

# QR코드를 이용한 택배 안심배송 서비스

팀	명 :	택배왔조
지도	교수 :	이병천 교수님
팀	장 :	권현주
팀	원 :	신세영 유현진

2021. 10. 24  
중부대학교 정보보호학과

# 목 차

## 1. 서론

1.1 연구 배경 .....	4
1.2 연구 필요성 .....	4
1.3 연구 목표 및 주제 선정 .....	5

## 2. 관련 연구

2.1 QR Code .....	5
2.2 전자 서명 .....	5
2.3 X.509 인증서 .....	5
2.4 Node.js .....	6
2.5 Angular .....	6
2.6 Ionic .....	6
2.7 Firebase .....	7
2.8 Node-forge .....	7
2.9 JavaScript .....	7
2.10 TypeScript .....	7

## 3. 본론

3.1 서비스 설계 .....	8
3.2 DB 자료구조 설계 .....	9
3.3 프로그램 구성 및 구성원 별 상세내용 .....	14
3.3.1 고객 .....	14
3.3.2 택배기사 .....	14

3.3.3 관리자 .....	14
3.4 기능 별 상세내용 데모 .....	15
3.4.1 고객 회원가입, 로그인, 기본 배송지 지정 .....	15
3.4.2 고객 주문단계 .....	17
3.4.3 관리자 로그인 .....	18
3.4.4 관리자 고객 주문 리스트 확인 및 QR 운송장 출력 ..	19
3.4.5 관리자 택배기사 계정 생성 .....	20
3.4.6 택배기사 로그인 << 및 전자서명>> .....	22
3.4.7 택배기사 택배 배정 및 배송지 지도 표시 .....	22
3.4.8 택배기사 택배 배송 완료처리 .....	23
3.4.9 고객 택배 확인 .....	24
3.4.10 고객 주문 리스트 확인 및 환불처리 .....	25

## 4. 결론

4.1 결론 및 기대효과 .....	26
4.2 향후 과제 .....	26

## 5. 별첨

5.1 참고 자료 .....	27
5.2 소스 코드 .....	27
5.3 발표 자료 .....	64

# 1. 서론

## 1.1 연구 배경

한국 통합물류협회에 따르면 국내의 택배 산업은 코로나바이러스 사태로 인해 2020년 국내 총 택배 물량이 33억7373만 개로 집계되었으며 27억8980만 개였던 전년 대비 20.9%의 증가를 보였다. 택배 산업은 크게 확장되는 반면에 운송장에는 고객의 개인정보가 그대로 노출되어 있어 사회 공학적 공격과 피싱 등 개인정보를 이용한 범죄들이 계속 해서 발생하고 있다. 2012년 9월 행정안전부에서는 “택배를 받고 나서 운송장을 함부로 방치하여 소중한 개인정보가 노출되어 스팸, 보이스피싱 등에 악용되는 사례가 많다”라고 밝혔으며 2018년 8월 인터넷 방송인의 택배 운송장을 촬영하여 장난전화, 스토킹 등 피해 사례를 밝히기도 했다. 그리고 최근 2021년 3월에 발생한 노원 세 모녀 살인 사건은 택배 운송장에 그대로 노출되어 있는 개인정보가 이용된 것으로 알려져 택배 운송장에 무방비하게 노출되어 있는 개인정보에 대한 우려의 목소리가 점차 커지고 있다.

## 1.2 연구 필요성

아래의 그림은 우리가 흔히 알고 있는 택배의 운송장이다. 운송장을 살펴보면 보낸 사람의 이름, 주소, 우편번호, 전화번호 그리고 받는 사람의 주소, 전화번호, 등기번호 심지어 어떤 상품을 주문했는지까지 개인정보가 그대로 노출되어있는 것을 확인 할 수 있다. 또한, 택배를 배달하는 기사님의 관점으로 바라봤을 때 운송장에는 바코드로 해당 운송장을 표시하고 있기 때문에 기사님들이 바코드 리더기를 소지하고 다녀야하는 번거로움을 감수해야한다.

[ 그림 1-1. 기존 운송장 ]

## 1.3 연구 목적 및 주제 선정

본 프로젝트에서는 개인정보(이름, 주소, 전화번호, 주문번호 등)가 포함되어있는 기존의 운송장과는 다르게 QR코드로 대체하여 개인정보를 보호하면서도 택배 업무를 효율적으로 수행할 수 있도록 하는 안심택배서비스를 설계하였다. 구체적으로는 택배물품에 운송장번호를 인쇄하고 이것이 포함된 QR코드를 부착한다. 택배기사는 기존의 바코드 리더기가 아닌 본인의 휴대폰의 서비스용 안심 택배 웹앱으로 로그인하여 인증되어있는 택배 기사임을 확인한 후 QR코드를 스캔하여 해당물품을 배정받을 수 있고 배달주소는 서버에서 보내주게 되며 지도상에 주소가 표시된다. 인가되지 않은 타인이 QR코드를 스캔하

면 서버에서 정보를 제공하지 않는다. 배달이 완료된 후 구매자는 QR코드를 스캔하여 자신의 물품이 맞는지 확인할 수 있는 웹앱을 구현하고자한다.

## 2. 관련 연구

### 2.1 QR Code

QR코드란 흑백 격자무늬 형태로 정보를 나타내는 매트릭스 형식의 이차원 코드로 2000년에 ISO/IEC 18004 국제 규격으로 인정되었다.

QR코드는 바코드의 수십 배 내지 수백 배 정도의 대용량 정보 수납이 가능하고, 모양 또는 기능에 따라 Micro QR코드, iQR코드, Frame QR코드, SQRC로 분류된다. QR코드는 오류 복원 기능을 통하여 오염 및 손상이 있더라도 최대 30%까지 복원이 가능하고, QR 코드 내부의 3개의 네모 격자(Position Required Patterns)로 어느 방향에서든 인식이 가능하다. QR코드는 특정 정보를 카메라를 가진 다른 컴퓨터로 광학적으로 전달할 수 있는 매우 유용한 수단이다. 웹사이트 주소 등 고정된 정보를 전달하는 수단뿐만 아니라 제로페이 등의 지불수단, 로그인 수단 등으로 최근 용도가 대폭 확대되고 있으며 최근 우리에게 가장 흔한 QR코드는 QR체크인이라 생각된다. 최근 코로나 19 사태로 인하여 어느 곳을 방문하더라도 수기 작성대신 QR체크인을 많이 사용하는 추세이며 스캔을 진행해야 실내로 입장이 가능하다.

QR코드의 사양은 아래의 표와 같다.

[표 1] QR코드 사양

코드크기	21×21cell ~ 177×177cell ( 4cell/변 씩 증가 )	
정보 종류 및 정보량	숫자	최대 7,089문자
	영숫자	최대 4,296문자
	8bit byte	최대 2,953문자
	영자	최대 1,817문자
오류복원 능력 (데이터 복원기능)	Level L	약 7% 복원가능
	Level M	약 15% 복원가능
	Level Q	약 25% 복원가능
	Level H	약 30% 복원가능
코드 연결 기능	최대 16분할 ( 좁고 긴 공간에 인쇄 )	

### 2.2 전자서명

법률 제 17354호 전자서명법의 제2조 정의에는 '전자서명이란 서명자가 해당 전자문서에 서명하였음을 나타내기 위해 전자문서에 첨부되거나 논리적으로 결합된 전자적 형태의 정보를 말한다'라고 명시되어있으며, 공개키 알고리즘을 이용하여 데이터에 대한 무결성을 보장하고, 인증과 부인방지를 도모하는 기술을 말한다.

### 2.3 X.509 인증서

인증서란 개인이 사용하는 공개키를 인증기관이 인증해주는 전자문서이며, X.509 인증

서는 개인정보와 공개키를 결합하여 인증기관이 서명한 문서로서 해당 공개키를 해당 개인이 소유하고 있음을 증명하며 인증서를 이용한 전자서명은 거래의 부인방지 기능을 제공한다. 공개키기반구조(PKI)는 X.509 인증서의 신뢰를 제공한다. 최근 공인인증서의 폐지에 따라 사설인증서가 다양하게 사용될 것으로 예상된다.

Version Number	
Serial Number	
Signature Algorithm ID	
Issuer Name	
Validity Period	Not Before
	Not After
Subject Name	
Subject Public Key Info	
Public Key Algorithm	
Subject Public Key	
Issuer Unique Identifier (optional)	
Subject Unique Identifier (optional)	
Extensions (optional)	
Certificate Signature Algorithm	
Certificate Signature	

[ 그림 2-1. X.509 인증서의 구조 ]

## 2.4 Node.js

Node.js는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 논블로킹(Non-blocking)[3] I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다.

내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능케 한다.

## 2.5 Angular

Angular는 Typescript를 기반으로 개발된 개발 플랫폼으로 확장 가능한 컴포넌트 구조로 웹 애플리케이션을 만드는 프레임워크이다. 라우팅, 폼 관리, 클라이언트-서버 통신 등 웹 개발에 필요한 라이브러리를 조화롭게 통합한 모음집이며, 애플리케이션 개발, 빌드, 테스트, 수정에 필요한 개발자 도구를 제공한다.

## 2.6 Ionic

Ionic Framework는 하이브리드 개발 Framework로 HTML, CSS 및 JavaScript와 같은 웹 기술을 사용하여 Angular, React 및 Vue와 같은 프레임워크에 대한 통합을 사용하여 고성능 고품질의 모바일 및 데스크톱 앱을 빌드하기 위한 오픈 소스 UI 툴킷이다. 하이브리드 웹앱 개발 방식이란 한 번의 개발로 웹과 모바일 앱을 제공하는 개발방식을 말한다.

## 2.7 Firebase

Firebase 인증은 사용자 인증 시 필요한 백엔드 서비스와 사용하기 쉬운 SDK, 기성 UI 라이브러리를 제공한다. 사용자를 로그인시키려면 우선 Firebase 인증 SDK는 사용자로부터 이메일과 비밀번호 또는 제휴 ID 공급업체에서 받은 인증토큰을 사용자 인증 정보로 받고 Firebase의 백엔드 서비스가 정보를 확인하여 클라이언트에 응답을 반환한다. 이러한 인증을 통해 사용자를 식별한다.

Firebase에서 제공하는 데이터베이스의 보안 규칙은 기본 규칙, 개발 환경 규칙, 프로덕션에 즉시 사용 가능한 규칙으로 3가지 규칙이 있고 세분화 하면 총 7가지 규칙이 있다. 보안 규칙은 인증으로 식별된 사용자가 데이터베이스에 접근할 수 있는 데이터를 정의한다. 따라서 악의적인 사용자로부터 데이터를 보호한다.

Firebase에서는 Firestore 실시간 Database, Hosting을 제공한다.

## 2.8 Node-forge

Node-forge는 npm에서 제공하는 모듈로 Forge 소프트웨어는 JavaScript의 TLS 프로토콜, 암호화 유틸리티 세트 및 많은 네트워크 리소스를 활용하는 웹 앱 개발을 위한 모듈이다.

## 2.9 JavaScript

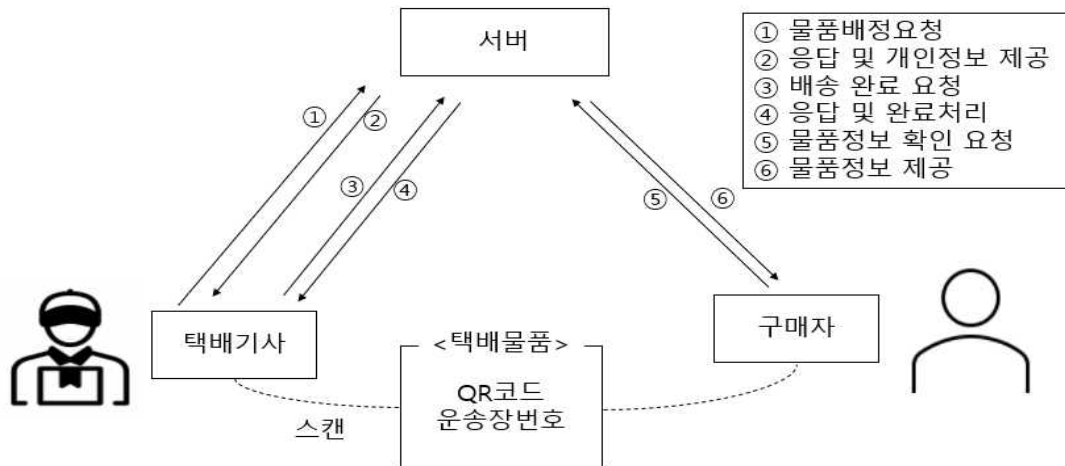
Javascript는 1995년 미국의 넷스케이프 커뮤니케이션즈에서 처음 개발되었으며, 웹 페이지에서 사용자로부터 특정 이벤트나 입력 값을 받아 동적인 처리를 목적으로 고안된 객체 기반의 스크립트 프로그래밍 언어이다. 주로 웹 브라우저 내에서 사용되는 언어였으나, 자바스크립트 기반의 런타임 플랫폼 (Node.js 등...)들이 개발되면서 서버측 프로그램 개발에도 사용이 크게 확대되었다.

## 2.10 Typescript

TypeScript는 2014년 마이크로소프트에서 개발되었으며, JavaScript, AngularJS, ReactJS, Node.js 등 JavaScript 기반 프로그램은 TypeScript 프로그램으로 동작하는 웹 클라이언트와 서버 프로그램 개발에 모두 사용되는 오픈 소스 프로그래밍 언어이다. 2017년 구글의 공식 개발 언어로 채택되어, 구글의 클라이언트 개발 환경에서 사용이 가능하며, 웹 어플리케이션 개발 프레임워크인 AngularJS 개발에 TypeScript가 사용되었다.

### 3. 본론

#### 3.1 시스템 설계



[ 그림 3-1. 서버, 택배기사, 구매자 시스템 설계 ]

- ① 공개키와 개인키가 발급되어있는 택배기사가 물품을 배정 받기 위해 QR코드 운송장을 스캔하여 전자서명을 생성한다. 전자서명에는 택배기사의 정보, 배정 시간, 운송장 번호가 포함되어있다.
- ② 택배기사가 자신에게 배정된 구매자를 확인하기 위해 서버에 정보를 요청한다.
- ③ 택배기사가 배송 완료 후 서버에 완료처리를 요청하기 위해 전자서명을 생성한다. 전자서명에는 택배기사의 정보와 운송장 번호, 배달 완료 시간을 포함하고 있다. 서버는 생성된 전자서명을 이용하여 서명 검증 후 완료처리를 한다.
- ④ 택배기사의 정보와 배송 상태 및 배송 현황을 데이터베이스에 업데이트하여 관리자 페이지 및 고객 웹앱에서 해당 정보를 확인 할 수 있다.
- ⑤ 택배가 도착하게 되면 구매자는 웹앱을 통해 택배에 부착된 QR코드를 스캔하여 해당 QR코드의 값의 필드 값 정보를 요청한다.
- ⑥ 해당 고객이 주문을 한 것이 확인이 되면 스캔 값으로 QR코드의 필드 값이 출력되지만 만약, 자신의 택배가 아닌 경우 스캔 값이 표시되지 않는다.



### 3.2 DB 자료구조 설계

본 프로젝트에서는 google의 Firebase-Firestore를 사용하였으며, Firestore는 NoSQL Database이며, 크게 Collection, Document, Field 또는 세부 Collection으로 구성된다. 데이터베이스 구성은 다음과 같다.

<a href="#">🏠</a> > <a href="#">products</a> > <a href="#">Ko3E0BMPozfrs...</a>	<a href="#">🏠</a> > <a href="#">products</a> > <a href="#">Ko3E0BMPozfrs...</a>	<a href="#">🏠</a> > <a href="#">products</a> > <a href="#">Ko3E0BMPozfrs...</a>
<div>capstone2-178ba</div> <div>+ 컬렉션 시작</div> <div> <a href="#">Delivery</a>  <a href="#">carts</a>  <a href="#">orders</a>  <b>products</b> &gt;  <a href="#">profile</a> </div>	<div>products</div> <div>+ 문서 추가</div> <div> <div>Ko3E0BMPozfrsoxqAhMi &gt;</div> <div>hHXB7BxHsSEm6Qk0vjN</div> <div>qmWDJN5q0ziQbLqcJiTG</div> <div>qyqaF9QJvayuesIs94Ge</div> </div>	<div>Ko3E0BMPozfrsoxqAhMi</div> <div>+ 컬렉션 시작</div> <div>+ 필드 추가</div> <div> <div>description: "화제의 그 신작! 겹겹이 쌓인 크루아상을 경험해보세요!"</div> <div>i1: "https://ifh.cc/g/WnkR3f.jpg"</div> <div>i2: "https://ifh.cc/g/AhNm0k.jpg"</div> <div>image: "https://ifh.cc/g/4EhdPg.jpg"</div> <div>price: "2,200원"</div> <div>title: "크루아상 (Croissant)"</div> </div>

[ 그림 3-2 products Collection ]

products collection은 고객 전용 웹앱에서 주문 부분의 상품 내용의 정보들이 포함되어 있다. products 의 document는 하나의 상품을 나타내고 있으며 필드 값은 description은 상세설명, image, i1, i2는 고객 웹앱에서 확인 할 수 있는 이미지의 주소, price는 해당 collection에 해당하는 제품의 가격, title은 제품의 이름을 뜻한다.

<a href="#">🏠</a> > <a href="#">Delivery</a> > <a href="#">qQojIqyFvNv3Q...</a>	<a href="#">🏠</a> > <a href="#">Delivery</a> > <a href="#">qQojIqyFvNv3Q...</a>	<a href="#">🏠</a> > <a href="#">Delivery</a> > <a href="#">qQojIqyFvNv3Q...</a>
<div>capstone2-178ba</div> <div>+ 컬렉션 시작</div> <div> <a href="#">Delivery</a> &gt;  <a href="#">carts</a>  <a href="#">orders</a>  <a href="#">products</a>  <a href="#">profile</a> </div>	<div>Delivery</div> <div>+ 문서 추가</div> <div> <div>DL2W4erjrqZrwCScY2jr8hXNINA2</div> <div>qQojIqyFvNv3Q0xptpd91R9FI133 &gt;</div> </div>	<div>qQojIqyFvNv3Q0xptpd91R9FI133</div> <div>+ 컬렉션 시작</div> <div>D_orders</div> <div>+ 필드 추가</div> <div> <div>email: "watzzo@test.com"</div> <div>name: "이왓조"</div> <div>phonenumber: "010-9876-5432"</div> </div>

[ 그림 3-3 Delivery Collection ]

Delivery Collection은 회원가입을 진행한 기사의 정보가 들어가 있으며, Delivery의 Document.id 값이 해당 계정의 uid값을 의미한다. 필드 값에는 회원가입이 진행된 계정(email), 기사님 성함, 전화번호 등이 저장된다. 기사가 여러명인 점을 감안하여 계정별로 document\_id를 지정하였고, 이에 따라 세부 collection인 D\_orders를 생성하였다.

<a href="#">🏠</a> > <a href="#">Delivery</a> > <a href="#">qQoJlqyFvNv3QOxtpd91R9FI33</a> > <a href="#">D_orders</a> > <a href="#">ZfwHJ2DMvRO...</a>		
<div>qQoJlqyFvNv3QOxtpd91R9FI33</div> <div>+ 컬렉션 시작</div> <div>D_orders &gt;</div> <div>+ 필드 추가</div> <div>email: "watzzo@test.com"</div> <div>name: "이왓조"</div> <div>phonenummer: "010-9876-5432"</div>	<div>D_orders</div> <div>+ 문서 추가</div> <div>ZfwHJ2DMvROJAiHeG5Dv &gt;</div>	<div>ZfwHJ2DMvROJAiHeG5Dv</div> <div>+ 컬렉션 시작</div> <div>+ 필드 추가</div> <div>Ko3E08MPozfrsoxqAhMi: 4</div> <div>address: "경기 고양시 덕양구 대자동 613-1"</div> <div>d_number: "41281_225"</div> <div>deliverystart: 2021년 10월 5일 오후 1시 53분 59초 UTC+9</div> <div>deliverysuccess: 2021년 10월 5일 오후 1시 51분 36초 UTC+9</div> <div>detailaddress: "세종관 8층"</div> <div>dname: "이왓조"</div> <div>dstate: "배송시작"</div>

[ 그림 3-4 Delivery Collection의 세부 Collection D\_orders ]

Delivery Collection 내부의 세부 Collection인 D\_orders에는 해당 기사에게 배정되어있는 택배 정보가 저장된다. 기사에 따라 D\_orders가 각각 생성되어있기 때문에 기사 별로 자신이 어느 택배를 배정 받았는지 쉽게 알 수 있다.

