클릭한다.

그럴 경우, 인증번호 인증 페이지가 등장한다.



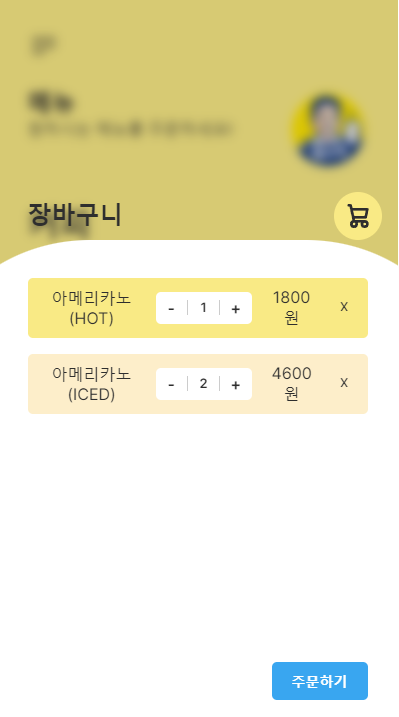
가게 안에 배치된 인증번호 디스플레이를 통해 인증번호를 입력했다면 다음과 같은 주문페이지가 등장한다.



상세보기를 클릭시 오른쪽 사진과 같이 리뷰를 확인하고 설명을 읽을 수 있다.

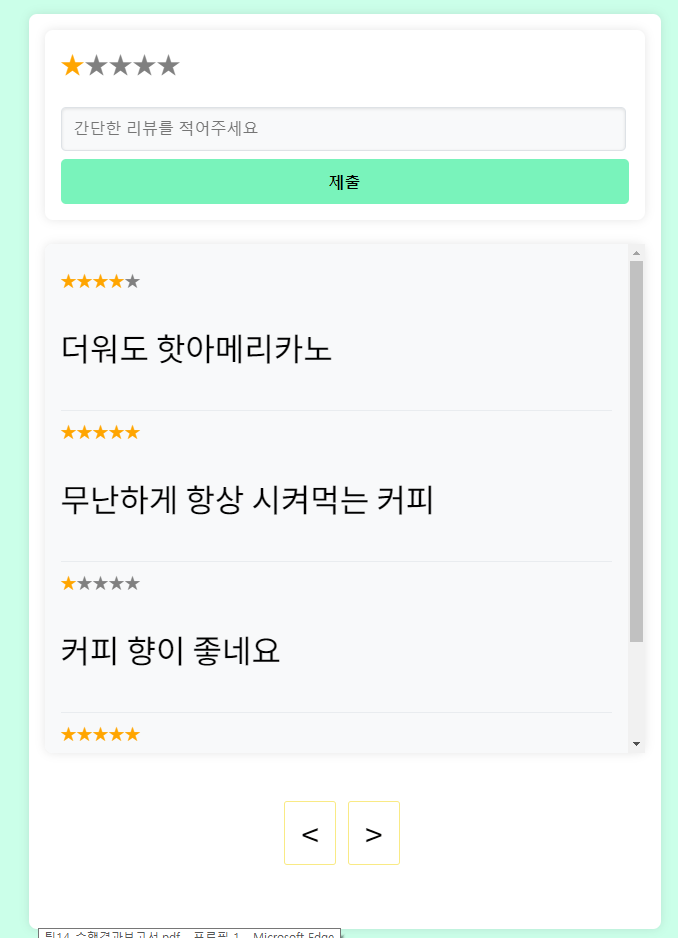
담기 버튼을 클릭하여 메뉴를 장바구니에 담을 수 있다.

오른쪽 하단 카트 아이콘을 클릭하면 다음과 같이 장바구니 내역을 볼 수 있다.



수량을 조절하거나 주문을 진행할 수 있다.

음식을 먹은 뒤 리뷰를 작성하고자 할 때는, 아까 주문과 같은 방식으로 리뷰 페이지에 접근할 수 있다. 다만 이경우 인증번호가 필요 없다.



리뷰 목록 페이지에는 이 기기에서 2시간 이내에 주문한 내역이 있는 메뉴만이 등장한다.

