

QR코드와 인증서를 이용한 간편 결제 시스템 구현

팀 명 : Coconut Pay

지도 교수 : 이병천 교수님

팀 장 : 박종훈

팀 원 : 권혁민

마민기

윤정민

2019. 10.

중부대학교 정보보호학과

목 차

1. 서론

1.1 연구 목적 및 주제 선정	2
-------------------------	---

2. 관련 연구

2.1 기존 간편결제 지불 시스템 분석	3
2.2 구현 환경	5
2.2.1 Javascript	5
2.2.2 Node.js	5
2.2.3 MySQL	5
2.2.4 Vue.js	5
2.2.5. QR Code	6
2.2.6 Heroku	6
2.2.7 JWT	6
2.2.8 X.509 인증서	6

3. 본론

3.1 시스템 구성	7
3.1.1 참여자 별 설계	8
3.1.2 기능 별 설계	10
3.2 DB 자료구조 설계	14
3.3 프로그램 구성	17
3.3.1 웹페이지 기능	20
3.3.2 인증서 발급 기능	25
3.3.3 QR Code 결제 기능	26
3.3.4 전자 서명 기능	34

4. 결론

4.1 결론	38
4.2 기대효과	38
4.3 향후 과제	38

5. 별첨

5.1 참고자료	38
5.2 소스코드	38
5.3 발표 PPT 자료	181

1. 서 론

1.1 연구 배경

현재 결제대행 시장에서 가장 많이 쓰이는 결제방식은 여러 플러그인 및 프로그램을 설치해야 하거나 휴대폰 어플리케이션을 설치해야하는 등의 번거로운 과정을 거쳐야 하는 불편함을 해소하고자 QR코드를 이용한 간편 결제 시스템 개발을 시작하게 되었다.

1.2 연구 필요성

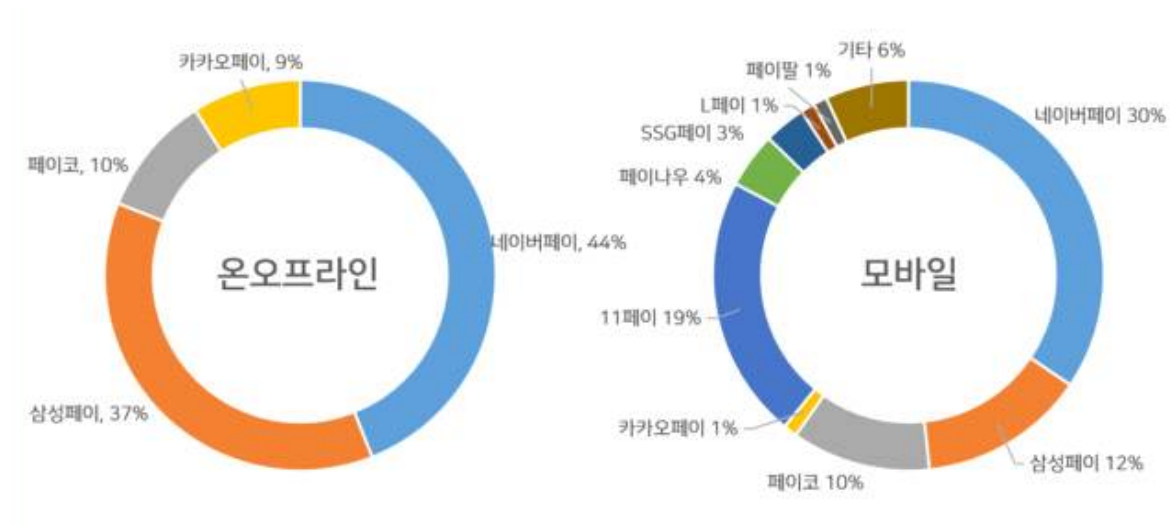
QR코드는 1차원 바코드와는 다르게 많은 데이터를 담을 수 있고, 오류 정정기능, 디자인 분야에서도 많은 이점을 가지기 때문에 국내뿐만 아니라 전 세계적으로 마케팅, 안내, 디자인 등의 여러 분야에서 쓰이는 기술임. 그러므로 카메라로 찍어 결제버튼만 누르면 결제가 진행되는 간편 결제 시스템에 도입하는 것이 매우 중요하다고 판단하였다. 또한, 편리함의 증가로 인해 발생하는 취약점을 보완하기 위해 X.509 기반의 인증서를 활용하여 사용자 및 거래를 검증하여 더 안전한 거래를 보장할 수 있다.