<a href="#">🏠</a> > <a href="#">carts</a> > <a href="#">dMz7s8MMJhP...</a>		
<div>capstone2-178ba</div> <div>+ 컬렉션 시작</div> <div>Delivery</div> <div><b>carts</b> &gt;</div> <div>orders</div> <div>products</div> <div>profile</div>	<div>carts</div> <div>+ 문서 추가</div> <div>22RpNb2aChmtXtBN37gn</div> <div>3mFFQhQ2k3W6RQj0qWU7</div> <div>BGZIsFUBfULRpCjNViV</div> <div>EawZyn79xkojzcbnheqD</div> <div>GQ655A57Qe9zi3rXysUY</div> <div>MhdcghcRQPnD2WupO58G</div> <div>OWrITiwpGZfcM9ckNIYo</div> <div>VHneZu0xQXpD7akXHhow</div> <div><b>dMz7s8MMJhPm46yAsCTn</b> &gt;</div> <div>hKZvU81mPk6JYSmRb5aS</div> <div>nAKDFk3VnMqOIWHGR9V4</div> <div>number</div> <div>oQXNbqaQQ4eGqUL5BS15</div>	<div>dMz7s8MMJhPm46yAsCTn</div> <div>+ 컬렉션 시작</div> <div>+ 필드 추가</div> <div>address: "경기 고양시 덕양구 대자동 613-1"</div> <div>d_number: "41281_227"</div> <div>deliverystart: 2021년 10월 5일 오후 2시 26분 49초 UTC+9</div> <div>deliverysuccess: 2021년 10월 5일 오후 2시 26분 49초 UTC+9</div> <div>detailaddress: "세종관"</div> <div>dname: "미지정"</div> <div>dstate: "주문완료"</div> <div>hXB7BxHsSEm6Qk0vjN: 4</div> <div>lastUpdate: 2021년 10월 5일 오후 2시 26분 49초 UTC+9</div> <div>name: "김준식"</div> <div>ordercomplete: 2021년 10월 5일 오후 2시 26분 49초 UTC+9</div> <div>phonenummer: "010-0000-0000"</div>

[ 그림 3-5 carts Collection ]

carts 컬렉션의 document\_id 값은 localhost 저장소 값으로 설정되어있으며, 임시 저장소 같은 개념으로 사용된다. 예를 들어 실제 마트라고 생각해보자, 카트에 물건을 집어넣은 후 계산 후에는 물건들이 자신의 장바구니에 들어가고 카트는 비어있듯이 고객이 주문완료 버튼을 누르게 되면 해당 carts내부의 필드 값들은 orders, 그리고 profile collection의 cart에 저장되며 이후 carts collection에는 lastUpdate 값만 존재한다.

capstone2-178ba	cards	number
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
Delivery	22RpNb2aChmtXtBN37gn	+ 필드 추가
<b>cards</b> >	3mFFQhQ2k3W6RQj0qWU7	number: 228
orders	BGZIBsFUBfULRpCjNViV	
products	EawZyn79xkojzcbnheqD	
profile	GQ655A57Qe9zi3rXysUY	
	MhdcghcRQPnD2WupO58G	
	OWrITiwpGZfcM9ckNIYo	
	VHneZu0xQXpD7akXHhow	
	dMz7s8MMJhPm46yAsCTn	
	hKZvU81mPk6JYSmRb5aS	
	nAKDFk3VnMq0IWHGR9V4	
	<b>number</b> >	
	oQXNbqaQQ4eGqUL5BS15	

[ 그림 3-6 cards Collection number document ]

cards Collection의 number부분은 order로 넘어가는 값을 하나씩 카운트하여 택배 코드에 중복현상 예방을 위하여 포함된다.

capstone2-178ba	orders	71d6N18Ritnwmf6SXu2h
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
Delivery	71d6N18Ritnwmf6SXu2h >	+ 필드 추가
cards	XXdav0oZU7mDZNAG2s0L	address: "경기 고양시 덕양구 대자동 613-1"
<b>orders</b> >	ZfwHJ2DMvROJAiHeG5Dv	d_number: "41281_227"
products		deliverystart: 2021년 10월 5일 오후 2시 26분 49초 UTC+9
profile		deliverysuccess: 2021년 10월 5일 오후 2시 26분 49초 UTC+9
		detailaddress: "세종관"
		dname: "미지정"
		dstate: "주문완료"
		hHXB7BxHsSEMn6Qk0vjN: 4
		lastUpdate: 2021년 10월 5일 오후 2시 26분 49초 UTC+9
		name: "김춘식"
		ordercomplete: 2021년 10월 5일 오후 2시 26분 49초 UTC+9
		phonenummer: "010-0000-0000"

[ 그림 3-7 orders Collection ]

orders Collection은 cards Collection에 있는 필드 값들이 orders에 저장된 값이다. cards Collection의 필드와 다른 점은 d\_number인 택배 코드가 추가된 점이다. 41281은 입력된 주소 값의 시군구 코드로 해당 그림에서는 '경기 고양시 덕양구'까지 코드를 41281로 표기한다. 해당 값과 cards의 number값을 합한 것이 택배코드인 d\_number이다.

<a href="#">🏠</a> > <a href="#">profile</a> > 928VHywUSxO... <a href="#">✎</a>		
<div>capstone2-178ba</div> <div>+ 컬렉션 시작</div> <div>Delivery</div> <div>carts</div> <div>orders</div> <div>products</div> <div>profile &gt;</div>	<div>profile</div> <div>+ 문서 추가</div> <div>7SdG4doq6ZQu7nUaMM5xefC6zsS2</div> <div>928VHywUSx0Qji5xcYeyqPvic112 &gt;</div>	<div>928VHywUSx0Qji5xcYeyqPvic112</div> <div>+ 컬렉션 시작</div> <div>cart</div> <div>+ 필드 추가</div> <div>address: "경기 고양시 덕양구 대자동 613-1"</div> <div>detailaddress: "세종관"</div> <div>email: "aa@naver.com"</div> <div>name: "김춘식"</div> <div>phonenummer: "010-0000-0000"</div> <div>si: "41281"</div>

[ 그림 3-8 profile Collection ]

profile Collection은 앞에서 말한 Delivery Collection과 유사한 형태로 구성되어있다. profile Collection의 document값은 고객 계정의 uid이며 필드 값은 고객의 정보로 구성되어있다. 해당 그림은 고객이 기본 배송지를 입력한 경우이고, 만약 기본 배송지가 없다면 email과 name만 필드에 존재하게 된다.

<a href="#">🏠</a> > <a href="#">profile</a> > 928VHywUSxO... > <a href="#">cart</a> > ZfwHJ2DMvRO... <a href="#">✎</a>		
<div>928VHywUSx0Qji5xcYeyqPvic112</div> <div>+ 컬렉션 시작</div> <div>cart &gt;</div> <div>+ 필드 추가</div> <div>address: "경기 고양시 덕양구 대자동 613-1"</div> <div>detailaddress: "세종관"</div> <div>email: "aa@naver.com"</div> <div>name: "김춘식"</div> <div>phonenummer: "010-0000-0000"</div> <div>si: "41281"</div>	<div>cart</div> <div>+ 문서 추가</div> <div>71d6N18Ritnwmf6SXu2h</div> <div>XXdav0oZU7mDZNAG2s01</div> <div>ZfwHJ2DMvROJAiHeG5Dv &gt;</div>	<div>ZfwHJ2DMvROJAiHeG5Dv</div> <div>+ 컬렉션 시작</div> <div>+ 필드 추가</div> <div>Ko3E0BMPozfrsoxqAhMi: 4</div> <div>address: "경기 고양시 덕양구 대자동 613-1"</div> <div>d_number: "41281_225"</div> <div>deliverystart: 2021년 10월 5일 오후 1시 53분 59초 UTC+9</div> <div>deliversuccess: 2021년 10월 5일 오후 1시 51분 36초 UTC+9</div> <div>detailaddress: "세종관 8층"</div> <div>dname: "이왔조"</div> <div>dstate: "배송시작"</div> <div>hHXB7BxHsSEMn6Qk0vjN: 4</div> <div>lastUpdate: 2021년 10월 5일 오후 1시 53분 59초 UTC+9</div> <div>name: "김춘식"</div> <div>ordercomplete: 2021년 10월 5일 오후 1시 51분 36초 UTC+9</div>

[ 그림 3-9 profile Collection의 세부 cart Collection ]

profile Collection 내부의 세부 Collection인 cart를 살펴보면 해당 고객이 주문 완료한 값들이 저장되어있는 것을 확인할 수 있다.

address: "경기 고양시 덕양구 대자동 613-1"  
d\_number: "41281\_227"  
deliverystart: 2021년 10월 5일 오후 2시 35분 4초 UTC+9  
deliverysuccess: 2021년 10월 5일 오후 2시 35분 20초 UTC+9  
detailaddress: "세종관"  
dname: "이왔조"  
dstate: "배송완료"  
hXB7BxHsSEMn6Qk0vjN: 4  
lastUpdate: 2021년 10월 5일 오후 2시 35분 20초 UTC+9  
name: "김춘식"  
ordercomplete: 2021년 10월 5일 오후 2시 26분 49초 UTC+9  
phonenummer: "010-0000-0000"  
si: "41281"  
사업자: "택배왔조"  
사업자번호: "AA"

[ 그림 3-10 고객의 배송정보 필드 값 ]

다음은 carts, orders, Delivery/d\_orders, profile/cart에 들어가는 필드 값이다. 절차에 따라 특정 필드 값들은 들어가지 않는 경우가 있지만, 가장 많은 필드 값들이 포함되어있는 profile Collection 특정 uid document의 cart collection의 하나의 주문리스트에 대한 필드 값이다.

해당 그림의 필드 값은 기사가 배송을 완료한 상태인 것을 감안하고 확인하게 되면, address는 고객이 주문 시에 작성한 주소 값이고, detailaddress는 배송지의 상세 주소이다. d\_number는 위에서 언급한 것과 같이 시군구 코드(해당 그림에서는 si필드에 해당된다.)와 cart의 number값이 결합된 값이다. name은 배달받을 고객의 이름이고, ordercomplete는 고객이 주문하기를 누른 시점의 시간이고, delivertstart의 값은 기사가 배정되고 배달을 시작한 시간이며, deliverysuccess는 기사가 배달완료 처리를 진행한 시간이다. dname 필드는 해당 택배에 배정된 기사의 이름이고, dstate는 해당 택배의 상태를 나타낸다.

### 3.3 프로그램 구성 및 구성원 별 상세 내용

#### 3.3.1 고객

고객은 사이트에 접속하여 회원가입을 진행한 뒤 로그인한다. 고객은 해당 사이트에서 기본 배송지 추가 및 수정이 가능하며 원하는 항목을 주문하고 주문 항목들을 확인할 수 있다. 배송 올 택배 물품을 확인하기 위해 서버에 요청하여 자신의 정보와 서버에 존재하는 정보와 확인한 뒤 정보를 제공받는다. 제공받은 정보에는 받는 사람, 보내는 사람의 정보와 택배 배송현황 및 배송할 택배기사의 정보까지 확인할 수 있다.

#### 3.3.2 택배기사

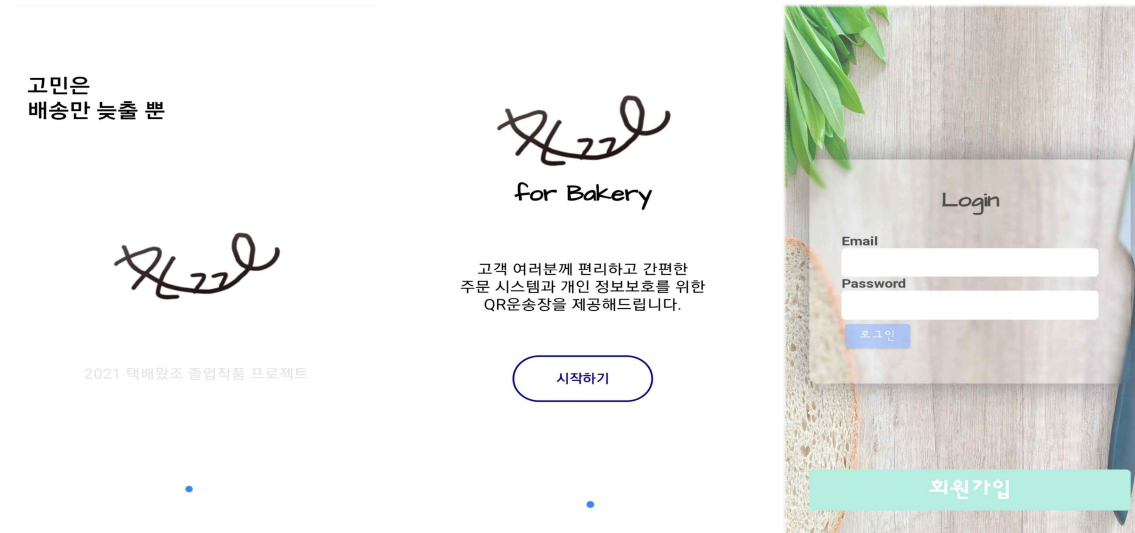
택배기사가 택배회사에 소속이 되어 있다고 가정하자. 택배기사는 관리자에서 생성한 계정을 통하여 로그인이 가능하며, 택배기사가 회사에 소속될 때 택배기사의 정보가 서버에 등록된다. 택배기사는 최초 로그인 시 서버에서 공개키와 개인키를 발급받는다. 택배를 배정받기 위하여 QR스캔기능을 사용하여 택배 물품에 부착된 QR코드를 스캔한다. 스캔한 QR코드 안에는 식별번호가 존재한다. 데이터베이스에서 식별번호를 가진 물품 정보에 택배기사의 정보를 추가하여 택배기사를 배정해주며, 택배기사의 정보를 갖고 있는 택배 물품을 모두 가져올 수 있다. 또한 택배기사의 가시성을 높이기 위해 지도에 구매자의 주소를 마커로 표시하는 위치기반 서비스를 제공한다.

#### 3.3.3 관리자

관리자는 관리자 전용 페이지에서 택배 기사 계정을 생성할 수 있으며, 기사 리스트 정보에서 생성된 기사 목록을 확인할 수 있다. 주문리스트에서는 고객의 이름, 전화번호, 주소, 주문목록, QR운송장, 배정기사, 배송상태, 배송 현황 등의 정보가 제공되며 QR운송장을 일괄 인쇄할 수 있는 기능을 제공한다.

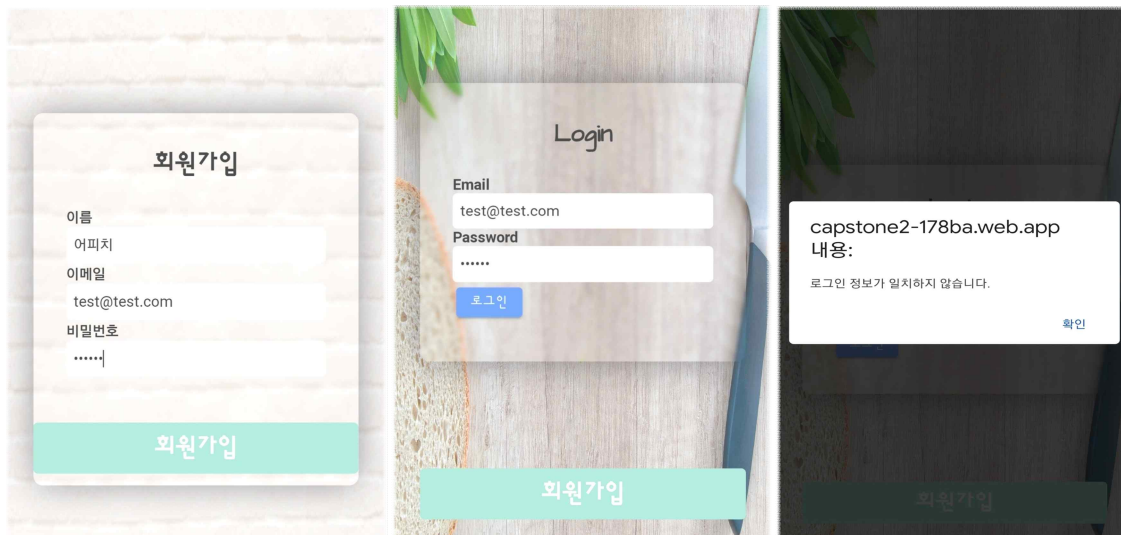
### 3.4 기능 별 상세 내용 데모

#### 3.4.1 고객 회원가입, 로그인, 기본 배송지 지정



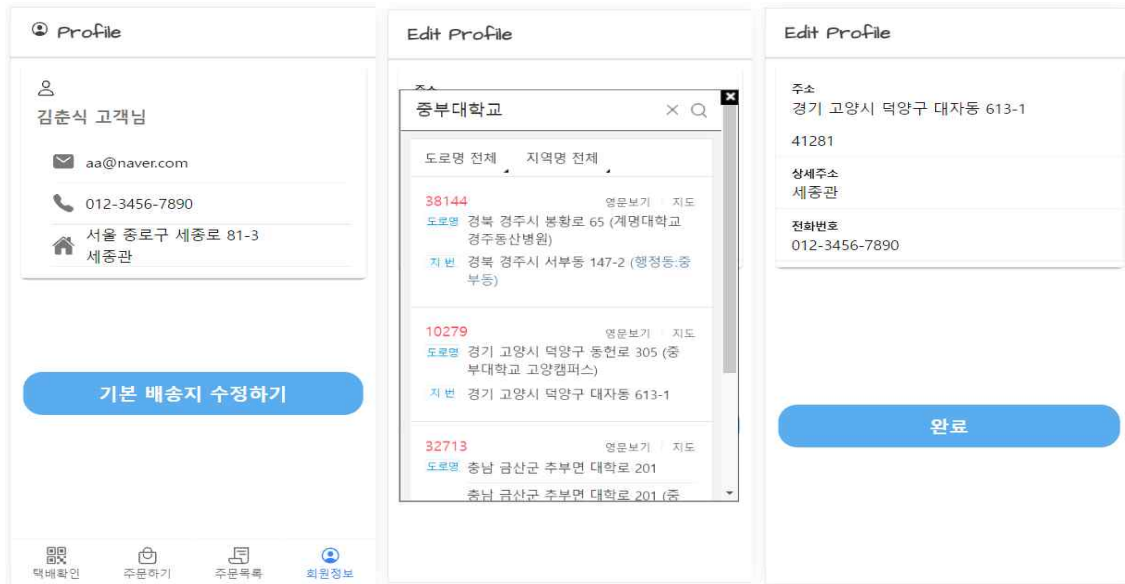
[ 그림 3-11 고객 사이트 첫 페이지 및 로그인 페이지 ]

고객 홈페이지 접속 시 첫 화면에 wattzo 로고와 해당 서비스에 대한 간단한 설명 및 로그인, 회원가입 페이지가 표시된다.