해당 메뉴를 클릭하면 오른쪽 사진과 같은 리뷰 작성 페이지가 등장한다. 별 모양 버튼을 클릭하여 별점을 선택한 뒤, 간단한 리뷰를 작성하여 ‘제출’ 버튼을 누르면 리뷰가 작성된다.

## 배포 가이드

**1) AWS ec2 서버 인스턴스를 실행 후, ssh 터미널로 접속한다.**

-> 본 가이드는 ubuntu를 기준으로 작성되었다.

**2) Amazon S3 버킷을 생성한다.**

S3 버킷을 생성할 때, ACL을 활성화하고 퍼블릭 엑세스를 허용한다.

S3 버킷 사용자를 생성하고, 해당 사용자의 access-key와 secret-key를 기록한다.

**3) 해당 git repository의 master 브랜치를 Clone한다.**

$ git clone https://github.com/kookmin-sw/capstone-2024-32.git

**4) 해당 디렉토리에 접근하여 setup.sh 쉘 스크립트를 실행한다.**

$ chmod +x setup.sh

$ ./setup.sh

-> 도중에 뜨는 do you want to continue와 같은 부분에서는 모두 승인한다.

**5) docker-compose.yaml 파일을 수정한다.**

services.mysql.environment에서

MYSQL\_ROOT\_PASSWORD에 루트 비밀번호를 작성한다.

그 후에 services.spring-app.environment에 .

SPRING\_DATASOURCE\_PASSWORD에 위에 작성한 루트 비밀번호를 작성한다.

**6) src/main/resources/application.yaml을 수정한다.**

spring.datasources.password에 위에 작성한 루트 비밀번호를 작성한다.

그 후에 qrcode.url에 현재 ec2 인스턴스의 퍼블릭 IPv4 DNS 주소:8080을 작성한다.

그 후에 버킷명과 액세스키, 시크릿키를 cloud.aws.s3.bucket과 credentials.access-key, credentials.secret-key에 입력한다.

**7) 이후에 docker compose와 gradlew을 통해 빌드한 후 실행한다.**

$ sudo docker compose up -d mysql # MySQL 컨테이너 실행

$ chmod 755 ./gradlew # gradle 활용하여 빌드

$ ./gradlew clean build

$ sudo docker compose build spring-app # Spring 애플리케이션 빌드

$ sudo docker compose up -d # 빌드된 파일 전부 실행

**8) 확인**

퍼블릭 IPv4 DNS 주소에 8080 포트로 접속하면 해당 application이 작동하는지 확인할 수 있다.