1.3 연구 목표

현재 사용 중인 대부분의 결제대행 서비스는 여러 플러그인 및 프로그램을 설치해야 하므로 DNS Poisoning 공격에 노출 시 변조된 플러그인 사이트로 유도 되어 악성코드가 설치되어 공격자가 의도적으로 사용자의 정보를 수집하거나(Troijan) 악의적인 목적으로 사용하는 경우(DDoS)가 발생할 수 있다. 이를 방지하기 위해 별도의 플러그인 및 프로그램을 설치할 필요 없이 모든 PC 및 모바일 기기에 기본으로 탑재되어 있는 브라우저를 이용하여 QR코드를 촬영 후 서버로 정보를 보냄으로써 이용자들의 PC가 좀비 PC가 될 여지를 줄일 수 있다. 뿐만 아니라 최초 결제 시에도 설치 후 브라우저를 재시작 해야 하는 플러그인을 설치하지 않아 사용자 입장에서 결제하던 브라우저 창을 닫지 않고 결제가 되므로 더욱 좋은 편의성을 제공할 수 있다. 흔히 편리함이 늘면 보안에 취약점이 생기는 경우가 많은데, 해당 연구에서는 늘어난 편리함으로 인해 취약해지는 보안을 보완하기 위해 X.509 기반 PKI 인증서를 활용한 전자서명 및 검증 기술을 이용하여 검증된 사용자인지, 검증된 거래인지 확인도 가능하게 개발한다. 또한 현업에서 많이 사용 중인 Node.js, MySQL, Vue.js 같은 기술의 숙련도를 높이기 위해 웹서비스를 개발하는 경험을 쌓는 기회가 될 수 있다. 마지막으로 안정적이고 연속적인 서비스를 제공하기 위해 클라우드 서버에 업로드 하여 지속적으로 모니터링 및 유지, 보수를 하면서 나아가 가능성이 있다면 창업을 하여 새로운 시장의 가능성을 열 수 있도록 만드는 것이 이번 연구의 목표이다.

2. 관련연구

2.1 기존의 결제방식과의 차이점



[그림 2-1] 국내 간편결제 시장 점유율

2.9.1 기존의 간편 결제 지불 시스템

국내에서 가장 많이 쓰이는 간편 결제 지불 시스템은 하드웨어 방식인 삼성페이를 제외하고 네이버페이가 온, 오프라인에서는 44%의 점유율을, 모바일에서는 30%의 점유율을 차지하고 있다. 네이버페이는 네이버페이 페이지에서 카드번호와 유효기간 등의 정보를 입력 후 등록하여 사용이 가능하다. 그러나 간편한 만큼 카드를 도난 및 분실하게 되면 해당 카드를 가진 사람이 사용 가능하다는 단점이 있다. 그러므로 카드나 휴대폰을 분실하더라도 인가받지 않은 사람이 결제를 하지 못하도록 하는 안전장치를 추가하면서 편의성면에서 생기는 불편함이 없도록 만드는 것이 본 과제의 목표이다.

2.9.2 본 과제에서 사용된 보안 기술

2.9.2.1 이중토큰

JWT 토큰을 이용하여 무상태 인증을 제공할 수 있도록 개선한 기술이다. JWT방식의 토큰인증은 서버가 발급하는 토큰을 서비스 요청 시마다 서버에 반복해서 전송하는 방식으로 스니핑 공격에 취약하므로 HTTPS와 함께 사용되어야 한다는 제약이 있다. 사용자가 서버에 초기인증에 성공하면 서버는 추후 인증하기 위해 사용자에게 공개토큰과 비밀토큰을 발급하고 토큰 발급 및 인증에 사용할 비밀키를 생성한다. 공개토큰에는 사용자 정보를 HMAC 서명값을 생성하고 포함시킨다. 이후 공개토큰에 대한 HMAC 서명값을 생성하고 비밀토큰에 포함시킨다. 공개토큰은 인증된 사용자임을 나타내기 위해 서버에 반복해서 전송하는 정보로서 서명된 ID와 같은 역할을 하며 비밀토큰은 외부로 노출되지 않고 난수화 인증 정보 계산에만 사용하는 정보로 서명된 패스워드와 같은 역할을 한다. 두