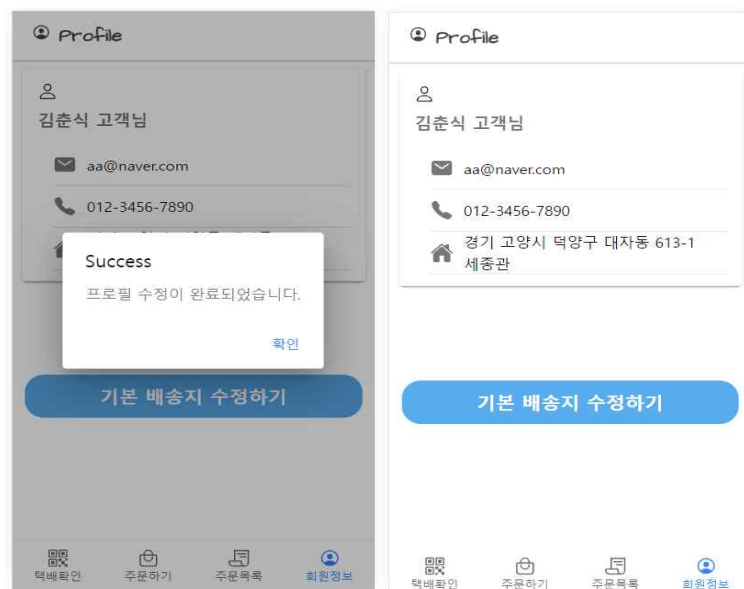
[ 그림 3-12 회원가입 화면, 로그인 불일치 알림 ]

이름, 이메일, 비밀번호로 회원가입 후에 로그인을 진행한다. 만약 로그인 정보가 일치하지 않는다면 [그림 3-12]의 우측 그림과 같이 '로그인 정보가 일치하지 않습니다.' 알림이 표시된다.



[ 그림 3-13 회원정보 및 기본 배송지 수정전 ]

기본 배송지를 추가 또는 수정하고 싶은 경우에는 하단 탭의 회원정보에서 기본 배송지 수정하기를 클릭 후 수정하고 싶은 주소, 상세주소, 전화번호 등을 입력한다.

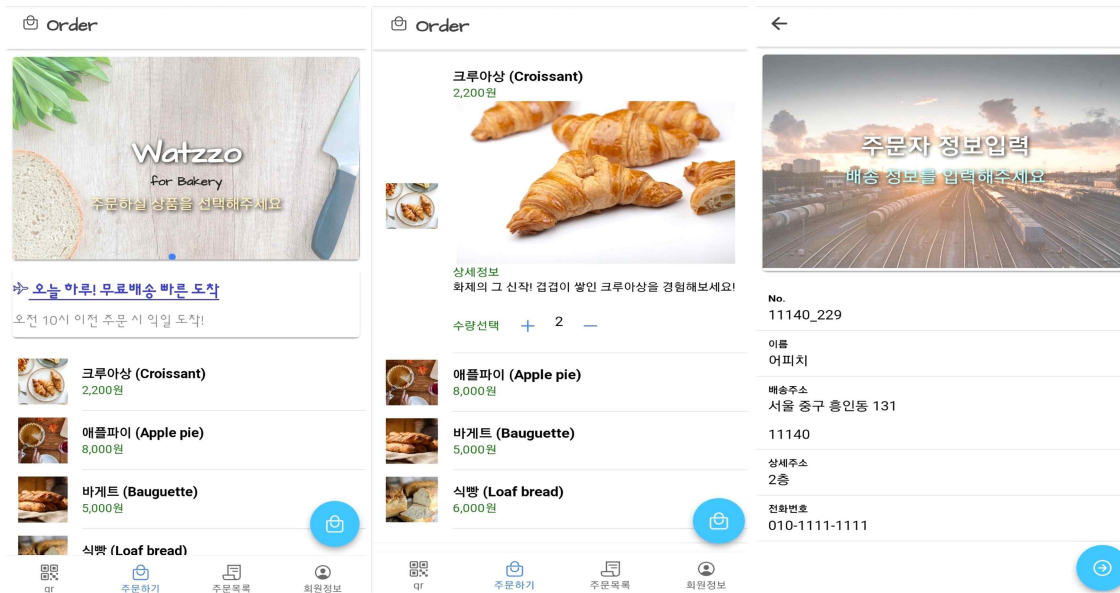


[ 그림 3-14 기본 배송지 수정후 회원정보 ]

원하는 기본 배송지로 수정을 진행하고 완료 버튼을 클릭하면 '프로필 수정이 완료되었습니다.' 알림과 함께 회원정보 페이지로 이동된다. 회원 정보 페이지에서는 수정된 전화번호, 주소 값을 확인할 수 있다. 이후 기본 배송지는 주문단계에서 default 값으로 입력되어 보다 편하게 주문기능을 이용할 수 있다.



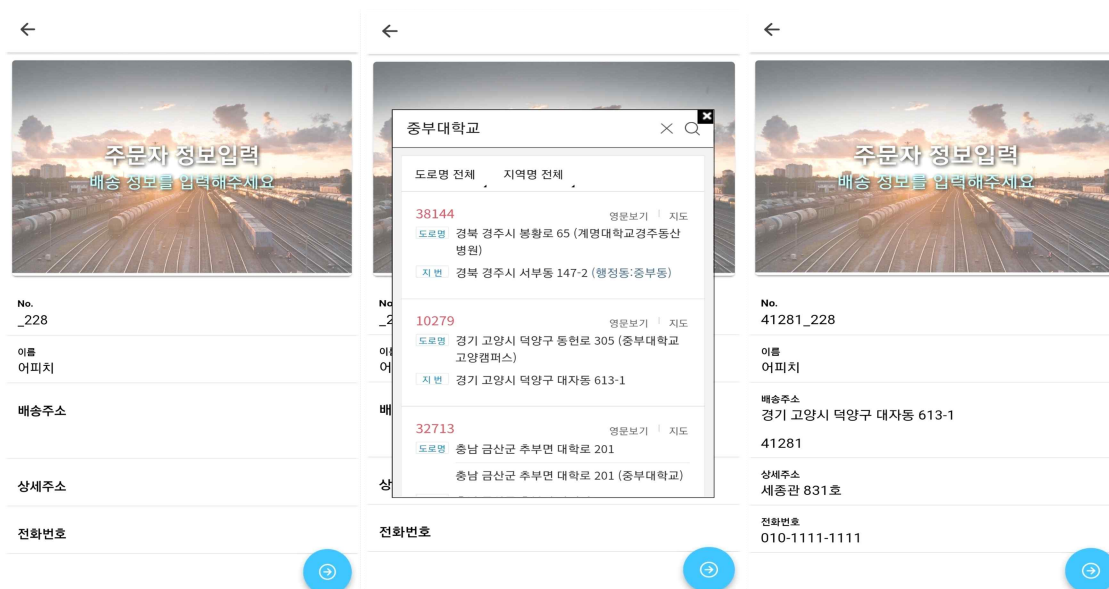
### 3.4.2 고객 주문단계



[ 그림 3-15 고객 주문, 주문자 정보입력 기본배송지가 기본값으로 들어간 모습 ]

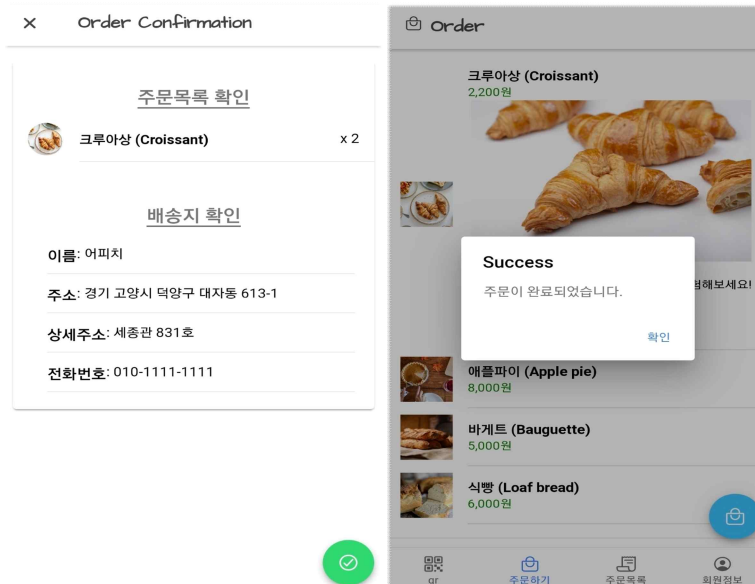
아래 탭에서 '주문하기'로 이동하면 다음 그림과 같이 주문할 수 있는 목록들이 표시된다. 해당 목록을 클릭하게 되면 관련 이미지들과 상세정보를 확인 할 수 있으며 수량을 선택하여 우측 하단의 바구니 아이콘을 클릭하여 주문을 진행할 수 있다.

만약 사용자의 기본 배송지가 저장되어있는 경우 별도로 입력하지 않아도 배송주소, 상세주소, 전화번호 등이 기본값으로 입력되어있다.



[ 그림 3-16 기본 배송지가 없는 경우 ]

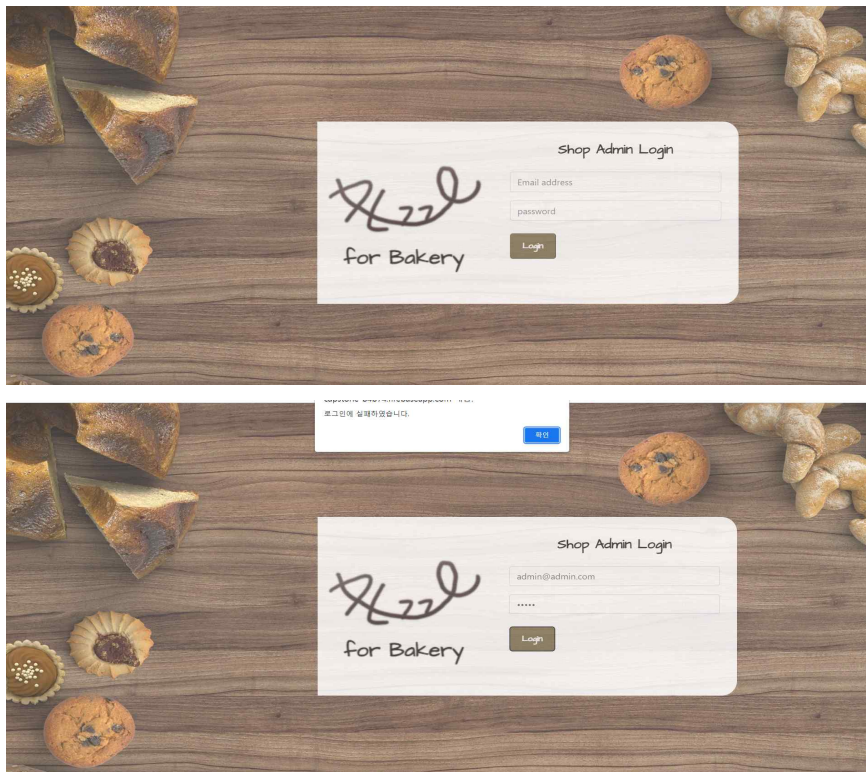
기본 배송지가 저장되어있지 않은 경우 직접 작성을 진행할 수 있다. 만약 기본 배송지가 아닌 다른 곳으로 배송하고 싶다면 클릭 후에 수정이 가능하다. 이후 우측 하단의 진행 버튼을 클릭하면 주문목록과 배송지를 확인 할 수 있는 페이지가 표시된다.



[ 그림 3-17 주문목록, 배송지 확인 및 주문완료 ]

주문 완료 직전 주문목록과 배송지를 확인 할 수 있으며 우측 하단의 완료 표시를 클릭하게 되면 '주문이 완료되었습니다' 알림을 확인 할 수 있다. 목록들은 하단 탭의 주문 목록에서 확인이 가능하다.

### 3.4.3 관리자 로그인



[ 그림 3-18 관리자 페이지 로그인 화면 ]


관리자 페이지는 별도의 회원가입 없이 'admin@admin.com' 으로 로그인이 가능하다.

### 3.4.4 관리자 고객 주문 리스트 확인 및 QR 운송장 출력

Watzzo Home OrderList Delivery Link							
김준식	010-0000-0000	경기 고양시 덕양구 대자동 613-1 세종관	크루아상 (Croissant) : 애플파이 (Apple pie) : 4 바게트 (Baguette) : 식빵 (Loaf Bread) :	QR운송장	이왔조	배송완료	주문완료 (배송준비) : 2021/10/5 2:26 PM 배송시작 : 2021/10/5 2:35 PM 배송완료 : 2021/10/5 2:35 PM
김준식	010-0000-0000	서울 송파구 신천동 29 롯데월드	크루아상 (Croissant) : 2 애플파이 (Apple pie) : 2 바게트 (Baguette) : 식빵 (Loaf Bread) :	QR운송장	미지정	주문완료	주문완료 (배송준비) : 2021/10/5 2:21 PM 배송시작 : 2021/10/5 2:21 PM 배송완료 : 2021/10/5 2:21 PM
김준식	010-0000-0000	경기 고양시 덕양구 대자동 613-1 세종관 8층	크루아상 (Croissant) : 4 애플파이 (Apple pie) : 4 바게트 (Baguette) : 1 식빵 (Loaf Bread) :	QR운송장	이왔조	배송시작	주문완료 (배송준비) : 2021/10/5 1:51 PM 배송시작 : 2021/10/5 1:53 PM 배송완료 : 2021/10/5 1:51 PM
어피치	010-1111-1111	경기 고양시 덕양구 대자동 613-1 세종관 831호	크루아상 (Croissant) : 2 애플파이 (Apple pie) : 바게트 (Baguette) : 식빵 (Loaf Bread) :	QR운송장	이왔조	환불요청	주문완료 (배송준비) : 2021/10/5 3:28 PM 배송시작 : 2021/10/5 3:33 PM 배송완료 : 2021/10/5 3:33 PM

[ 그림 3-19 관리자 페이지 고객 주문리스트 ]

관리자 페이지에서는 고객이 주문한 리스트를 확인할 수 있으며, 배송상태에 따라 리스트의 색이 다르게 표시된다. 고객이 주문을 완료하는 경우에는 흰색, 기사가 운송장 QR 코드를 인식하여 택배를 배정받아 배송을 시작하는 경우 노란색, 기사가 배송을 완료하는 경우 파란색으로 표시되며, 배송이 완료되고 고객이 환불을 요청하는 경우 빨간색으로 표시된다.

Watzzo Home OrderList Delivery Link							
운송장인쇄							
이름	전화번호	주소	주문목록	운송장	배정 기사	배송 상태	배송 현황
김준식	010-0000-0000	경기 고양시 덕양구 대자동 613-1 세종관	크루아상 (Croissant) : 애플파이 (Apple pie) : 4 바게트 (Baguette) : 식빵 (Loaf Bread) :	QR운송장 	이왔조	배송완료	주문완료 (배송준비) : 2021/10/5 2:26 PM 배송시작 : 2021/10/5 2:35 PM 배송완료 : 2021/10/5 2:35 PM
어피치	010-1111-1111	서울 중구 동인동 131 2층	크루아상 (Croissant) : 2 애플파이 (Apple pie) : 2 바게트 (Baguette) : 2 식빵 (Loaf Bread) : 2	QR운송장	김당근	배송시작	주문완료 (배송준비) : 2021/10/5 5:05 PM 배송시작 : 2021/10/5 7:16 PM 배송완료 : 2021/10/5 5:05 PM

[ 그림 3-20 리스트 각각 QR 운송장 표시]

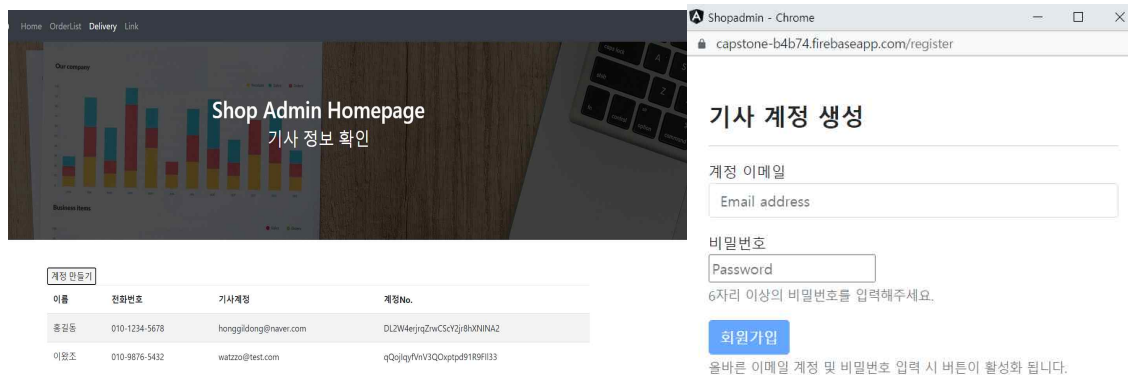
관리자 페이지의 리스트 중 QR운송장 버튼을 클릭하면 각각의 QR운송장을 확인할 수 있으며, 주문 리스트 화면의 우측 '운송장 인쇄' 버튼을 클릭하게 되면 인쇄 페이지가 생성된다.



[ 그림 3-21 QR운송장 인쇄 ]

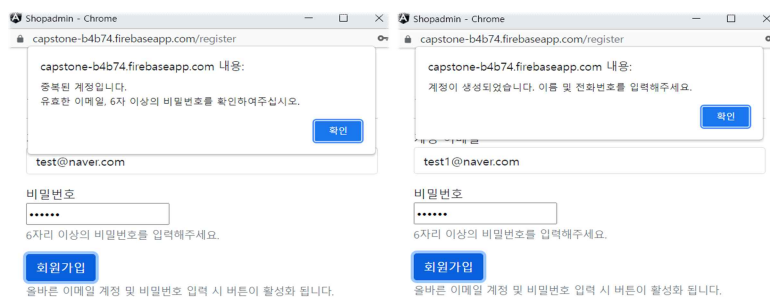
[그림 3-20]의 좌측 그림은 관리자 리스트 내에서 운송장 인쇄 버튼을 클릭하였을 때 표시되는 운송장 인쇄 페이지이며, 상단의 인쇄 버튼을 클릭하면 [그림 3-20] 우측 그림과 같이 인쇄가 가능하다.

### 3.4.5 관리자 택배기사 계정 생성



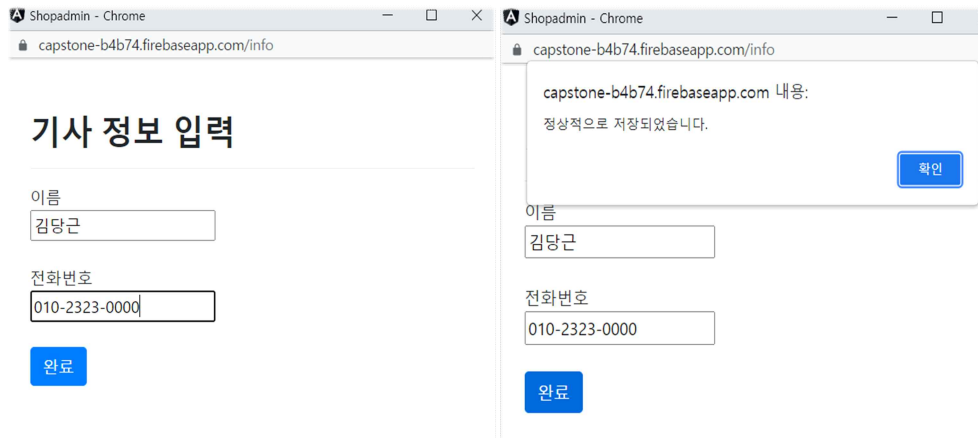
[ 그림 3-22 택배기사 리스트, 기사 계정 생성 ]

관리자 페이지에서는 [그림 3-21]의 좌측 그림처럼 생성되어있는 기사의 정보를 확인할 수 있다. 또한, 리스트 좌측 상단의 '계정 만들기' 버튼을 클릭하면 [그림 3-21]의 우측 그림처럼 기사 계정을 생성할 수 있다.