## 테스트 케이스

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 테스트 방법 | 기대 결과 | 테스트 결과 |
| 회원관리 | 로그인  페이지 | 로그인 창 이동 | 로그인하지 않고는 접속할 수 없는 url에 접근한다.  (ex: “/admin/seat/view”) | 로그인 창으로 이동한다. | 성공 |
| 입력 값 유효성 확인 | 아이디와 비밀번호를 입력한다. | 하나라도 입력하지 않았을 경우 로그인이 되지 않는다. | 성공 |
| 유효한 사용자인지 확인 | 등록된 사용자의 아이디와 비밀번호를 입력한다. | 로그인에 성공한다. | 성공 |
| 유효하지 않은 사용자 확인 | 등록되지 않은 사용자의 아이디와 비밀번호를 입력한다. | ‘로그인에 실패하였습니다.’라는 알림이 뜬다. | 성공 |
| 회원가입  페이지 | 회원가입 창 이동 | 로그인 페이지에서 회원가입 버튼을 누른다. | 회원가입 페이지로 이동한다. | 성공 |
| 입력 값 유효성 확인 | 아이디와 비밀번호, 비밀번호 확인, 이름을 입력한다. | 하나라도 입력하지 않았을 경우 회원가입이 되지 않는다. | 성공 |
| 비밀번호 확인 | 비밀번호가 비밀번호 확인 값과 일치하지 않게 입력한다. | 일치하지 않을 경우, 경고 애니메이션이 출력된다. | 성공 |
| 아이디 입력값 유효성 확인 | 중복된 아이디를 입력한다. | ‘아이디가 중복입니다.’라는 경고를 띄운다. | 성공 |
| 비밀번호 입력값 유효성 확인 | 8자리 이하의 비밀번호를 입력한다. | ‘비밀번호는 8~16자 사이로 지어주세요’라는 경고를 띄운다 | 성공 |
| 회원가입 | 정상적으로 중복되지 않은 아이디를 입력하여 회원가입을 시도한다. | 회원가입에 성공하고 로그인창으로 이동한다. | 성공 |
| 테이블  관리 | 테이블 현황 | 테이블  현황표 이동 | 사이드바에서 ‘테이블 현황’ 버튼을 누른다. | 테이블 현황표 페이지로 이동한다. | 성공 |
| 새로고침 | ‘새로고침’ 버튼을 누른다. | 서버시간과 현황표가 최신화된다. | 성공 |
| 테이블 관리 | 테이블 생성 유효값 확인 | 테이블 현황 페이지에서 추가할 테이블의 이름을 입력하지 않고 + 버튼을 누른다. | 테이블이 생성되지 않고 경고창이 뜬다. | 성공 |
| 테이블 생성 | 테이블 현황 페이지에서 유효한 테이블의 이름을 입력하고 + 버튼을 누른다. | 해당 이름을 가진 테이블이 생성된다. | 성공 |
| 테이블 삭제 | 테이블 현황 페이지에서 이미 존재하는 테이블의 삭제 버튼을 누른다. | 해당 이름을 가진 테이블이 삭제된다. | 성공 |
| 테이블 이름 수정 | 테이블 현황 페이지에서 테이블의 이름 수정 버튼을 클릭 후 이름을 입력한다. | 해당 이름으로 테이블이 변경된다. | 성공 |
| 테이블 비우기 | 테이블 현황 페이지에서 특정 테이블의 ‘비우기’ 버튼을 클릭한다. | 해당 테이블의 주문 내역이 비워진다. | 성공 |
| 테이블 결제 페이지 이동 | 테이블 현황 페이지에서 특정 테이블의 결제하기 버튼을 클릭한다. | 테이블 결제 페이지로 이동한다. | 성공 |
| 테이블 결제하기 | 테이블 결제 페이지에서 결제하기 버튼을 누른다. | 해당 테이블의 주문 내역이 사라지고, 테이블 현황 페이지로 이동한다. | 성공 |
| 테이블 결제 내역 상세보기 | 테이블 결제 페이지에서 ‘상세보기’ 버튼을 누른다. | 해당 테이블의 주문 내역이 json 형태로 출력된다. | 성공 |
| QR | QR | QR 생성 | 테이블 현황 페이지에서 원하는 테이블의 QR 버튼을 누른다. | 해당 테이블의 주문 페이지로 이동할 수 있는 QR 코드가 등장한다. | 성공 |