토큰은 상호 연관되어있어 비밀키를 알고 있는 서버는 공개토큰이 주어진다면 언제든지 비밀토큰을 계산할 수 있으므로 검증이 가능하기 때문에 비밀토큰을 관리할 필요가 없다. 클라이언트는 서버가 발급해준 두 개의 토큰을 브라우저의 저장소에 저장하고 사용한다.

2.9.2.2 마스터인증서/추가인증서

사용자가 최초로 발급받은 인증서로 사용자의 메인PC에 저장되는 인증서를 마스터인증서라고 부른다. 차후에 다른 모바일 기기나 PC에도 인증받기 위해 인증서를 발급받아 저장하는데 이 인증서는 추가인증서라고 불리며 마스터 인증서를 이용하여 서명하게 된다. 즉, 마스터 인증서의 서명주체는 서버이고 발급주체는 사용자 메인 PC이다. 추가인증서의 서명주체는 사용자의 마스터 인증서이고 발급주체는 사용자의 모바일 기기나 메인 PC를 제외한 다른 PC들이다.

2.9.3 결제시스템에 대한 공격 및 대응 모델

공격자 입장에서는 가장 일반적으로 SQL 인젝션이나 크로스 사이트 스크립팅 공격을 시도할 수 있다. SQL 인젝션은 클라이언트 입력 데이터를 조작하여 서버의 데이터베이스 테이블을 삭제하거나 데이터를 변조하는 공격이다. 가장 흔한 방법은 WHERE 절을 참으로 만들어 높은 권한의 계정을 탈취하거나 데이터베이스 내부의 구조나 데이터를 조회할 수 있는 방법이 있다. 크로스 사이트 스크립팅은 자바스크립트를 사용한 환경에서 공격하는 경우가 많은데, 여러 사용자가 접근 가능한 게시판 등에 공격의 서버 등에 쿠키정보같은 정보를 전송하는 악의적인 코드를 삽입하여 사용자가 게시판에 접근하거나 게시판에 글을 쓸 때마다 정보를 공격자에게 전송하는 방법이 가장 흔하게 사용된다. 이 SQL 인젝션과 크로스 사이트 스크립팅 등의 공격기법은 공격방법이 단순하고 가장 기초적이지만 그에 대한 피해는 공격 난이도에 비해 심각한 수준이고 많은 사이트들이 이에 대한 대비를 하지 않아 공격을 받는 경우가 많다. 그러나 크로스 사이트 스크립팅 공격을 할 경우, 일반적으로 입력 폼에 alert('xss')를 입력하는데 Vue.js로 제작된 프론트 페이지에서 입력 폼에 해당 함수를 입력 시 TypeError를 출력한다. 그 이유는 Vue 인스턴스에서 해당 expresssion을 찾지만 alert는 자바스크립트의 함수이기 때문에 작동하지 않는다. 또한 본 사이트에서는 XSS에 취약한 HTML 동적 렌더링을 사용하지 않기 때문에 다른 사이트에 비해 상대적으로 XSS에 대한 대비가 되어있다. SQL 인젝션 공격의 경우에는 본 사이트의 서버에서는 Node.js를 사용한다. Node.js에 적용할 수 있는 패키지가 많은데 본 사이트에서는 그 중에 SQL DB 서버에 연결할 수 있는 패키지인 node-mysql을 사용한다. node-mysql은 sql문을 점검하여 SQL 인젝션을 방지할 수 있다.

2.2 구현 환경

2.2.1 Javascript

Javascript는 1995년 브렌던 아이크가 설계하여 Netscape Communications, Mozilla Foundation, ECMA International에서 공동으로 개발한 객체기반의 스크립트 프로그래밍 언어이다. 웹 브라우저 내에서 주로 사용하며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 갖고 있다. 또한, 런타임 환경과 같이 서버 사이드 네트워크 프로그래밍에도 사용되고 있다. C언어의 기본 구문에 바탕을 뒀기 때문에 C언어와 구문이 유사하다. 현재 2019년 기준으로 ECMA 262 10th Edition이 최신 버전이다.