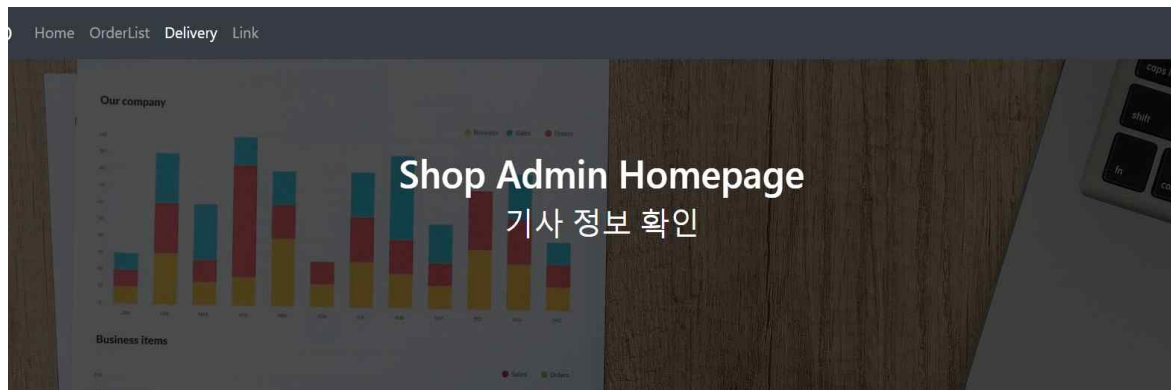
[ 그림 3-23 중복된 계정, 정상적인 계정 생성 ]

만약 중복된 계정의 경우 [그림 3-22]의 좌측 그림과 같이 중복된 계정 알림이 표시된다. 중복된 계정이 아닌 경우 정상적으로 계정이 생성된다.



[ 그림 3-24 기사 정보 입력 ]

[그림 3-22]의 계정 생성이 완료되면 기사 정보를 입력할 수 있으며, 이름과 전화번호를 입력하여 완료버튼을 눌러 계정 생성을 완료한다.



계정 만들기			
이름	전화번호	기사계정	계정No.
홍길동	010-1234-5678	honggildong@naver.com	DL2W4erjqZrwCScY2jr8hXNINA2
김당근	010-2323-0000	test1@naver.com	X0K6maLWPgPBytrfzIffM9CieQW2
이왓조	010-9876-5432	watzzo@test.com	qQoJlqyfVnV3QOxptpd91R9FI133

[ 그림 3-25 생성이 완료된 기사 계정 ]

기사 계정 생성이 완료되면 기사 정보 확인 페이지에서 리스트에 계정이 추가된 것을 확인할 수 있다.



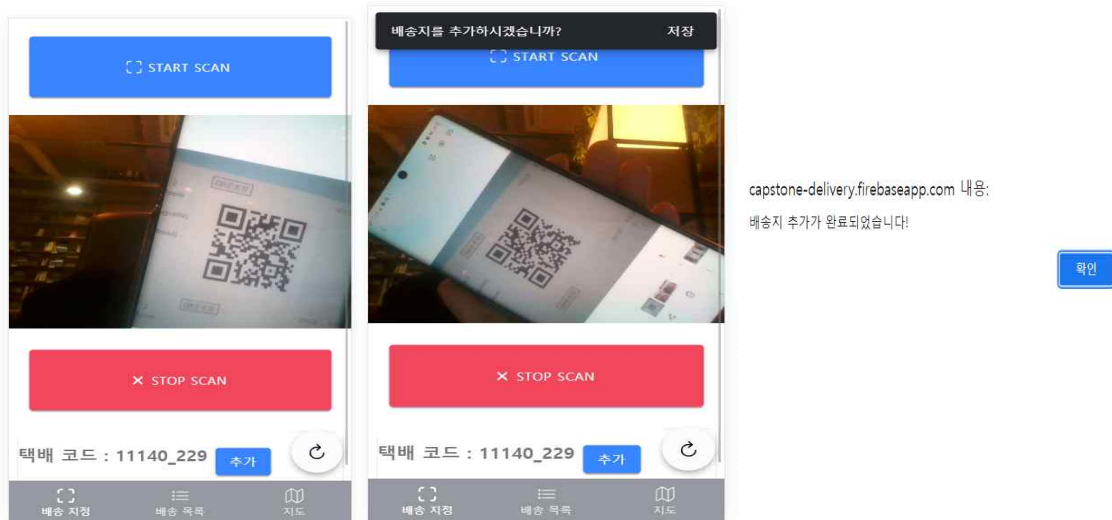
### 3.4.6 택배기사 로그인 및 인증서 발급



[ 그림 3-26 택배기사 로그인 및 배송목록 페이지 ]

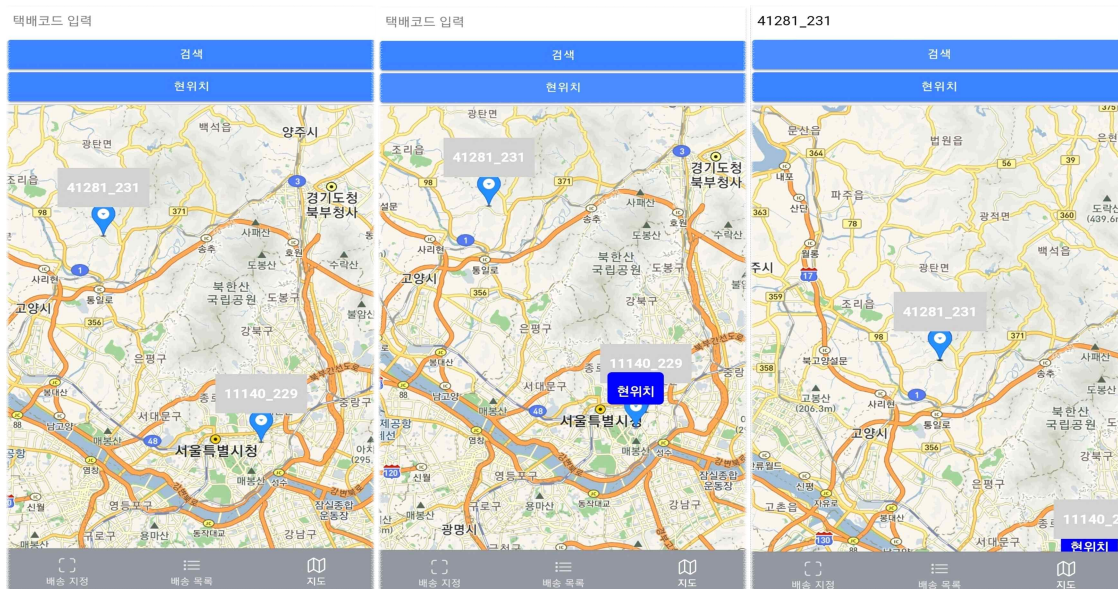
앞선 3.4.5 관리자 페이지에서 생성한 택배기사 계정으로 택배기사 전용 페이지에서 로그인을 진행한다. 로그인이 진행되면 기사가 인증서를 발급받았는지 확인한다. 발급을 안한 기사는 인증서를 발급받아 저장한다. 로그인이 정상적으로 실행되면 [그림 3-25] 우측 그림과 같이 배송목록 페이지가 표시된다.

### 3.4.7 택배기사 택배 배정 및 배송지 지도 표시



[ 그림 3-27 택배기사 택배 배정 ]

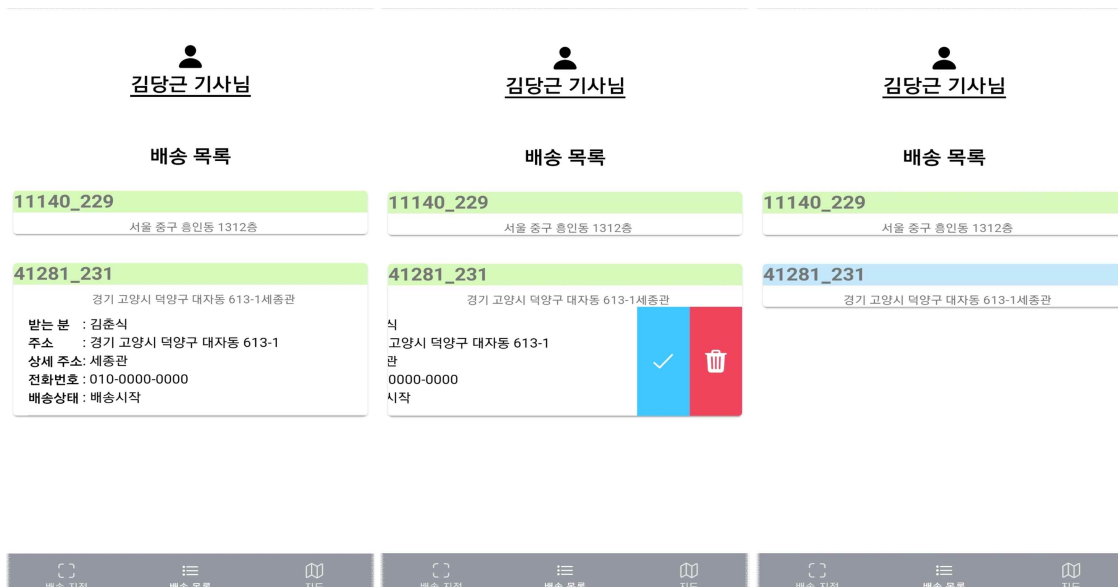
하단 탭의 '배송 지정' 클릭 후 카메라 권한을 부여한다. 권한을 부여하면 자동으로 택배를 배정받기 위한 QR스캐너가 실행된다. 이후 택배에 부착된 QR 운송장을 스캔하면 하단에 택배코드가 표시된다. 해당 택배 코드를 배정받으려면 '추가'버튼을 클릭하고 상단에 '배송지를 추가하시겠습니까?' 옆의 저장을 클릭하면 '배송지 추가가 완료되었습니다!' 알림이 표시되며 리스트에 추가가 완료된다.



[ 그림 3-28 지도에 리스트 항목 표시 ]

지도 탭에서는 각각 배송 목록에 해당하는 주소에 택배 코드로 마커가 표시된다. 상단의 현 위치를 클릭하면 두 번째 그림처럼 현 위치가 표시되며 검색창에서 택배코드를 입력하면 검색을 시도한 택배 코드를 중심으로 화면이 이동된다. 세 번째 그림 상단에서 '41281\_231'을 입력하고 검색을 눌렀을 때 화면에서 '41281\_231' 주소지를 중심으로 화면이 이동된 것을 확인할 수 있다.

### 3.4.8 택배기사 택배 배송 완료처리



[ 그림 3-29 택배 배송 완료처리 ]

배송 목록에서 각 리스트를 클릭하면 세부적인 내용이 표시된다. 표시된 내용을 우측으로 슬라이드 하면 두 번째 그림처럼 완료와 삭제 버튼이 표시되고 완료버튼을 클릭하면 해당 택배가 배송 완료 처리로 배송상태가 업데이트된다. 또한 기사정보, 운송장번호,

완료한 시간에 대한 서명값을 생성하여 저장한다. 삭제 버튼은 만약 기사가 QR코드를 잘못 스캔한 경우 리스트에서 삭제하기 위함이다. 삭제 버튼을 클릭하는 경우 배송상태도 택배 배정을 받기 이전 상태인 주문 완료로 상태 값이 업데이트된다. 기사의 배송목록에서 배송상태를 한눈에 파악하기 쉽도록 배송 시작의 경우에는 녹색, 배송이 완료된 상태는 청색으로 표시되며 만약 환불요청이 있는 경우 적색으로 표시된다.



[ 그림 3-30 배송완료 처리 ]

정상적으로 배송완료 처리가 되는 경우 [그림 3-30] 좌측 그림과 같이 '완료 처리했습니다' 알림이 발생하고, 이미 완료 처리가 되어있는 항목을 다시 완료처리를 하는 경우에는 '이미 완료 처리했습니다.' 알림이 발생한다. [그림 3-30] 우측 그림은 위에서 설명한 서명값이 complete-signature 필드안에 저장된 모습이다.

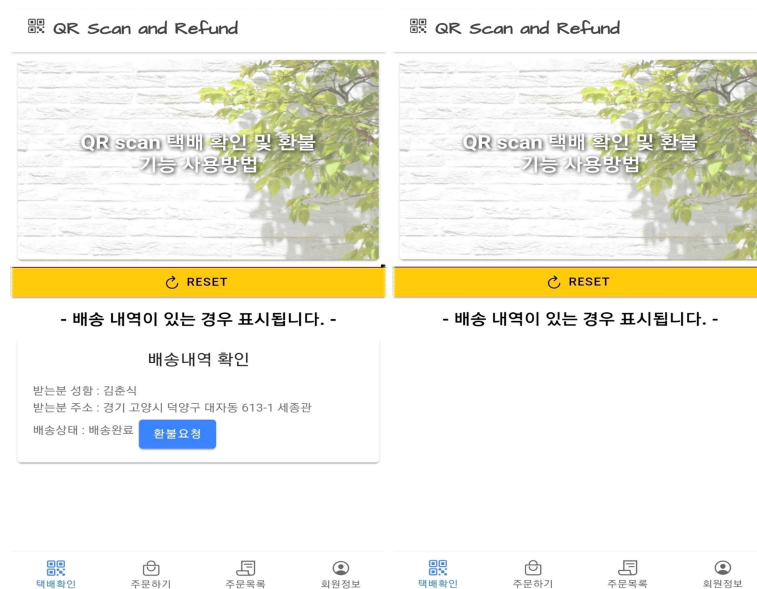
### 3.4.9 고객 택배 확인



[ 그림 3-31 고객 택배 확인 ]



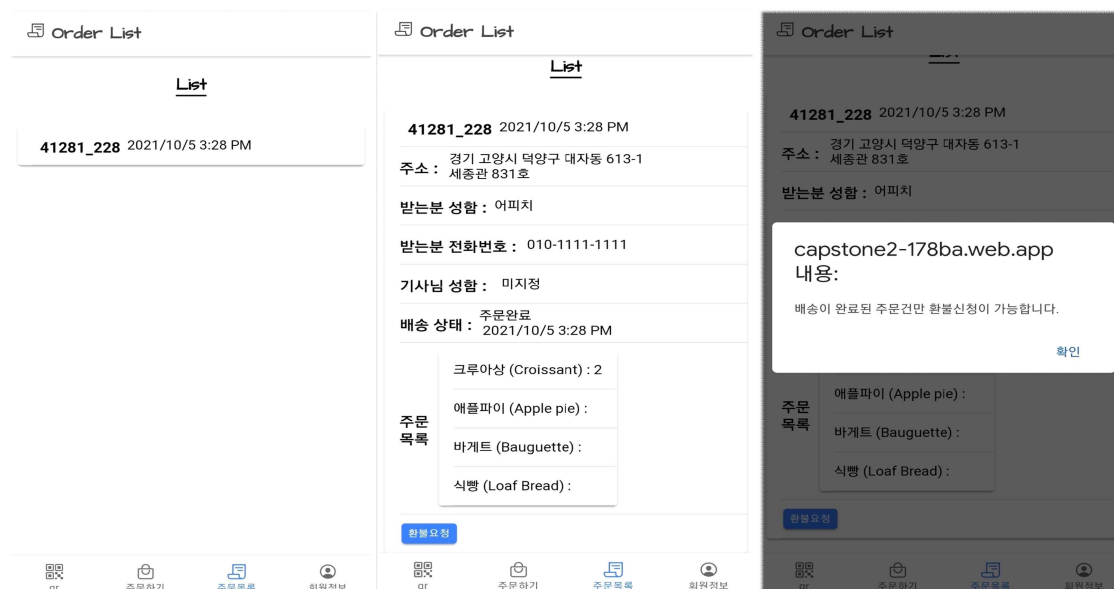
고객 전용 사이트에서 로그인을 진행하고 하단의 택배확인 탭을 클릭하면 [그림 3-31]의 좌측 그림처럼 페이지가 나타난다. 화면 중간 부분의 'Start scan' 버튼을 클릭하고 카메라 권한을 부여하면 우측 그림처럼 스캔 화면이 표시된다.



[ 그림 3-32 스캔 완료 화면 ]

로그인한 계정에 주문내역이 있는 경우 [그림 3-32]의 좌측 그림처럼 스캔한 QR코드에 해당하는 정보들이 화면에 표시되어 본인의 택배임을 알 수 있다. 그러나, 자신의 택배가 아닌 다른 고객이 주문한 택배의 QR코드를 스캔하는 경우 데이터를 불러올 수 없기 때문에 [그림 3-32]의 우측 화면처럼 표시된다.

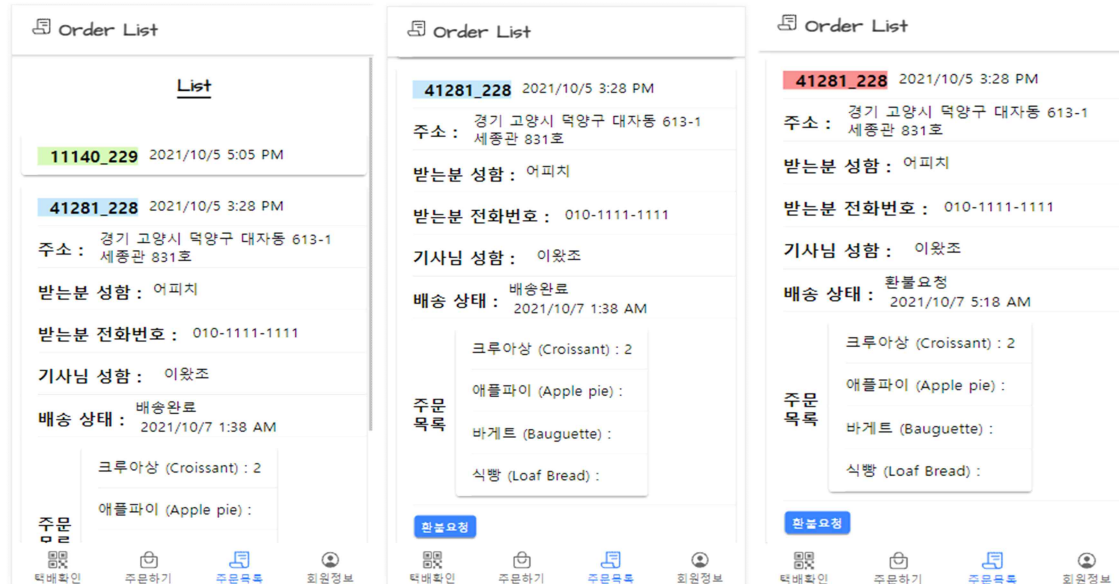
### 3.4.10 고객 주문 리스트 확인 및 환불처리



[ 그림 3-33 고객 주문 리스트 ]

로그인 되어있는 고객 페이지에서 하단의 주문목록으로 고객이 주문한 내역을 확인할

수 있다. 첫 번째 그림처럼 택배 코드와 주문시간이 표시되어있으며 클릭하게 되면 상세 정보들이 표시된다. 상세정보 하단의 환불요청 버튼은 택배의 배송상태가 '배송 완료'의 경우에만 환불 신청이 가능하다. [그림 3-33]에서 두 번째 그림을 확인하면 배송상태가 '배송 완료'가 아닌 '주문완료' 상태임을 알 수 있다. '주문완료' 혹은 '배송시작'에서 환불 요청 버튼을 클릭하게 되면 [그림 3-33]의 세 번째 그림처럼 '배송이 완료된 주문건만 환불신청이 가능합니다.' 알림이 발생하고 배송상태는 변화하지 않는다.