| QR 주소 취약점 검증 | ‘/guest/\*/\*/login’ 주소에 \*에 해당하는 곳에 랜덤한 값을 입력하고 이동한다. | 에러 페이지가 등장한다. | 성공 |
| QR 이동 | 생성된 QR이 제공하는 링크로 이동한다. | QR 손님 페이지가 등장한다. | 성공 |
| 회원정보 관리 | 마이페이지 | 마이페이지 이동 | 사이드바에서 마이페이지를 클릭한다. | 마이페이지 설정 폼으로 이동한다. 기존에 작성한 정보가 이미 채워져있다. | 성공 |
| 유효값 검증 | 가게 이름을 공란으로 바꾸고 ‘저장’ 버튼을 누른다. | 경고창이 뜬다. | 성공 |
| 비밀번호 검증 | 기존 비밀번호를 입력하지 않고 정보를 수정한 뒤 ‘저장’을 누른다. | 경고창이 뜬다. | 성공 |
| 사진 업로드 | 로고 사진을 업로드한 뒤 ‘저장’을 누른다. | 해당 마이페이지에 로고 사진이 적용된다. | 성공 |
| 사진 제외 업로드 | 로고 사진을 제외하고 나머지 정보를 변경한 뒤 ‘저장’을 누른다. | 로고 사진만 공란인 채로 정보 변경이 적용된다. | 성공 |
| 비밀번호 변경 유효값 검증 | 비밀번호 변경과 비밀번호 변경 확인을 유효하지 않은 값으로 설정한 뒤 저장을 누른다. | ‘비밀번호는 8~16자 사이로 지어주세요.’와 같은 경고 메시지가 뜬다. | 성공 |
| 비밀번호 변경 | 변경 비밀번호와 비밀번호 확인 칸을 유효한 값으로 채우고 기존 비밀번호도 작성한 뒤 ‘저장’을 누른다. | 비밀번호가 변경된다. | 성공 |
| 메뉴 관리 | 메뉴 페이지 | 메뉴 페이지 이동 | 사이드바에서 ‘메뉴 관리’를 클릭한다. | 메뉴 페이지로 이동한다.  이때 메뉴 카테고리와 메뉴 통계, 메뉴 목록이 일치해야 한다. | 성공 |
| 메뉴 카테고리 | 메뉴 카테고리  변경 | 메뉴 카테고리를 클릭하여 선택한다. | 해당 카테고리에 속하는 메뉴 리스트가 보여진다. | 성공 |
| 메뉴 카테고리 유효값 검증 | 메뉴 카테고리 입력칸에 유효하지 않은 값을 넣고 ‘추가’ 버튼을 클릭한다. | 경고창이 등장한다. | 성공 |
| 메뉴 카테고리  생성 | 메뉴 카테고리 입력 칸에 유효한 값을 넣고 ‘추가’ 버튼을 클릭한다. | 메뉴 카테고리가 추가된다. | 성공 |
| 메뉴 카테고리 삭제 | 특정 카테고리를 선택한 뒤, 삭제 버튼을 클릭한다. | 메뉴 카테고리가 삭제된다. | 성공 |
| 메뉴 관리 | 메뉴 추가  페이지 이동 | ‘메뉴 관리’ 페이지에서 메뉴 추가 버튼을 클릭한다. | 메뉴 추가 페이지로 이동한다. | 성공 |
| 메뉴 유효값 검증 | 메뉴 추가 페이지에서 유효하지 않은 값을 입력한 뒤 ‘저장’ 버튼을 클릭한다. | 경고창 혹은 에러페이지가 등장한다. | 성공 |
| 메뉴 사진 업로드 | 메뉴 추가 페이지에서 사진을 업로드한다. | 해당 사진이 메뉴에 반영된다. | 성공 |
| 메뉴 빈 사진 업로드 | 사진만을 제외하고 나머지 유효한 값들을 입력한 뒤, 저장한다. | 사진만 비어있는 메뉴가 생성된다. | 성공 |
| 메뉴 추가 | 메뉴 추가 페이지에서 유효한 값을 입력한 뒤 메뉴를 추가한다. | 메뉴가 생성된다. | 성공 |
| 메뉴 수정 페이지 이동 | 메뉴 관리 페이지에서 특정 메뉴의 ‘수정’ 버튼을 클릭한다. | 해당 메뉴의 정보가 담긴 수정 폼으로 이동한다. | 성공 |
| 메뉴 수정 유효값 검증 | 메뉴 수정 페이지에서 유효하지 않은 값을 입력한 뒤 저장한다. | 경고창이 나타난다. | 성공 |