[ 그림 3-34 배송 상태별 고객 리스트 ]

고객 주문 리스트에서 '주문완료'의 경우에는 택배코드가 흰색, 배송 시작의 경우에는 녹색, 배송 완료의 경우에는 청색, 환불 요청의 경우에는 적색으로 표시된다.

## 4. 결론

### 4.1 결론 및 기대효과

언커넥티드 시대에 발맞춰 택배, 배달 서비스의 사용률이 크게 증가하고 있으며 그에 따라 개인정보 노출에 대한 범죄 또한 증가하는 추세이다.

본 프로젝트에서는 정보가 그대로 노출되어 있는 기존의 운송장이 아닌 QR코드를 사용하여 일차적으로 개인정보보호를 진행하며, 택배기사 혹은 택배 수령인 이외의 인물이 QR코드를 인식하여 개인정보를 수집할 수 있는 부분을 보완하여 로그인되어있는 사용자의 정보와 DB에 저장되어있는 수령자 정보가 일치하는 경우에만 개인정보를 확인할 수 있도록 하여 개인정보보호가 더욱 강화되었을 것으로 생각한다.

### 4.2 향후 과제

향후 과제로는 기사 계정 등록 후에 첫 로그인 시 기사 사용 기기의 UUID 혹은 IMEI 등의 기기의 고유 값을 이용하여 특정 기기에서만 사용이 가능하도록 인증서를 사용하여 구현할 계획이며, 배송지가 표시되는 지도 부분에서 인공지능을 활용하여 효율적인 배달

순서를 표시해주고 더 나아가 교통상황 등 실시간으로 효율적인 경로를 나타낼 계획이다. 또한, 고객과 기사가 소통할 수 있는 메신저와 같은 기능을 추가하여 배송 시작, 배송 완료 등 알림을 구현할 계획이다.

## 5. 별첨

### 5.1 참고자료

- [1] DENSO WAVE, the Inventor of QR Code, QR코드의 사양 및 특징,  
<https://www.qrcode.com/ko/>
- [2] 양형규, QR 코드의 보안 취약점과 대응 방안 연구, 한국인터넷방송통신학회 논문지, 제 12권 제 1호, 2012년 2월
- [3] 정소윤, 공개키 기반(PKI)의 전자서명 인증제도 활성화 방안 연구, 2001년 12월
- [4] 김무환, 택배서비스 개인정보보호를 위한 QR코드 운송장과 인식 프로토타입 구현, 2015년 12월

### 5.2 소스코드

전체 소스코드는 아래의 Github 주소에 업로드 완료하였습니다.

Github 주소 : [https://github.com/kwonhyeonju/2021\\_Capstone](https://github.com/kwonhyeonju/2021_Capstone)

```
shop - /register/register.page.ts
import { Component, OnInit } from '@angular/core';
import { AuthenticateService } from '../services/authentication.service';
import { FormGroup, FormBuilder, Validators, FormControl } from '@angular/forms';
import { Router } from '@angular/router';
import { AlertController } from '@ionic/angular';
import { LoadingController } from '@ionic/angular';
@Component({
  selector: 'app-register',
  templateUrl: './register.page.html',
  styleUrls: ['./register.page.scss'],
})
export class RegisterPage implements OnInit {

  validations_form: FormGroup;
  errorMessage: string = '';
  successMessage: string = '';
  ischecked:string='';
  constructor(
    public loadingController: LoadingController,
    private authService: AuthenticateService,
```

```

    private router: Router,
    private formBuilder: FormBuilder,
    private alertCtrl: AlertController
  ) { }

  async presentLoading() {
    const loading = await this.loadingController.create({
      cssClass: 'my-custom-class',
      message: 'Please wait...',
      duration: 1000
    });
    await loading.present();
  }

  ngOnInit() {
    this.validations_form = this.formBuilder.group({
      email: new FormControl('', Validators.compose([
        Validators.required,
        Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+$')
      ])),
      name: new FormControl('', Validators.compose([
        Validators.required,
      ])),
      password: new FormControl('', Validators.compose([
        Validators.minLength(5),
        Validators.required
      ])),
    });
  }

  registerUser(value) {
    this.presentLoading();
    try {
      this.authService.userRegistration(value).then(response => {
        console.log(response);
        if (response.user) {
          response.user.updateProfile({
            displayName: value.name,
            email: value.email
          });
          alert('회원가입이 완료되었습니다.')
          this.router.navigate(['login']);
        }
      }, error => {

```

```

        alert('중복된 계정입니다.\r\n유효한 이메일, 6자 이상의 비밀번호를 확인하여주십시오.')
    }
    )
    } catch (error) {
    }
    }
}
}

```

#### shop - /register/register.page.html

```

<ion-content>
  <form
    class="form"
    [formGroup]="validations_form"
    (ngSubmit)="registerUser(validations_form.value)">
    <h1>회원가입</h1>
    <br>
    <ion-label>이름</ion-label>
    <ion-input type="text" formControlName="name" >
    </ion-input>
    <ion-label>이메일</ion-label>
    <ion-input type="text" formControlName="email" >
    </ion-input>
    <ion-label>비밀번호</ion-label>
    <ion-input type="password" formControlName="password" >
    </ion-input>
    <button class="submit-btn" class="btn-primary" [disabled]="!validations_form.valid"
    type="submit">회원가입
    </button>
  </form>
</ion-content>

```

#### shop - /login/login.page.ts

```

import { Component, OnInit } from '@angular/core';
import { AngularFireStore } from '@angular/fire/firestore';
import { FormGroup, FormBuilder, Validators, FormControl } from '@angular/forms';
import { NavController } from '@ionic/angular';
import { AuthenticateService } from '../services/authentication.service';
import firebase from 'firebase/app';
import { LoadingController } from '@ionic/angular';

```

```

const INCREMENT = firebase.firestore.FieldValue.increment(1);
@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {
  validations_form: FormGroup;
  errorMessage: string = '';
  constructor(
    public loadingController: LoadingController,
    private navCtrl: NavController,
    private authService: AuthenticateService,
    private formBuilder: FormBuilder,
    private firestore: AngularFirestore
  ) { }
  async presentLoading() {
    const loading = await this.loadingController.create({
      cssClass: 'my-custom-class',
      message: 'Please wait...',
      duration: 1000
    });
    await loading.present();
  }
  ngOnInit() {

    this.validations_form = this.formBuilder.group({
      email: new FormControl('', Validators.compose([
        Validators.required,
        Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+$')
      ])),
      password: new FormControl('', Validators.compose([
        Validators.minLength(5),
        Validators.required
      ])),
    });
  }

  LoginUser(value) {
    this.presentLoading();
    try {
      this.authService.loginFireauth(value).then(resp => {

```

```

        console.log(resp);
        this.authService.setUser({
            username: resp.user.displayName,
            uid: resp.user.uid
        })
        if (resp.user) {
            const userProfile = this.firestore.collection('profile').doc(resp.user.uid);
            userProfile.get().subscribe(result => {
                this.navCtrl.navigateForward(['shop/home']);
                if (result.exists) {
                    this.navCtrl.navigateForward(['shop/home']);
                } else {
                    this.firestore.doc(`profile/${this.authService.getUserUid()}`).set({
                        name: resp.user.displayName,
                        email: resp.user.email })
                }
            })
        }
    },err=>{
        alert('로그인 정보가 일치하지 않습니다.')
    }
    )
}
catch(err){
    console.log(err);
}
}
goToRegisterPage() {
    this.navCtrl.navigateForward('/register');
}
}

```

#### shop - /login/login.page.html

```

<ion-content class="ion-padding">
    <form
        class="form"
        [formGroup]="validations_form"
        (ngSubmit)="LoginUser(validations_form.value)">
        <h1>Login</h1>
        <br>
        <ion-label position="floating">Email</ion-label>
    </form>
</ion-content>

```

```

    <ion-input type="text" formControlName="email"></ion-input>
    <ion-label position="floating" >Password</ion-label>
    <ion-input type="password" formControlName="password" class="form-controll"
required></ion-input>
    <ion-button class="submit-btn" type="submit" [disabled]="!validations_form.valid">로그
인</ion-button>
    <br>
</form>
<button (click)="goToRegisterPage()">회원가입</button>

</ion-content>

```

### shop - /login/login.page.ts

```

import { Component, OnInit } from '@angular/core';
import { AngularFireStore } from '@angular/fire/firestore';
import { FormGroup, FormBuilder, Validators, FormControl } from '@angular/forms';
import { NavController } from '@ionic/angular';
import { AuthenticateService } from '../services/authentication.service';
import firebase from 'firebase/app';
import { LoadingController } from '@ionic/angular';
const INCREMENT = firebase.firestore.FieldValue.increment(1);
@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {
  validations_form: FormGroup;
  errorMessage: string = '';
  constructor(
    public loadingController: LoadingController,
    private navCtrl: NavController,
    private authService: AuthenticateService,
    private formBuilder: FormBuilder,
    private firestore: AngularFireStore
  ) { }
  async presentLoading() {
    const loading = await this.loadingController.create({
      cssClass: 'my-custom-class',
      message: 'Please wait...',
      duration: 1000
    });
  }
}

```



```

    });
    await loading.present();
  }
  ngOnInit() {

    this.validations_form = this.formBuilder.group({
      email: new FormControl('', Validators.compose([
        Validators.required,
        Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+$')
      ])),
      password: new FormControl('', Validators.compose([
        Validators.minLength(5),
        Validators.required
      ])),
    });
  }

  LoginUser(value) {
    this.presentLoading();
    try {
      this.authService.loginFireauth(value).then(resp => {
        console.log(resp);
        this.authService.setUser({
          username: resp.user.displayName,
          uid: resp.user.uid
        })
        if (resp.user) {
          const userProfile = this.firestore.collection('profile').doc(resp.user.uid);
          userProfile.get().subscribe(result => {
            this.navCtrl.navigateForward(['shop/home']);
            if (result.exists) {
              this.navCtrl.navigateForward(['shop/home']);
            } else {
              this.firestore.doc(`profile/${this.authService.getUserUid()}`).set({
                name: resp.user.displayName,
                email: resp.user.email })
            }
          })
        }
      }, err=>{
        alert('로그인 정보가 일치하지 않습니다.')
      })
    }
  }

```

```

    )
  }
  catch(err){
    console.log(err);
  }
}
goToRegisterPage() {
  this.navCtrl.navigateForward('/register');
}
}

```

### shop - /login/login.page.html

```

<ion-content class="ion-padding">
  <form
    class="form"
    [formGroup]="validations_form"
    (ngSubmit)="LoginUser(validations_form.value)">
    <h1>Login</h1>
    <br>
    <ion-label position="floating">Email</ion-label>
    <ion-input type="text" formControlName="email"></ion-input>
    <ion-label position="floating" >Password</ion-label>
    <ion-input type="password" formControlName="password" class="form-controll"
    required></ion-input>
    <ion-button class="submit-btn" type="submit" [disabled]="!validations_form.valid">로그인</ion-button>
    <br>
  </form>
  <button (click)="goToRegisterPage()">회원가입</button>

</ion-content>

```

### shop - /orderlist/orderlist.page.html

```

<ion-header>
  <ion-toolbar>
    <ion-buttons slot="end">
      </ion-buttons>
    <ion-title class="titlemain"><ion-icon name="receipt-outline"></ion-icon>&nbsp;Order List</ion-title>
  </ion-toolbar>
</ion-header>

```

```
<ion-content>
<ion-list>
  <h4>List</h4><br>
  <ion-card *ngFor="let p of U_items |async" (click)="p.expanded=!p.expanded">
    <ion-item>
      <h5 class="{{p.dstate}}">&nbsp;&nbsp;&nbsp;{{p.d_number}}</h5>
&nbsp;&nbsp;&nbsp;{{p.ordercomplete.toDate() | date: 'y/M/d h:mm a'}}
    </ion-item>
    <div [hidden]="!p.expanded">
      <ion-item>
        <h5>주소 :</h5>&nbsp;&nbsp;&nbsp; {{p.address}} <br>&nbsp;&nbsp;&nbsp; {{p.detailaddress}}
      </ion-item>
      <ion-item>
        <h5>받는분 성함 :</h5> &nbsp;&nbsp;&nbsp;{{p.name}}
      </ion-item>
      <ion-item>
        <h5>받는분 전화번호 :</h5>&nbsp;&nbsp;&nbsp; {{p.phonenumber}}
      </ion-item>
      <ion-item>
        <h5>기사님 성함 :</h5>&nbsp;&nbsp;&nbsp; {{p.dname}}
      </ion-item>
      <ion-item>
        <h5>배송 상태 :</h5> &nbsp;&nbsp;&nbsp;{{p.dstate}} <br>&nbsp;&nbsp;&nbsp;{{p.lastUpdate.toDate() | date: 'y/M/d h:mm a'}}
      </ion-item>
      <ion-item>
        <h5>주문<br>목록</h5>
        <ion-card>
          <ion-item>크루아상 (Croissant) : {{p.Ko3E0BMPozfrsoxqAhMi}}</ion-item>
          <ion-item>애플파이 (Apple pie) : {{p.hHXB7BxHsSEMn6Qk0vjN}}</ion-item>
          <ion-item>바게트 (Bauguette) : {{p.qmWDJN5q0ziQbLqcJiTG}}</ion-item>
          <ion-item>식빵 (Loaf Bread) : {{p.qyqaF9QJvqyuesIs94Ge}}</ion-item>
        </ion-card>
      </ion-item>
      <ion-item>
        <ion-button (click)="ref(p)">환불요청</ion-button>
      </ion-item>
    </div>
  </ion-card>
</ion-list>
</ion-content>
```

**shop - /orderlist/orderlist.page.ts**

```
import { Component, OnInit } from '@angular/core';
import { NavController } from '@ionic/angular';
import { menuController } from '@ionic/core';
import firebase from 'firebase/app';
import { AngularFireStore } from '@angular/fire/firestore';
import { AuthenticateService } from '../services/authentication.service';
import { Observable } from 'rxjs';
import Timestamp = firebase.firestore.Timestamp;
@Component({
  selector: 'app-orderlist',
  templateUrl: './orderlist.page.html',
  styleUrls: ['./orderlist.page.scss'],
})
export class OrderlistPage implements OnInit {
  orders : Observable<any[]>;
  profile: any;
  profileName: any;
  profileEmail: any;
  oname:any;//주문자이름
  oaddress:any;//주문자주소
  odaddress:any;//주문자 상세주소
  dd:any;//배송상태
  a:any;//a상품 수량
  b:any;//b상품 수량
  c:any;//c상품 수량
  U_items : Observable<any[]>;
  U_orders : Observable<any[]>;

  constructor(private navCtrl: NavController, private database: AngularFireStore,
private authservice: AuthenticateService) { }

  ngOnInit() {
    firebase.auth().onAuthStateChanged(user => {
      console.log("AUTH_USER", user);
      if(user){
        const result =this.database.doc(`/profile/${this.authservice.getUserUid()}`);
        var userProfile=result.valueChanges();
      }
    });
  }
}
```

```

        userProfile.subscribe(profile=>{
            console.log("PROFILE::",profile);
            this.profileName=profile['name'];
            this.profileEmail=profile['email'];
        }
    )
}
}
)

this.U_items=this.authservice.getU_orders();
}
ref(p){
    if(p.dstate=="배송완료"){

this.database.doc(`profile/${this.authservice.getUserUid()}/cart/${p.Document_ID}`).update({
    dstate:"환불요청",lastUpdate:Timestamp.now()
})
    this.database.collection('orders').doc(p.Document_ID).update({dstate      :      "환불요청",
lastUpdate:Timestamp.now()});

    this.database.collectionGroup('D_orders', ref => ref.where('dstate', '==', "배송완료"
)).get().forEach(
        e => { e.docs.map(
            doc => { if (doc.id == p.Document_ID) {
                var path = doc.ref.path
                this.database.doc(path).update({          dstate          :          "환불요청",
lastUpdate:Timestamp.now() }) }
            })
        })
    }else{
        alert("배송이 완료된 주문건만 환불신청이 가능합니다.")
    }
}
}

```

#### shop – /services/authentication.service.ts

```

import { Injectable } from "@angular/core";
import { AngularFireAuth } from '@angular/fire/auth';
import { AngularFirestore, AngularFirestoreCollection, AngularFirestoreDocument }

```

```

from '@angular/fire/firestore';
import { BehaviorSubject } from 'rxjs';
import { Plugins } from '@capacitor/core';
const { Storage } = Plugins;
import firebase from 'firebase/app';
import { HomePage } from '../shop/home/home.page';
import { ThrowStmt } from "@angular/compiler";
import Timestamp = firebase.firestore.Timestamp
export interface User {
  username: string;
  uid: string;
}
const CART_STORAGE_KEY = 'MY_CART';
const INCREMENT = firebase.firestore.FieldValue.increment(1);
const DECREMENT = firebase.firestore.FieldValue.increment(-1);
@Injectable({
  providedIn: 'root'
})
export class AuthenticateService {
  private user: User;
  profileName:any;
  profileEmail:any;
  name: HomePage["name"];
  address: HomePage["address"];
  cart = new BehaviorSubject({});
  productsCollection: AngularFireStoreCollection;
  cartKey = null;
  orderKey = null;
  cartsCollection: AngularFireStoreCollection;
  orderCollection: AngularFireStoreCollection;
  cartsDocument:AngularFirestoreDocument;
  orderDocument:AngularFirestoreDocument;
  U_ordersCollection:AngularFirestoreCollection;
  authKey = null;
  constructor(
    private afAuth: AngularFireAuth,
    private afs: AngularFireStore,
    public fireservices: AngularFireStore,
  ) {
    this.productsCollection = this.afs.collection('products');
    this.cartsCollection = this.afs.collection('carts');
  }

```

```

    this.loadCart();
  }
  getOrders(){
    this.orderCollection= this.afs.collection(`profile/${this.getUserUid()}/cart`);
    return this.orderCollection.valueChanges({idField : 'id'});
  }
  getU_orders(){
    r           e           t           u           r           n
    this.afs.collection(`profile/${this.getUserUid()}/cart`).valueChanges({idField: 'Document_ID'
  })
  }
  getOrders2(){
    this.orderCollection=this.afs.collection('orders')
    return this.orderCollection.valueChanges({idField : 'id'});
  }

  getNum(){
    this.cartsDocument=this.afs.collection('carts').doc('number');
    return this.cartsDocument.valueChanges({idField: 'id'});
  }
  setUser(user: User) {
    return this.user = user;
  }

  getUserUid(): string {
    return this.user.uid
  }
  loginFireauth(value) {
    return new Promise<any>((resolve, reject) => {
      this.afAuth.signInWithEmailAndPassword(value.email, value.password).then(
        res => resolve(res),
        error => reject(error)
      )
    })
  }

  userRegistration(value) {
    return new Promise<any>((resolve, reject) => {
      firebase.auth().createUserWithEmailAndPassword(value.email,
value.password).then(
        res => resolve(res),
        error => reject(error)
      )
    })
  }

```

```

    )
  })
}
registerUser(value) {
  return new Promise<any>((resolve, reject) => {

    this.afAuth.createUserWithEmailAndPassword(value.email, value.password)
      .then(
        res => resolve(res),
        err => reject(err))
  })

}
userDetails() {
  return this.afAuth.user
}
create_user(Record) {
  return this.afs.doc(`profile/${this.getUserUid()}`).update(Record);
}
create_Newuser(Record) {
  this.afs.collection('carts').doc('number').update({
    number: INCREMENT
  });
  return this.afs.collection('carts').doc(this.cartKey).update(Record);
}
getProducts() {
  return this.productsCollection.valueChanges({ idField: 'id' });
}

async loadCart() {
  const result = await Storage.get({ key: CART_STORAGE_KEY });
  console.log('Cart from stoarge: ', result);

  if (result.value) {
    this.cartKey = result.value

    this.afs.collection('carts').doc(this.cartKey).valueChanges().subscribe((result: any)
=> {
      console.log('cart changed :', result);
      this.cart.next(result || {});
    })
  }
}

```



```

    } else {

        const fbDocument = await this.afs.collection('carts').add({
            lastUpdate: firebase.firestore.FieldValue.serverTimestamp(),

        });
        console.log('new cart:', fbDocument);
        this.cartKey = fbDocument.id;
        await Storage.set({ key: CART_STORAGE_KEY, value: this.cartKey });
    }
}

addToCart(id) {
    this.afs.collection('carts').doc(this.cartKey).update({
        [id]: INCREMENT,
        ordercomplete: Timestamp.now(),
        lastUpdate: Timestamp.now(),
        deliverystart: Timestamp.now(),
        deliveriesuccess: Timestamp.now(),
        사업자번호: "AA",
        사업자: "택배왔조",
        dname: "미지정",
        dstate: "주문완료",
    });
}

addUser() {
    this.afs.collection('carts').doc(this.cartKey).update({
        이름: this.name,
        주소: this.address
    });
}

removeFromCart(id) {
    this.afs.collection('carts').doc(this.cartKey).update({
        [id]: DECREMENT,
        lastUpdate: firebase.firestore.FieldValue.serverTimestamp()
    });
}

checkoutCart() {
    firebase.auth().onAuthStateChanged(user => {
        console.log('AUTH_USER', user);
        if (user) {
            const result = this.afs.doc(`/profile/${this.getUserUid()}`);

```

```

        var userProfile = result.valueChanges();
        userProfile.subscribe(profile => {
            console.log('PROFILE::', profile);
            console.log(Object.keys(profile));
            this.profileName = profile['name'];
            this.profileEmail = profile['email'];
        });
        const docRef = this.afs
            .collection(`profile/${this.getUserUid()}/cart`)
            .doc();
        docRef.set(this.cart.value);
        const orderRef = this.afs.collection('orders').doc(docRef.ref.id);
        orderRef.set(this.cart.value);
    }
});

this.afs
    .collection('carts')
    .doc(this.cartKey)
    .set({
        lastUpdate: firebase.firestore.FieldValue.serverTimestamp()
    });
}

async checkoutCart2() {
    await this.afs.collection('orders').add(this.cart.value);

    firebase.auth().onAuthStateChanged(user => {
        console.log("AUTH_USER", user);

        if(user){
            const result =this.afs.doc(`/profile/${this.getUserUid()}`);
            var userProfile=result.valueChanges();
            userProfile.subscribe(profile=>{
                console.log("PROFILE::",profile);
                this.profileName=profile['name'];
                this.profileEmail=profile['email'];
            })
            this.afs.collection(`profile/${this.getUserUid()}/cart`).add(this.cart.value);
        }
    })
}

```

```

        this.afs.collection('carts').doc(this.cartKey).set({
            lastUpdate: firebase.firestore.FieldValue.serverTimestamp()
        })
    }
}

```

**admin-/list/list.component.html**

[illegible]

```

                <a      class="nav-link"      aria-current="page"      href="main"><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-house-fill" viewBox="0 0 16 16">
                <path fill-rule="evenodd" d="m8 3.293 6 6V13.5a1.5 1.5 0 0 1-1.5
1.5h-9A1.5 1.5 0 0 1 2 13.5V9.293l6-6zm5-.793V6l-2-2V2.5a.5.5 0 0 1 .5-.5h1a.5.5 0 0
1 .5.5z"/>
                <path fill-rule="evenodd" d="M7.293 1.5a1 1 0 0 1 1.414 0l6.647
6.646a.5.5 0 0 1-.708.708L8 2.207 1.354 8.854a.5.5 0 1 1-.708-.708L7.293 1.5z"/>
                </svg>&nbsp;Home</a>
        </li>
        <li class="nav-item">
                <a      class="nav-link      active"      href="list"><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-cart-check-fill" viewBox="0 0 16 16">
                <path d="M.5 1a.5.5 0 0 0 0 1h1.111.401 1.607 1.498 7.985A.5.5 0 0
0 4 12h1a2 2 0 1 0 0 4 2 2 0 0 0 0-4h7a2 2 0 1 0 0 4 2 2 0 0 0 0-4h1a.5.5 0 0 0
.491-.408l1.5-8A.5.5 0 0 0 14.5 3H2.89l-.405-1.621A.5.5 0 0 0 2 1H.5zM6 14a1 1 0 1
1-2 0 1 1 0 0 1 2 0zm7 0a1 1 0 1 1-2 0 1 1 0 0 1 2 0zm-1.646-7.646-3 3a.5.5 0 0
1-.708 0l-1.5-1.5a.5.5 0 1 1 .708-.708L8 8.293l2.646-2.647a.5.5 0 0 1 .708.708z"/>
                </svg>&nbsp;OrderList</a>
        </li>
        <li class="nav-item">
                <a      class="nav-link"      href="delivery"><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-truck" viewBox="0 0 16 16">
                <path d="M0 3.5A1.5 1.5 0 0 1 1.5 2h9A1.5 1.5 0 0 1 12
3.5V5h1.02a1.5 1.5 0 0 1 1.17.563l1.481 1.85a1.5 1.5 0 0 1 .329.938V10.5a1.5 1.5 0 0
1-1.5 1.5H14a2 2 0 1 1-4 0H5a2 2 0 1 1-3.998-.085A1.5 1.5 0 0 1 0 10.5v-7zm1.294
7.456A1.999 1.999 0 0 1 4.732 11h5.536a2.01 2.01 0 0 1 .732-.732V3.5a.5.5 0 0
0-.5-.5h-9a.5.5 0 0 0-.5.5v7a.5.5 0 0 0 .294.456zM12 10a2 2 0 0 1 1.732 1h.768a.5.5 0
0 0 .5-.5V8.35a.5.5 0 0 0-.11-.312l-1.48-1.85A.5.5 0 0 0 13.02 6H12v4zm-9 1a1 1 0 1
0 0 2 1 1 0 0 0 0-2zm9 0a1 1 0 1 0 0 2 1 1 0 0 0 0-2z"/>
                </svg>&nbsp;Delivery</a>
        </li>
        <li class="nav-item">
                <a      class="nav-link"      href="link"><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-link-45deg" viewBox="0 0 16 16">
                <path d="M4.715 6.542 3.343 7.914a3 3 0 1 0 4.243
4.243l1.828-1.829A3 3 0 0 0 8.586 5.5L8 6.086a1.002 1.002 0 0 0-.154.199 2 2 0 0 1
.861 3.337L6.88 11.45a2 2 0 1 1-2.83-2.83l.793-.792a4.018 4.018 0 0 1-.128-1.287z"/>

```

```

        <path d="M6.586 4.672A3 3 0 0 0 7.414 9.51.775-.776a2 2 0 0
1-.896-3.346L9.12 3.55a2 2 0 1 1 2.83 2.83l-.793.792c.112.42.155.855.128
1.287l1.372-1.372a3 3 0 1 0-4.243-4.243L6.586 4.672z"/>
        </svg>&nbsp;Link</a>
    </li>
</ul>
</div>
</div>
</nav>
<button style="float:right" onclick="window.open('qr','window','location=no,
directories=no,resizable=no,status=no,toolbar=no,menubar=no,
width=500,height=400,left=0, top=0, scrollbars=yes');return false"> 운송장인쇄 </button>
<table class="table">
<thead>
<tr>
<th>이름</th>
<th>전화번호</th>
<th>주소</th>
<th>주문목록</th>
<th>운송장</th>
<th>배정 기사</th>
<th>배송 상태</th>
<th>배송 현황</th>
</tr>
</thead>
<tbody>
<tr *ngFor="let p of orders | async" class="{{p.dstate}}">

<td>{{p.name}}</td>
<td>{{p.phonenumber}}</td>
<td>{{p.address}} {{p.detailaddress}}</td>
<td>
크루아상 (Croissant) : {{p.Ko3E0BMPozfrsoxqAhMi}}<br>
애플파이 (Apple pie) : {{p.hHXB7BxHsSEMn6Qk0vjN}}<br>
바게트 (Baguette) : {{p.qmWDJN5q0ziQbLqcJiTG}}<br>
식빵 (Loaf Bread) : {{p.qyqaF9QJvqyuesIs94Ge}}<br>
</td>
<td>
<button data-toggle="collapse" [attr.data-target]="'#demo'+p.id">QR운송장
</button>

<div id="demo{{ p.id }}" class="collapse">

```

```

        <div id="selectArea">
            <qrcode [qrdata]="p.id" [width]="256"
[errorCorrectionLevel]="M"></qrcode>
        </div>
    </div>
</td>
<td>{{p.dname}}</td>
<td class="state">{{p.dstate}}</td>
<td *ngIf="p.ordercomplete"> 주문완료 (배송준비) : <br>
{{p.ordercomplete.toDate() | date: 'y/M/d h:mm a'}}
<br>
배송시작 : <br> {{p.deliverystart.toDate() | date: 'y/M/d h:mm a'}}
<br>
배송완료 :<br> {{p.deliverysuccess.toDate() | date: 'y/M/d h:mm a'}}
</td>
</tr>
</tbody>
</table>
</div>
</div>
</main>
<footer>
    Watzzo | Copyright &copy; 2021 캡스톤디자인 택배왔조
</footer>
</body>
</html>

```

#### admin-/list/list.component.ts

```

import { Inject, LOCALE_ID, Pipe, PipeTransform, Component, OnInit } from
'@angular/core';
import { Router } from "@angular/router";
import { AngularFireStore, AngularFireStoreCollection, AngularFireStoreDocument}
from '@angular/fire/firestore';
import firebase from 'firebase/app';
import { Observable } from 'rxjs';
import { OrderService } from '../service/order.service';
import firestore from 'firebase';
import Timestamp=firebase.firestore.Timestamp;
import { formatDate } from '@angular/common';
@Component({

```

```

    selector: 'app-list',
    templateUrl: './list.component.html',
    styleUrls: ['./list.component.scss']
  })
  export class ListComponent implements OnInit {
    dateadded:any;
    boxes:any;
    box:any;
    selectArea:any;
    orders : Observable<any[]>;
    public myAngularxQrCode: string = null;
    public modal:boolean=false;
    constructor(private OrderService :OrderService) {

    }
    ngOnInit(){
      this.orders=this.OrderService.getOrders();
    }
    printArea() {
      window.print();
    }
    fnPrint(){
      document.body.innerHTML = this.selectArea.innerHTML;
      window.print();
    }
  }
}

```

### **shop-/qrscan/qrscan.page.html**

```

<ion-header [translucent]="true">
  <ion-toolbar>

  <ion-title>    <ion-icon name="qr-code-outline">&nbsp;</ion-icon>&nbsp;QR Scan and
  Refund</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="card-background-page">
  <ion-card id="mcard">
    <ion-slides pager="true" [options]="slideOpts">

```

```

<ion-slide>
  
  <div class="card-title1">QR scan 택배 확인 및 환불 <br>기능 사용방법</div>
<!-- <div class="card-subtitle">주문하실 상품을 선택해주세요</div> -->
</ion-slide>
<ion-slide>
  
  <!-- <div class="card-title">Shop</div> -->

  <div class="card-title">1. 배송받으신 박스 위에 부착되어있는 <br>QR 코드를 'start scan'
  버튼을 <br>클릭하여 스캔합니다.</div>
</ion-slide>
<ion-slide>
  
  <!-- <div class="card-title">Shop</div> -->

  <div class="card-title">2. 자신의 상품이 맞는 경우 <br>상품 정보와 환불하기 버튼이 표시
  됩니다.</div>
</ion-slide>
<ion-slide>
  
  <!-- <div class="card-title">Shop</div> -->

  <div class="card-title">3. 환불하기 버튼을 클릭하여 <br>환불을 완료합니다.</div>
</ion-slide>
<ion-slide>
  
  <!-- <div class="card-title">Shop</div> -->

  <div class="card-title">4. 주문목록에서 배송상태의 <br>환불요청 표시를 확인합니다.</div>
</ion-slide>
</ion-slides>
</ion-card>

<button class="btn"(click)="startScan()">
  Start scan
</button>

<ion-button
  expand="full"
  (click)="reset()"
  color="warning"

```



```

      *ngIf="scanResult"
    >
      <ion-icon slot="start" name="refresh"></ion-icon>
      Reset
    </ion-button>

    <!-- Shows our camera stream -->
    <video #video [hidden]="!scanActive" width="100%"></video>

    <!-- Used to render the camera stream images -->
    <canvas #canvas hidden></canvas>

    <!-- Stop our scanner preview if active -->
    <ion-button
      expand="full"
      (click)="stopScan()"
      color="danger"
      *ngIf="scanActive"
    >
      <ion-icon slot="start" name="close"></ion-icon>
      Stop scan
    </ion-button>
    <h4>- 배송 내역이 있는 경우 표시됩니다. -</h4>
    <div *ngFor="let p of u_orders | async">
      <ion-card *ngIf="p.id == scanResult">
        <ion-card-header>
          <ion-card-title>배송내역 확인</ion-card-title>
        </ion-card-header>
        <ion-card-content>
          받는분 성함 : {{p.name}} <br>
          받는분 주소 : {{p.address}} {{p.detailaddress}} <br>
          배송상태 : {{p.dstate}}
          <ion-button (click)="showQrToast()">환불요청</ion-button>
        </ion-card-content>
      </ion-card>
    </div>
  </ion-content>

```

**shop-/qrscan/qrscan.page.ts**

```

import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { ToastController, LoadingController, Platform, AlertController } from '@ionic/angular';
import { ProductService } from '../services/product.service';
import { AuthenticateService } from '../services/authentication.service';
import { Observable } from 'rxjs';
import { NavController } from '@ionic/angular';
import firebase from 'firebase/app';
import { AngularFirestore } from '@angular/fire/firestore';
import jsQR from 'jsqr';
// import { qr } from '../models/qr';

@Component({
  selector: 'app-scanner',
  templateUrl: './qrscan.page.html',
  styleUrls: ['./qrscan.page.scss']
})
export class QrscanPage implements OnInit {
  orders : Observable<any[]>;
  profile: any;
  profileName: any;
  profileEmail: any;
  profilephone: any;
  profileaddress: any;
  profiledaddress: any;

  u_orders: Observable<any[]>;
  @ViewChild('video', { static: false }) video: ElementRef;
  @ViewChild('canvas', { static: false }) canvas: ElementRef;

  canvasElement: any;
  videoElement: any;
  canvasContext: any;
  scanActive = false;
  scanResult = null;
  loading: HTMLIonLoadingElement = null;

  constructor(
    private database: AngularFirestore,
    private navCtrl: NavController,
    private toastCtrl: ToastController,
    private loadingCtrl: LoadingController,

```

```

private plt: Platform,
public ProductService: ProductService,
private authService: AuthenticateService,
private alertCtrl: AlertController
) {
  const isStandaloneMode = () =>
    'standalone' in window.navigator && window.navigator['standalone'];
  if (this.plt.is('ios') && isStandaloneMode()) {
    console.log('I am a an iOS PWA!');
    // E.g. hide the scan functionality!
  }
}
ngAfterViewInit() {
  this.canvasElement = this.canvas.nativeElement;
  this.canvasContext = this.canvasElement.getContext('2d');
  this.videoElement = this.video.nativeElement;
}
ngOnInit() {
  firebase.auth().onAuthStateChanged(user => {
    console.log("AUTH_USER", user);

    if(user){
      const result =this.database.doc(`/profile/${this.authService.getUserUid()}`);

      var userProfile=result.valueChanges();
      userProfile.subscribe(profile=>{
        console.log("PROFILE::",profile);
        this.profileName=profile['name'];
        this.profileEmail=profile['email'];
        this.profilephone=profile['phonenummer'];
        this.profileaddress=profile['address'];
        this.profiledaddress=profile['detailaddress']
      })
    }

  })

  this.orders=this.authService.getOrders();
  this.u_orders = this.authService.getOrders();
}

```

```

async startScan() {
  // Not working on iOS standalone mode!
  const stream = await navigator.mediaDevices.getUserMedia({
    video: { facingMode: 'environment' }
  });
  this.videoElement.srcObject = stream;
  // Required for Safari
  this.videoElement.setAttribute('playsinline', true);
  this.loading = await this.loadingCtrl.create({});
  await this.loading.present();
  this.videoElement.play();
  requestAnimationFrame(this.scan.bind(this));
}
async scan() {
  if (this.videoElement.readyState === this.videoElement.HAVE_ENOUGH_DATA) {
    if (this.loading) {
      await this.loading.dismiss();
      this.loading = null;
      this.scanActive = true;
    }
    this.canvasElement.height = this.videoElement.videoHeight;
    this.canvasElement.width = this.videoElement.videoWidth;
    this.canvasContext.drawImage(
      this.videoElement,
      0,
      0,
      this.canvasElement.width,
      this.canvasElement.height
    );
    const imageData = this.canvasContext.getImageData(
      0,
      0,
      this.canvasElement.width,
      this.canvasElement.height
    );
    const code = jsQR(imageData.data, imageData.width, imageData.height, {
      inversionAttempts: 'dontInvert'
    });
    if (code) {
      this.scanActive = false;
      this.scanResult = code.data;
    } else {

```

```

        if (this.scanActive) {
            requestAnimationFrame(this.scan.bind(this));
        }
    } else {
        requestAnimationFrame(this.scan.bind(this));
    }
}

re(){
    this.database.doc(`orders/${this.scanResult}`).update({
        dstate:"환불요청"
    })
    this.navCtrl.navigateForward('/orderlist');
}

stopScan() {
    this.scanActive = false;
}

reset() {
    this.scanResult = null;
}

async showQrToast() {
    const toast = await this.toastCtrl.create({
        message: `환불요청 하시겠습니까?`,
        position: 'top',
        buttons: [
            {
                text: '환불요청 하기',
                handler: () => {
                    this.database.doc(`orders/${this.scanResult}`).update({
                        dstate:"환불요청"
                    })
                }
            }
        ]
    })
    toast.present()

    this.database.doc(`profile/${this.authService.getUserUid()}/cart/${this.scanResult}`).update({
        dstate:"환불요청"
    })
    alert("환불 요청이 완료되었습니다.")
    this.navCtrl.navigateForward('shop/orderlist');
}

}

```

```

    ]
  });
  toast.present();
}
}

```

### delivery-/scanner/scanner.page.html

```

<ion-content class="main">

  <ion-button block class="start-scan" expand="block" (click)="startScan()">
    <ion-icon slot="start" name="scan-outline"></ion-icon>
    Start scan
  </ion-button>

  <ion-fab horizontal="end" vertical="bottom" slot="fixed" (click)="reset("
*ngIf="scanResult">
    <ion-fab-button color="light">
      <ion-icon name="refresh"></ion-icon>
    </ion-fab-button>
  </ion-fab>

  <!-- Shows our camera stream -->
  <video #video [hidden]="!scanActive" width="100%"></video>

  <!-- Used to render the camera stream images -->
  <canvas #canvas hidden></canvas>

  <!-- Stop our scanner preview if active -->
  <ion-button class="stop-scan" expand="block" (click)="stopScan()" color="danger"
*ngIf="scanActive">
    <ion-icon slot="start" name="close"></ion-icon>
    Stop scan
  </ion-button>
  <div *ngFor="let p of u_orders | async">
    <ion-card *ngIf="p.id == scanResult">
      <h4>택배 코드 : {{p.d_number}} <ion-button class="add" (click)="showQrToast()">
추가</ion-button></h4>
    </ion-card>
  </div>
</ion-content>