| 메뉴 수정 저장 | 메뉴 수정 페이지에서 유효한 값을 입력한 뒤 저장한다. | 메뉴에 수정한 값이 반영된다. | 성공 |
| 메뉴 수정 페이지 URL 테스트 | 메뉴 수정 url (“/admin/item/update/\*”)에서 \*에 해당하는 부분에 랜덤한 숫자를 넣는다. | 에러 페이지로 이동한다. | 성공 |
| 메뉴 삭제 | 특정 메뉴를 선택하여 삭제한다. | 메뉴가 삭제된다. | 성공 |
| 메뉴 상세보기 | 메뉴 상세보기 페이지 이동 | 특정 메뉴의 돋보기 버튼을 클릭한다. | 해당 메뉴의 상세보기 페이지로 이동한다. | 성공 |
| 메뉴 상세보기 페이지 URL 테스트 | 메뉴 상세보기 url (“/admin/item/detail/\*”)에서 \*에 해당하는 부분에 랜덤한 숫자를 넣는다. | 에러 페이지로 이동한다. | 성공 |
| 메뉴 수정 버튼 테스트 | 메뉴 상세보기 페이지에서 메뉴 수정 버튼을 클릭한다. | 메뉴 수정 폼으로 이동한다. | 성공 |
| 메뉴 삭제 버튼 테스트 | 메뉴 상세보기 페이지에서 메뉴 삭제 버튼을 클릭한다. | 메뉴가 삭제된다. | 성공 |
| 리뷰 페이지네이션 테스트 | 메뉴 상세보기 페이지에서 리뷰 ‘<’, ‘>’를 클릭한다. | url에 page 파라미터가 생성되어 해당 page parameter가 변경된 값으로 이동한다. 이때 리뷰의 개수에 따라 이동 가능한 page가 정해져있다. | 성공 |
| 주문 관리 | 주문 대기열 | 주문 대기열 페이지 이동 | 사이드바에서 ‘주문현황’ 버튼을 클릭한다. | 주문 대기열 페이지로 이동한다. | 성공 |
| 주문 대기열 실시간 반영 검증 | 손님의 기기에서 주문을 진행한다. | 주문 대기열에 실시간으로 주문이 추가된다. | 성공 |
| 주문 상태 변경 | 주문의 상태를 나타내는 버튼 3개를 클릭한다. | 해당 상태 변경에 따라 ‘취소’ 혹은 ‘완료’라면 주문 대기열에서 주문이 사라진다.  ‘진행중’이라면 주문의 상태가 변경된다. | 성공 |
| 인증 번호 | 인증번호 | 인증번호 페이지 이동 | 사이드바에서 인증번호 버튼을 클릭한다. | 인증번호 페이지로 이동한다. | 성공 |
| 인증번호 실시간 반영 검증 | 30분을 기다리거나 손님의 기기에서 인증번호 검증을 실패한다. | 인증번호가 교체된다. | 성공 |
| 인증번호 검증 | 인증번호 입력 | 손님의 기기에서 인증번호 검증 페이지로 접속하여 인증번호를 입력한 뒤 제출한다. | 인증번호가 틀리다면 인증번호가 변경되고, 인증번호가 맞다면 주문 페이지로 이동한다. | 성공 |
| 주문 | 주문 | 메뉴 선택 | 주문 페이지에서 특정 메뉴를 클릭한다. | 장바구니에 해당 메뉴가 1개 담긴다. | 성공 |
| 메뉴 상세보기 | 주문 페이지에서 특정 메뉴의 상세보기 버튼을 클릭한다. | 해당 메뉴의 설명과 리뷰 목록이 나타난다. | 성공 |
| 장바구니 수량 조절하기 | 장바구니 버튼을 클릭 후 수량 조절 버튼을 클릭하거나 제거한다. | 해당 조작이 장바구니에 반영된다. | 성공 |
| 주문하기 | 장바구니에서 주문 버튼을 클릭한다. | 장바구니의 내용이 주문된다. | 성공 |
| 리뷰 | 리뷰하기 | 리뷰 페이지 이동 | 손님의 기기에서 QR을 촬영하여 페이지를 이동한 뒤 리뷰하기 버튼을 클릭한다. | 해당 기기에서 2시간 이내로 주문한 내역이 있는 메뉴의 목록이 등장한다. | 성공 |
| 리뷰 메뉴 선택하기 | 리뷰 페이지에서 목록에 있는 특정 메뉴를 클릭한다. | 해당 메뉴의 리뷰 작성 페이지로 이동한다. | 성공 |
| 리뷰 작성하기 | 리뷰 작성 페이지에서 리뷰를 작성한다. | 작성된 리뷰가 리뷰 목록에 추가된다. | 성공 |