```

### delivery – /maps/maps.page.ts

```
import { Component, OnInit } from '@angular/core';
import { ItemService } from '../services/item.service';

declare var kakao;

@Component({
  selector: 'app-maps',
  templateUrl: './maps.page.html',
  styleUrls: ['./maps.page.scss']
})
export class MapsPage implements OnInit {
  map: any;
  marker: any;
  coords: any;
  items?: any[];
  geocoder: any;
  findID: any;
  arr1?: any = [];
  myoverlay: any;
  closeButton = false;
  constructor(public itemService: ItemService) {}
  ngOnInit() {
    var mapContainer = document.getElementById('map'), // 지도를 표시할 div
        mapOption = {
          center: new kakao.maps.LatLng(37.5421183825596, 126.948522081226), // 지도의
중심좌표
          level: 3 // 지도의 확대 레벨
        };
    //추가
    this.map = new kakao.maps.Map(mapContainer, mapOption);
  }
  async ionViewWillEnter() {
    console.log('ion');
    var geocoder = new kakao.maps.services.Geocoder();
    this.findID = null;
    this.arr1 = await this.itemService.mapFirstItems().then(newData => {
      console.log(newData);
      this.renderMap(newData, this.map, geocoder);
      this.move(newData, this.map, geocoder);
      this.arr1 = newData;
    });
  }
}
```

```

}

renderMap(arr1, map, geocoder) {
  setTimeout(() => {
    // 살짝 딜레이 줘야 화면에 맵에 쪽바로 그려진다.
    var center = map.getCenter();
    console.log(center.getLat()); // 37.542118382559
    console.log(center.getLng()); // 126.57066121198349
    // 주소-좌표 변환 객체를 생성합니다
    var imageSrc =
      'https://t1.daumcdn.net/localimg/localimages/07/mapapidoc/marker_red.png';
    // 마커 이미지의 이미지 크기 입니다
    var imageSize = new kakao.maps.Size(24, 35);
    // 마커 이미지를 생성합니다
    var markerImage = new kakao.maps.MarkerImage(imageSrc, imageSize, {
      offset: new kakao.maps.Point(15, 0)
    });
    var length = arr1.length;
    console.log(length);
    map.relayout();
    for (let i = 0; i < length; i++) {
      setTimeout(() => {
        console.log(i);
        // 주소로 좌표를 검색합니다
        console.log();
        var name = arr1[i].name;
        var phone = arr1[i].phonenumber;
        var address = arr1[i].address;
        var d_number= arr1[i].d_number;
        console.log(name);
        geocoder.addressSearch(address, function(result, status) {
          // 정상적으로 검색이 완료됐으면
          if (status === kakao.maps.services.Status.OK) {
            var addr = result[0].road_address.address_name;
            var coords = new kakao.maps.LatLng(result[0].y, result[0].x);
            var contents =
              '<div class="customoverlay" style=" position: relative;width: auto;height: auto;background: #808289;padding: 15px 10px;">' +
              '<div class="title" style=" overflow: hidden;color: #fff;font-size: 16px;font-weight: bold;background: url("../../assets/icon/png") no-repeat right 20px center;margin-bottom: 8px;">' +
              d_number +

```



```

        '</div>' +
        '</div>';
    var l_box = document.createElement('div');
    l_box.style.cssText =
        'width:                auto;height:auto;overflow:hidden;border-radius:
5px;border-bottom: 2px solid #fff;background: #fff;';
    var s_box = document.createElement('div');
    s_box.style.cssText =
        'position: relative;margin: 3px 1px 0 3px;background:#fff;';
    var closeBtn = document.createElement('button');

    closeBtn.style.cssText =
        'display:block; margin-left:auto;margin-right: 2px;';
    var closeImg = document.createElement('img');
    closeImg.src = '../../assets/icon/overlay_close.png';
    var content_ul = document.createElement('ul');
    content_ul.style.cssText =
        'width:auto; height:auto;overflow:hidden;margin:0;padding:0;';
    var li_add = document.createElement('li');
    var li_name = document.createElement('li');
    var li_phone = document.createElement('li');
    li_add.style.cssText =

'position:relative;background:#ccc;margin-bottom:2px;margin-right:3px;line-height:
1;padding:5px 10px;';
    li_name.style.cssText =

'position:relative;background:#ccc;margin-bottom:2px;margin-right:3px;line-height:
1;padding:5px 10px;';
    li_phone.style.cssText =

'position:relative;background:#ccc;margin-bottom:2px;margin-right:3px;line-height:
1;padding:5px 10px;';

    var span1 = document.createElement('span');
    var span2 = document.createElement('span');
    var span3 = document.createElement('span');
    span1.textContent = '주소:' + addr;
    span2.textContent = '이름:' + name;
    span3.textContent = '번호:' + phone;
    span1.style.cssText = 'display:inline-block;';
    span2.style.cssText = 'display:inline-block;';
    span3.style.cssText = 'display:inline-block;';

```

```

li_add.append(span1);
li_name.append(span2);
li_phone.append(span3);
closeBtn.append(closeImg);
l_box.appendChild(s_box).appendChild(closeBtn);
s_box.appendChild(content_ul);
content_ul.appendChild(li_add);
li_add.after(li_name);
li_name.after(li_phone);
var mcontents = l_box;

var marker = new kakao.maps.Marker({
    map: map, // 마커를 표시할 지도
    position: coords // 마커를 표시할 위치,
});
marker.setMap(map);
var customOverlay = new kakao.maps.CustomOverlay({
    map: map,
    zIndex: 4,
    position: coords,
    content: contents,
    removable: false,
    xAnchor: 0.5,
    yAnchor: 1.7
});
customOverlay.setMap(map);
}
kakao.maps.event.addListener(marker, 'click', function() {
    // 마커 위에 인포윈도우를 표시합니다
    var overlay = new kakao.maps.CustomOverlay({
        map: map,
        position: marker.getPosition(),
        xAnchor: 0.5,
        yAnchor: 1.81,
        removable: true
    });
    //0.4-0.5사이 조절
    overlay.setContent(mcontents);
    overlay.setMap(map);
    //console.log('ok');

    closeBtn.onclick = function() {

```

```

        overlay.setMap(null);
    };
    }); //click event (more info)
    }); //addressSearch
    }, 1000 * i); //for문 set time
} //for문
map.relayout();
}, 300); //set time
} //renderMap
move(arr1, map, geocoder) {
    var search_id = document.getElementById('search_id');
    var myplace_id = document.getElementById('myplace_id');
    search_id.onclick = () => {
        var findID = this.findID;
        var search_result = arr1.find(value => value.d_number === findID);
        geocoder.addressSearch(search_result.address, function(result, status) {
            if (status === kakao.maps.services.Status.OK) {
                var moveLatLon = new kakao.maps.LatLng(result[0].y, result[0].x);
                map.setCenter(moveLatLon);
            }
        });
        this.renderMap(arr1, map, geocoder);
    };
    myplace_id.onclick = () => {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(function(position) {
                console.log(position);
                var lat = position.coords.latitude, // 위도
                    lon = position.coords.longitude; // 경도
                var locPosition = new kakao.maps.LatLng(lat, lon); // 마커가 표시될 위치를
geolocation으로 얻어온 좌표로 생성합니다
                // 마커와 인포윈도우를 표시합니다
                map.setCenter(locPosition);
                myplace(locPosition, function(result, status) {
                    if (status === kakao.maps.services.Status.OK) {
                        console.log(result);
                        var detailAddr;
                        if (result[0].road_address == null) {
                            detailAddr = result[0].address.address_name;
                        } else {
                            detailAddr = result[0].road_address.address_name;
                        }
                    }
                });
            });
        }
    };
}

```

```

        console.log(detailAddr);
        var contents =
            '<div class="customoverlay" style=" position: relative;width:
auto;background: blue;padding: 12px 10px;height:40px ;border-radius:5px">' +
            '<div class="title" style=" overflow: hidden;color: #fff;font-size:
16px;font-weight: bold;margin-bottom: 8px;">' +
            "현위치" +
            '</div>' +
            '</div>';
        console.log(contents);
        // 마커를 클릭한 위치에 표시합니다
        var marker = new kakao.maps.Marker();
        marker.setPosition(locPosition);
        marker.setMap(map);
        var customOverlay = new kakao.maps.CustomOverlay({
            map: map,
            zIndex: 4,
            position: locPosition,
            content: contents,
            removable: false,
            xAnchor: 0.5,
            yAnchor: 1.7
        });
        customOverlay.setMap(map);
    }
    });
}
};

function myplace(locPosition, callback) {
    geocoder.coord2Address(
        locPosition.getLng(),
        locPosition.getLat(),
        callback
    );
}
}
}

```

### **delivery-/services/authentication.service.ts**

```
import { Injectable } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { AngularFireStore,AngularFirestoreCollection, } from '@angular/fire/firestore';
import firebase from 'firebase';
import { Observable } from 'rxjs';
export interface User{
  username : string;
  uid : string;
}
@Injectable({
  providedIn: 'root'
})
export class AuthenticationService {
  profileName: any;
  orderCollection: AngularFirestoreCollection;
  constructor(private afAuth: AngularFireAuth, private afs: AngularFirestore) { }
  private user : User;
  loginUser(value) {
    return new Promise<any>((resolve, reject) => {
      this.afAuth.signInWithEmailAndPassword(value.email, value.password)
        .then(
          res => resolve(res),
          err => reject(err))
    })
  }

  setUser(user: User){
    return this.user = user;
  }
  getUserUid():string{
    return this.user.uid;
  }

  getOrders2(){
    this.orderCollection=this.afs.collection('orders')
    return this.orderCollection.valueChanges({idField : 'id'});
  }
}
```

### **delivery-/services/item.service.ts**

```
import { Injectable } from '@angular/core';
import { AngularFire, AngularFireCollection, AngularFireDocument } from '@angular/fire/firestore';
import firebase from 'firebase';
import { AuthenticationService } from '../authentication.service';
import Timestamp = firebase.firestore.Timestamp;

@Injectable({
  providedIn: 'root'
})
export class ItemService {
  ordersCollection: AngularFireCollection;
  D_ordersCollection: AngularFireCollection;
  //
  ordersNotAssign: AngularFireCollection<any[]>; //ordersNotAssign :  dname ==
"미지정" => D_orders에 존재 안함.
  //
  MapitemsCollection: AngularFireCollection<any[]>;
  arr2?: any = [];

  constructor(public afs: AngularFire, private authservice : AuthenticationService)
  {

    const uid = this.authservice.getUserUid()
    this.ordersCollection = this.afs.collection('orders');
    this.D_ordersCollection = this.afs.collection('Delivery').doc(uid).collection('D_orders')

    this.ordersNotAssign = this.afs
      .collection('orders', ref => ref.where('dname', '==', '미지정'))

    this.MapitemsCollection = this.afs
      .collection('Delivery')
      .doc(uid)
      .collection('D_orders', ref => ref.where('dstate', '==', '배송시작'));
  }

  //map 에는 D_orders
  mapFirstItems() {
```

```

var Ddb = this.MapitemsCollection;
var NewData = [];
return new Promise((resolve, reject) => {
  Ddb.get().subscribe(async items => {
    this.arr2 = await items.forEach(doc => {
      var NewData2 = [...NewData, doc.data()];
      NewData = NewData2;
    });
    console.log(NewData);
    resolve(NewData);
  });
});
}

getItems() {
  return this.ordersCollection.valueChanges({idField : 'id'})
}

getD_orders() {
  return this.D_ordersCollection.valueChanges({idField : 'Document_ID'})
}

/* scan값 비교 */
getCompareItems(profileName, scanResult, uid){
  var db = this.afs;

  this.ordersNotAssign.get().toPromise()
  .then(function(querySnapshot){
    querySnapshot.forEach(function(doc){

      if(doc.id == scanResult){
        //console.log("doc.ref.path : ",doc.ref.path) // order/document id값을 나타냄.

        //D_orders
        const DdocRef = db.collection('Delivery').doc(uid).collection('D_orders').doc(scanResult);
        DdocRef.set(doc.data())
        DdocRef.update({dname : profileName, dstate : "배송시작", deliverystart :
Timestamp.now(),lastUpdate:Timestamp.now()})

        //Orders
        const OdocRef = db.collection('orders').doc(scanResult)
        OdocRef.update({dname : profileName, dstate : "배송시작", deliverystart :

```

```

Timestamp.now(),lastUpdate:Timestamp.now()})

        //profile
        const PdocRef = db.collectionGroup('cart', ref => ref.where('dname', '==', '
미지정'))
        PdocRef.get().forEach(querySnapshot => {
            querySnapshot.docs.forEach(doc => {
                if(doc.id == scanResult){
                    const path = doc.ref.path
                    db.doc(path).update({dname : profileName, dstate : "배송시작",
deliverystart : Timestamp.now(),lastUpdate:Timestamp.now()})
                    //console.log("추가되었습니다.")
                    alert("배송지 추가가 완료되었습니다!")
                }
            });
        })
    }
})
})
return this.MapitemsCollection;
}
}

```



### 5.3 발표 자료

# QR코드를 이용한 택배 안심배송 서비스

2021.10.22

중부대학교 정보보호학과  
지도교수 : 이병천 교수님



1 조

권현주  
신세영  
유현진

## 목차

- ☐ 조원 편성
- ☐ 주제 선정
- ☐ 구상도
- ☐ 추진 경과
- ☐ 개발 환경 및 개발 내용
- ☐ 개발 시스템 운영
- ☐ 결론 및 기대 효과



## 조원 편성

이 름	역 할
권현주(조장)	DB 구축 및 관리, 백엔드 설계 및 구현
신세영	프론트·백엔드 설계 및 구현
유현진	DB 구축 및 관리, 프론트·백엔드 설계 및 구현, 보고서 작성



## 주제 선정

224만 유튜브, 택배 운송장 무심코 버렸다가...

개인정보 유출돼 곤혹...분리수거시

택배 운송장 연락처 보고 음란문자 보낸 70대 징역 6월

이세현 기자 · 입력 · 2019.12.04 16:25:47 · 수정

입력 2021.04.01 (10:33) · 수정 2021.04.01 (10:55) 930% 스테라

“눈 가리고 아웅”... CJ대한통운, 택배 개인정보 유출 ‘나 몰라라’

A 정영열 기자 · 입력 2021.09.14 08:00 · 댓글 1

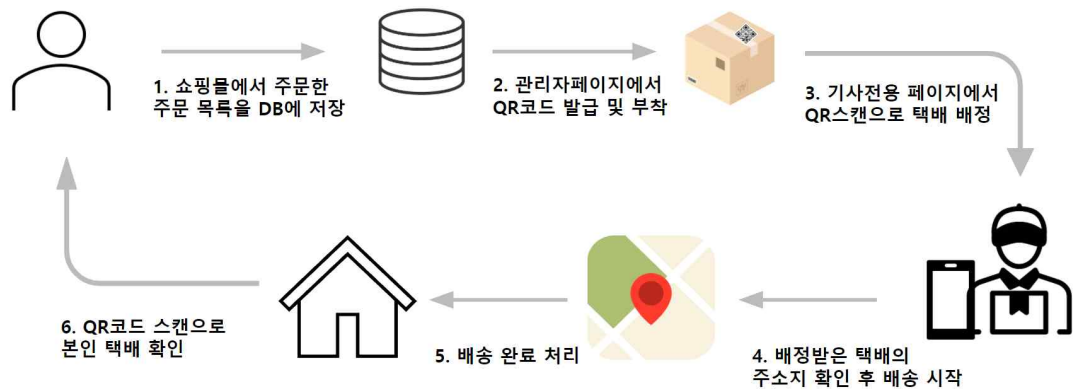
택배 운송장에 보낸사람·받는사람 이름, 주소, 휴대폰번호 그대로 노출  
스미싱 피해 무방비 노출... CJ측 “택배기사가 제거해야” 책임 떠넘기기

택배 운송장 연락처 보고 음란문자 보낸 70대 징역 6월

고객의 개인 정보 유출 방지를 위한 QR코드 안심배송 서비스 개발 추진



## 구상도



## 추진 경과

	3월	4월	5월	6월	7월	8월	9월	10월
자료조사 및 연구								
프론트엔드 개발								
백엔드 개발								
QR코드 생성, 인식/지도 시스템								
시스템 검증 및 보완								

## 개발 환경 및 개발 내용 (1/9)

### 개발 환경

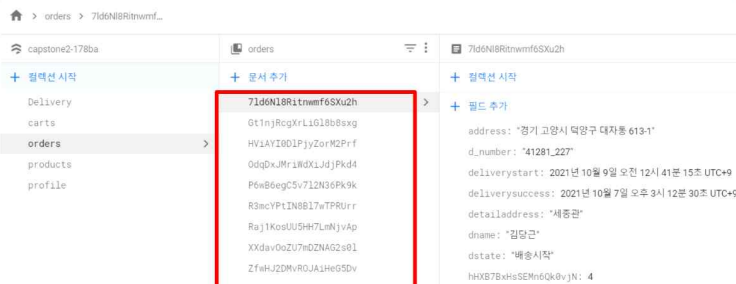


## 개발 환경 및 개발 내용 (2/9)

### 개발 내용(1/8)

```
import { QRCodeModule } from 'angularx-qrcode';
```

npm install angularx-qrcode  
명령어로 module 설치



NPM(Node Package Manager) :  
Node.js 기본 패키지 관리자

```
<tr *ngFor="let p of orders | async" class="{{p.dstate}}">  
<qr code [qrdata]="p.id" [width]="256" [errorCorrectionLevel]="M"></qr code>
```



## 개발 환경 및 개발 내용 (3/9)

### 개발 내용(2/8)

```
async startScan() {
  // Not working on iOS standalone mode!
  const stream = await navigator.mediaDevices.getUserMedia({
    video: { facingMode: 'environment' }
  });
  this.videoElement.srcObject = stream;
  // Required for Safari
  this.videoElement.setAttribute('playsinline', true);
  this.loading = await this.loadingCtrl.create({});
  await this.loading.present();
  this.videoElement.play();
  requestAnimationFrame(this.scan.bind(this));
}
```

카메라 권한을 허용하여 스캔

```
async scan() {
  if (this.videoElement.readyState === this.videoElement.HAVE_ENOUGH_DATA) {
    if (this.loading) {
      await this.loading.dismiss();
      this.loading = null;
      this.scanActive = true;
    }
    this.canvasElement.height = this.videoElement.videoHeight;
    this.canvasElement.width = this.videoElement.videoWidth;
    this.canvasContext.drawImage(
      this.videoElement,
      0,
      0,
      this.canvasElement.width,
      this.canvasElement.height
    );
    const imageData = this.canvasContext.getImageData(
      0,
      0,
      this.canvasElement.width,
      this.canvasElement.height
    );
    const code = jsQR(imageData.data, imageData.width, imageData.height, {
      inversionAttempts: 'dontInvert'
    });
    if (code) {
      this.scanActive = false;
      this.scanResult = code.data;
    } else {
      if (this.scanActive) {
        requestAnimationFrame(this.scan.bind(this));
      }
    }
  } else {
    requestAnimationFrame(this.scan.bind(this));
  }
}
```



## 개발 환경 및 개발 내용 (4/9)

### 개발 내용(3/8)

스캔값과 DB **profile/ (계정의uid) / cart(주문목록)**의 내용과 일치해야 데이터(이름, 주소, 전화번호)를 표시한다.

```
getOrders(){
  this.orderCollection= this.afs.collection('profile/${this.getUserUid()}/cart');
  return this.orderCollection.valueChanges({idField : 'id'});
}
```

```
<div *ngFor="let p of u_orders | async">
  <ion-card *ngIf="p.id == scanResult">
    <ion-card-header>
      <ion-card-title>배송내역 확인</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      받는분 성함 : {{p.name}} <br>
      받는분 주소 : {{p.address}} {{p.detailaddress}}
      배송상태 : {{p.dstate}}
      <ion-button (click)="showQrToast()">환불요청</ion-button>
    </ion-card-content>
  </div>
```

스캔 완료 후 표시되는 내용

### profile/ (계정의uid) / cart(주문목록)

profile	928VHywUSx0Qj15xcYeyqPvic112	cart
+ 문서 추가 78d04d0q6ZQu7HuaMM5xfC6zsS2 81C4ezGCKmaCvBAHCYos50H1QxX2 928VHywUSx0Qj15xcYeyqPvic112 BCf6xMzpG7N8BNDf6cgTS2Hh2t1 X0KomaLWpGpBytrfz1fF99C1eQW2	+ 컬렉션 시작 cart + 필드 추가 address: "경기 고양시 덕양구 대자동 613-1" detailaddress: "새동관" email: "aa@naver.com" name: "김준식" phoneNumber: "012-3456-7890" si: "41281"	+ 문서 추가 71d6N18R1tnnmf6SkU2h Gt1njRcgKRL1G18b8xq HV3AY18DIPjyZorM2Prf R3ecYPlIN8B17wTPRUrr Raj1KosUUSHH7LmHjvAp WZSMbgsWmtEORvGh8ec XKdavoZu7mdZNAG2s8I ZfwHJZDMvROJA1H6GSDv



## 개발 환경 및 개발 내용 (5/9)

### 개발 내용(4/8)

전체 주문 목록에서 스캔한 값과  
비교하여 같으면 기사에게 배정

```
<div *ngFor="let p of u_orders | async">
<ion-card *ngIf="p.id == scanResult">
  <h4>택배 코드 : {{p.d_number}} <ion-button class="add" (click)="showQrToast()">추가</ion-button></h4>

  async showQrToast() {
    const toast = await this.toastCtrl.create({
      message: '배송지를 추가하시겠습니까?',
      position: 'top',
      color: 'dark',
      duration: 5000,
      buttons: [
        {
          text: '저장',
          handler: () => {
            this.addD_orders();
          }
        }
      ]
    });
    toast.present();
  }

  addD_orders() {
    const uid = this.authService.getUserUid();
    this.D_items = this.itemService.getCompareItems(this.profil
  }

  getCompareItems(profileName, scanResult, uid) {
    var db = this.afs;

    this.ordersNotAssign.get().toPromise()
      .then(function(querySnapshot){
        querySnapshot.forEach(function(doc) {
          if(doc.id == scanResult){
            DdocRef.update({
              dname : profileName, dstate : "배송시작",
              deliverystart : Timestamp.now(),lastUpdate:Timestamp.now()});
          }
        });
      });
  }
}
```

배정된 시간, 배송상태,  
기사의 이름을 추가

## 개발 환경 및 개발 내용 (6/9)

### 개발 내용(5/8)

기사의 인증서 발급 여부  
확인후 인증서 발급

```
this.uid = this.authService.getUserUid()
this.afs.collection('Delivery').doc(this.uid).get().su
const key = Object.keys(doc.data())
console.log("key : ", key)
if( key.includes('privatekey')!= true){
  this.forgeservice.deliveryCert();
}

deliveryCert() {
  var forge = require('node-forge');

  //키쌍 생성
  let keyPair = forge.pki.rsa.generateKeyPair(2048);
  var publicKey = keyPair.publicKey;
  var privateKey = keyPair.privateKey;

  var publicKeyPem = forge.pki.publicKeyToPem(publicKey);
  var privateKeyPem = forge.pki.privateKeyToPem(privateKey);

  var caCert = forge.pki.createCertificate();
  caCert.publicKey = publicKey;
  // alternatively set public key from a csr
  //cert.publicKey = csr.publicKey;

  caCert.setSubject(this.caAttrs, "utf-8");
  caCert.setIssuer(this.caAttrs);
  caCert.setExtensions([{} // 인증기관의 인증
    {
      name: 'basicConstraints',
      cA: true
    }, {
      name: 'keyUsage',
      keyCertSign: true,
      digitalSignature: true,
      nonRepudiation: true,
      keyEncipherment: true,
      dataEncipherment: true
    }, {
      name: 'extKeyUsage',
      serverAuth: true,
      clientAuth: true,
      codeSigning: true,
      emailProtection: true,
      timeStamping: true
    }, {
      name: 'nsCertType',
      client: true,
      server: true,
      email: true,
      objsign: true,
      sslCA: true,
      emailCA: true,
      objCA: true
    }
  ], {
    //self-sign certificate
    caCert.sign(privateKey);
    localStorage.setItem('privatekey', privateKeyPem)
    localStorage.setItem('cert', caCertPem)
  });
}
```

## 개발 환경 및 개발 내용 (7/9)

### 개발 내용(6/8)

Kakao map에서 제공하는  
API를 기반으로 제작

```
for (let i = 0; i < length; i++) {
  setTimeout(() => {
    console.log();
    var name = arr1[i].name;
    var phone = arr1[i].phoneNumber;
    var address = arr1[i].address;
    var d_number = arr1[i].d_number;
    console.log(name);
    geocoder.addressSearch(address, function(result, status) {
      if (status === kakao.maps.services.Status.OK) {
        var addr = result[0].road_name;
        var coords = new kakao.maps.LatLng(result[0].y, result[0].x);
        var contents =
          '<div class="customover">' +
          '<div class="title" style="text-align: center;">' +
            d_number +
          '</div>' +
          '</div>';
        search_id.onclick = () => {
          var findID = this.findID;
          var search_result = arr1.find(value => value.d_number === findID);
          geocoder.addressSearch(search_result.address, function(result, status) {
            if (status === kakao.maps.services.Status.OK) {
              var moveLatLng = new kakao.maps.LatLng(result[0].y, result[0].x);
              map.setCenter(moveLatLng);
            }
          });
          this.renderMap(arr1, map, geocoder);
        };
      }
    });
  }, 1000);
}
```



## 개발 환경 및 개발 내용 (8/9)

### 개발 내용(7/8)

배송완료시 배송상태 수정

dstate: "배송완료"

```
complete(item){
  console.log("complete")
  if (item){
    this.itemService.D_ordersCollection.doc(item.Document_ID).update({dstate: "배송완료", deliverysuccess: true});
    this.itemService.ordersCollection.doc(item.Document_ID).update({dstate: "배송완료", deliverysuccess: true});
    this.afs.collectionGroup('cart', ref => ref.where('d', '==', item.Document_ID)).onSnapshot((snapshot) => {
      e => { e.docs.map(
        doc => { if (doc.id == item.Document_ID) {
          var path = doc.ref.path;
          // this.forgeservice.doc Cert(item.Document_ID)
          this.afs.doc(path).update({ dstate: "배송완료", signature: this.forgeservice.doc Cert(item.Document_ID).get('signature') });
        }
      )
    });
    this.showToast();
    if (item.dstate == "배송완료"){
      this.CheckToast();
    }
  }
}
```

기사정보, 운송장 번호,  
배송현황, 완료한  
시간에 대한 서명값을  
생성하여 저장





## 개발 환경 및 개발 내용 (9/9)

개발 내용(8/8)

배정취소시 배송상태 수정

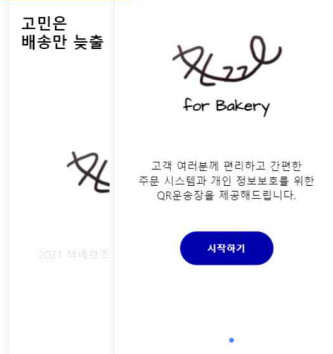
```
delete(item){
  console.log("delete")
  this.itemService.D_ordersCollection.doc(item.Document_ID).update({dname : "미지정", dstate : "배송준비", lastUpdate : Timestamp.now()})
  this.itemService.ordersCollection.doc(item.Document_ID).update({dname : "미지정", dstate : "배송준비", lastUpdate : Timestamp.now()})
  this.afs.collectionGroup('cart', ref => ref.where('dname', '!=', "미지정")).get().forEach(
    e => { e.docs.map(
      doc => { if (doc.id == item.Document_ID) {
        var path = doc.ref.path
        this.afs.doc(path).update({ dname : "미지정", dstate : "배송준비", lastUpdate : Timestamp.now() }) }
      })
    })
  return this.itemService.D_ordersCollection.doc(item.Document_ID).delete()
}
```



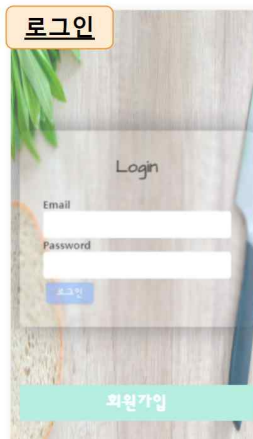
## 개발 시스템 운영 (1/12)

고객 페이지 - 로그인, 회원가입

초기화면



로그인



회원가입





## 개발 시스템 운영 (2/12)

고객 페이지 - 프로필, 기본 배송지

기본 배송지 설정x

김겨울 고객님의  
winter@season.com

기본 배송지 수정하기

기본 배송지 수정

주소  
경기 고양시 덕양구 대자동 613-1  
41281  
성세주소  
세종관 8층  
전화번호  
131-1111-1111

완료

다음 주소 API를 이용하여 주소 검색

기본 배송지  
설정 완료

김겨울 고객님의  
winter@season.com  
131-1111-1111  
경기 고양시 덕양구 대자동 613-1  
세종관 8층 831호

기본 배송지 수정하기

DB에 기본배송지 저장



## 개발 시스템 운영 (3/12)

고객 페이지 - 물품 주문 (1/2)

Order

Watzzo  
for Bakery

주문하실 상품을 선택해주세요

※ 오늘 하루! 무료배송 박스 도착!  
오전 10시 이전 주문 시 익일 도착!

크루아상 (Croissant)  
2,200원

애플파이 (Apple pie)  
8,000원

바게트 (Baguette)  
5,000원

클릭 시 상세정보  
열람 및 수량 선택

크루아상 (Croissant)  
2,200원

상세정보  
화제의 그 신작! 겹겹이 쌓인 크루아상을  
경험해보세요!

수량선택 + 1 -

기본 배송지가 등록된  
경우 기본 배송지로 설정

주문자 정보입력  
배송 정보를 입력해주세요

No.  
41281\_240

이름  
김겨울

배송주소  
경기 고양시 덕양구 대자동 613-1  
41281  
상세주소  
세종관 8층 831호

전화번호  
131-1111-1111



## 개발 시스템 운영 (4/12)

### 고객 페이지 - 물품 주문 (2/2) , 주문 목록

클릭 시 상세내용  
확인 가능

The screenshots show the customer interface for placing and tracking orders. The 'Order Confirmation' screen lists items like Croissant, Apple pie, Baguette, and Loaf bread. The 'Order' screen shows a success message. The 'Order List' screen shows a list of orders with status indicators like '배송 시작' (Delivery Start) and '배송 완료' (Delivery Complete). A red box highlights a specific order ID '41281\_240' in the list, which is then shown in a detailed view on the right.

**Order Confirmation**

주문목록 확인

- 크루아상 (Croissant) x 1
- 애플파이 (Apple pie) x 2
- 바게트 (Baguette) x 1
- 식빵 (Loaf bread) x 1

배송지 확인

이름: 김겨울

주: 경기 고양시 덕양구 대자동 613-1

상세주소: 세종관 8동 831호

전화번호: 131-1111-1111

**Order**

Success

주문이 완료되었습니다.

**Order List**

배송 시작

배송 완료

41281\_240 2021/10/18 1:45 AM

**Order Details**

41281\_240 2021/10/18 1:45 AM

주소: 경기 고양시 덕양구 대자동 613-1  
세종관 8동 831호

받는분 성함: 김겨울

받는분 전화번호: 131-1111-1111

기사님 성함: 미지정

배송 상태: 배송준비 2021/10/18 5:55 PM

주문 목록

- 크루아상 (Croissant) : 1
- 애플파이 (Apple pie) : 2
- 바게트 (Baguette) : 1
- 식빵 (Loaf Bread) : 1

주문요청



## 개발 시스템 운영 (5/12)

### 관리자 페이지 - 로그인, 메인 화면, 주문 목록

The screenshots show the admin interface. The '로그인' (Login) screen is on the left. The '메인 화면' (Main Screen) shows a dashboard with a 'Watzzo Admin page' header and a '주문정보 확인' (Check Order Information) section. The '주문 목록' (Order List) screen shows a table of orders with columns for name, phone number, address, order details, status, and delivery time.

**로그인**

**메인 화면**

**주문 목록**

Watzzo Admin page

주문정보 확인

이름	전화번호	주소	주문목록	주문상태	배달 기사	배송 상태	배송 완료
김준서	012-3456-7890	경기 고양시 덕양구 대자동 613-1 세종관	크루아상 (Croissant) : 2 애플파이 (Apple pie) : 1 바게트 (Baguette) : 1 식빵 (Loaf Bread) : 1	QR주문	송길동	배송완료	주문완료 (배송준비): 2021/10/17 11:23 PM 배송시작: 2021/10/18 9:22 PM 배송완료: 2021/10/18 9:42 PM
김산타	010-111-111	경기 고양시 덕양구 대자동 613-1 세종관 8동	크루아상 (Croissant) : 1 애플파이 (Apple pie) : 1	QR주문	미지정	주문완료	주문완료 (배송준비): 2021/10/18 17:36 AM



## 개발 시스템 운영 (6/12)

### 관리자 페이지 - QR 운송장

Watzzo Admin page  
주문정보 확인

이름	전화번호	주소	주문목록	운송장	배정 기사	배송 상태	배송 현황
김은식	012-3456-7890	경기 고양시 덕양구 대자동 613-1 세종관	크루아상 (Croissant) : 2 애플파이 (Apple pie) : 바게트 (Baguette) : 식빵 (Loaf Bread) :	QR운송장		배송완료	주문완료 (배송준비); 2021/10/17 11:23 PM 배송시작; 2021/10/18 6:03 PM 배송완료; 2021/10/18 6:03 PM

운송장인쇄

window.print() 함수  
실행으로 인쇄 가능



## 개발 시스템 운영 (7/12)

### 관리자 페이지 - 기사 계정 생성

기사 정보

기사 계정 생성

계정 이메일  
jb@univ.com

비밀번호  
6자리 이상의 비밀번호를 입력해주세요.

이름

전화번호

기사계정

이정원

091-111-111

jb@univ.com

기사 정보 입력

이름  
이정원

전화번호  
091-111-111

완료

기사 계정 생성 시 table에 표시

이름	전화번호	기사계정
이정원	091-111-111	jb@univ.com
홍길동	010-1234-5678	honggildong@naver.com
홍반장	01023456097	sy@naver.com

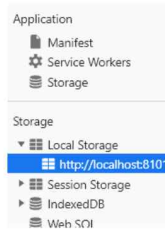
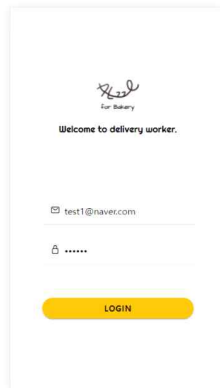
계정 만들기

회원가입

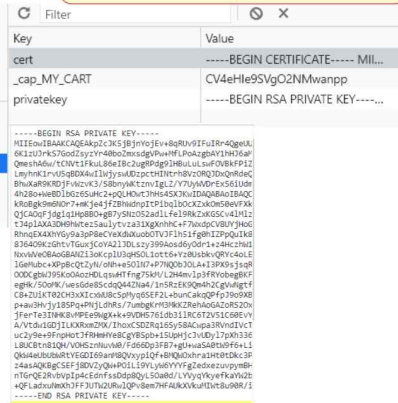


# 개발 시스템 운영 (8/12)

## 기사 페이지 - 로그인



## 기사의 개인키와 공개키 저장



# 개발 시스템 운영 (9/12)

## 기사 - 배송 배정, 배송 목록



추가 버튼 클릭 시  
배송 목록에 표시

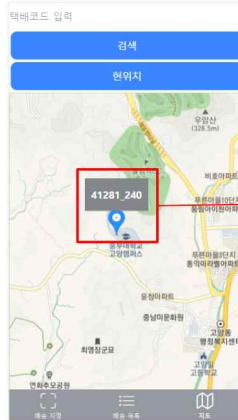


클릭 시 상세 내용 확인 가능  
배송상태에 따라 색을 다르게 표시

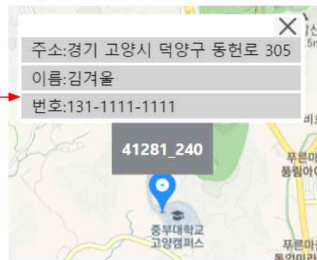


## 개발 시스템 운영 (10/12)

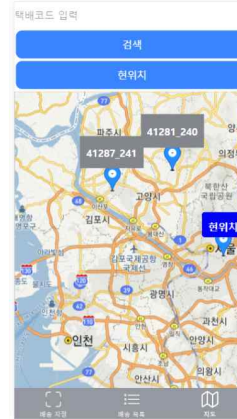
### 기사 페이지 - 배송지 확인



마커 클릭 시 상세 주소  
확인 가능



검색 클릭 시 해당  
주소지가 중심부로 표시



## 개발 시스템 운영 (11/12)

### 기사 페이지 - 배송 상태 변경



#### 배송 목록

41281_227	경기 고양시 덕양구 대자동 613-1세종관
41287_241	경기 고양시 일산서구 대화동 22392동
41281_240	경기 고양시 덕양구 대자동 613-1세종관 8층 831호

배송완료 시 체크 표시를  
클릭하여 DB 배송상태를  
배송완료로 변경,  
서명값을 생성하여 DB에 저장

#### 배송 목록

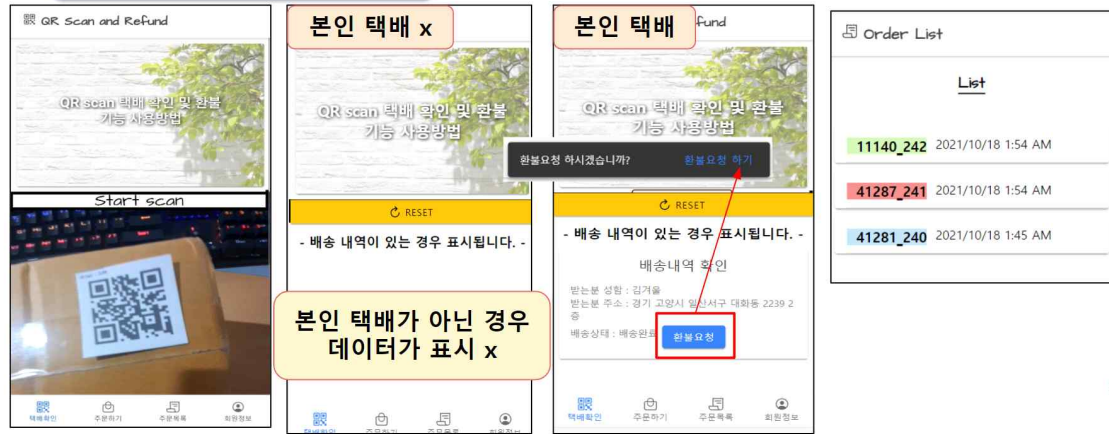
41281_227	경기 고양시 덕양구 대자동 613-1세종관
41281_240	경기 고양시 덕양구 대자동 613-1세종관 8층 831호

배송배정 스캔을 잘못  
진행한 경우 배정 취소



## 개발 시스템 운영 (12/12)

### 고객 페이지 - 배송 확인, 환불



## 결론 및 기대 효과

### □ 결론

- 기존의 운송장인 아닌 QR코드를 사용하여 일차적으로 쉽게 노출되는 부분을 보완하여 개인 정보보호를 강화하는데 성공
- 전자서명을 이용하여 고객-기사의 부인방지를 보장한다.

### □ 기대효과

- 개발 과정 중 어려운 부분은 기술 공유 및 협동으로 팀워크를 향상시키고 모든 조원들이 개발 과정에 참여하며 시스템 기획 및 프로그램 개발 역량 향상
- 2021.08.22 개인정보위에서 택배 운송장 개인 정보보호 강화 대책을 논의했다. 이에 개인 정보보호에 대한 중요성을 다시 한번 인식. 끝.



**감사합니다.**