2.2.2 Node.js

Node.js는 2009년에 V8 엔진으로 빌드 되어 Ryan Dahl에 의해 개발된 확장성 있는 서버 사이드 개발에 사용되는 소프트웨어 플랫폼이다. 웹 서버와 같이 확장성 있는 네트워크 프로그램 제작을 위해 고안되었다. 작성 언어로 자바스크립트를 활용하며, Non-blocking 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. 현재 2019년 기준으로 V10(코드명:Dubnium) 버전이 가장 최신버전이다.

2.2.3 MySQL

MySQL은 1995년에 MySQL AB에 의해 개발된 오픈소스 관계형 데이터베이스 관리 시스템이다. 다중 스레드, 다중 사용자 형식의 구조질의어 형식의 데이터베이스 관리 시스템으로서 오라클이 관리 및 지원하고 있으며, Qt처럼 이중 라이선스가 적용된다. 하나의 옵션은 GPL이며 GPL 이외의 라이선스로 적용시키려는 경우 전통적인 지적재산권 라이선스의 적용을 받는다.

2.2.4 Vue.js

Vue.js는 Evan You에 의해 개발되어 2014년에 발표된 웹 어플리케이션의 사용자 인터페이스를 만들기 위해 사용하는 오픈 소스 프로그레시브 자바스크립트 프레임워크이다. 코어 라이브러리는 선언형 렌더링과 컴포넌트 구성에 초점을 두며 기존 페이지에 임베드가 가능하므로 점진적으로 채택 가능한 구조를 갖추고 있다. 라우팅 상태관리, 빌드 도구화와 같이 복잡한 어플리케이션에 필요한 고급 기능들은 공식저금로 유지 보수되는 지원 라이브러리와 패키지를 통해 제공된다.

2.2.5 폼 입력 바인딩

Vue.js에서 지원하는 기술로, V-model 디렉티브를 사용하여 폼 input과 textarea 엘리먼트에 양방향 데이터 바인딩을 생성할 수 있다. 입력 유형에 따라 엘리먼트를 업데이트 하는 올바른 방법을 자동으로 선택한다.

2.2.6 QR Code

QR Code는 1994년 일본의 토요타 자동차의 자회사인 덴소 웨이브가 토요타 자동차만의 전용 차 키와 부품을 구별하고자 개발하여 사용하였다. QR Code는 흑백 격자무늬 패턴으로 정보를 나타내는 매트릭스 형식의 이차원 바코드이다. 주로 한국, 일본, 중국, 영국, 미국 등에서 많이 사용되며 명칭은 덴소 웨이브의 등록 상표 'Quick Response'에서 유래하였다. 종래에 많이 쓰이던 바코드의 용량제한을 극복하고 그 형식과 내용을 확장한 2차원의 바코드로 종횡의 정보를 가져서 숫자 외에 문자의 데이터를 저장할 수 있다.

2.2.7 Heroku

Heroku는 2007년 개발이 시작되어 어플리케이션 배치 모델로 사용되는 여러 프로그래밍 언어를 지원하는 클라우드 Paas이며 최초의 클라우드 플랫폼들 가운데 하나이다. 개발 당시에는 프로그래밍 언어를 Ruby만 지원했으나 지금은 JAVA, Node.js, Scala, Clojure, Python, PHP, Go를 지원한다. 이러한 이유로 헤로쿠는 개발자가 모든 언어 간 비슷한 방식으로 애플리케이션들을 빌드, 실행하고 스케일링할 수 있게 하므로 헤로쿠는 폴리글롯 플랫폼으로 간주된다.

2.2.8 JWT

JWT는 Json Web Token 의 약자로 2015년에 IETF에 의해 웹 표준 RFC7519로 정의되었다. C, JAVA, Python, C++, R, C#, PHP, Javascript, Ruby, Go, Swift 등의 프로그래밍 언어에서 지원되는 기술이다. 정보가 헤더, 페이로드(정보), 서명으로 나뉘어져 있기 때문에 모든 정보를 자체적으로 가지고 있다.

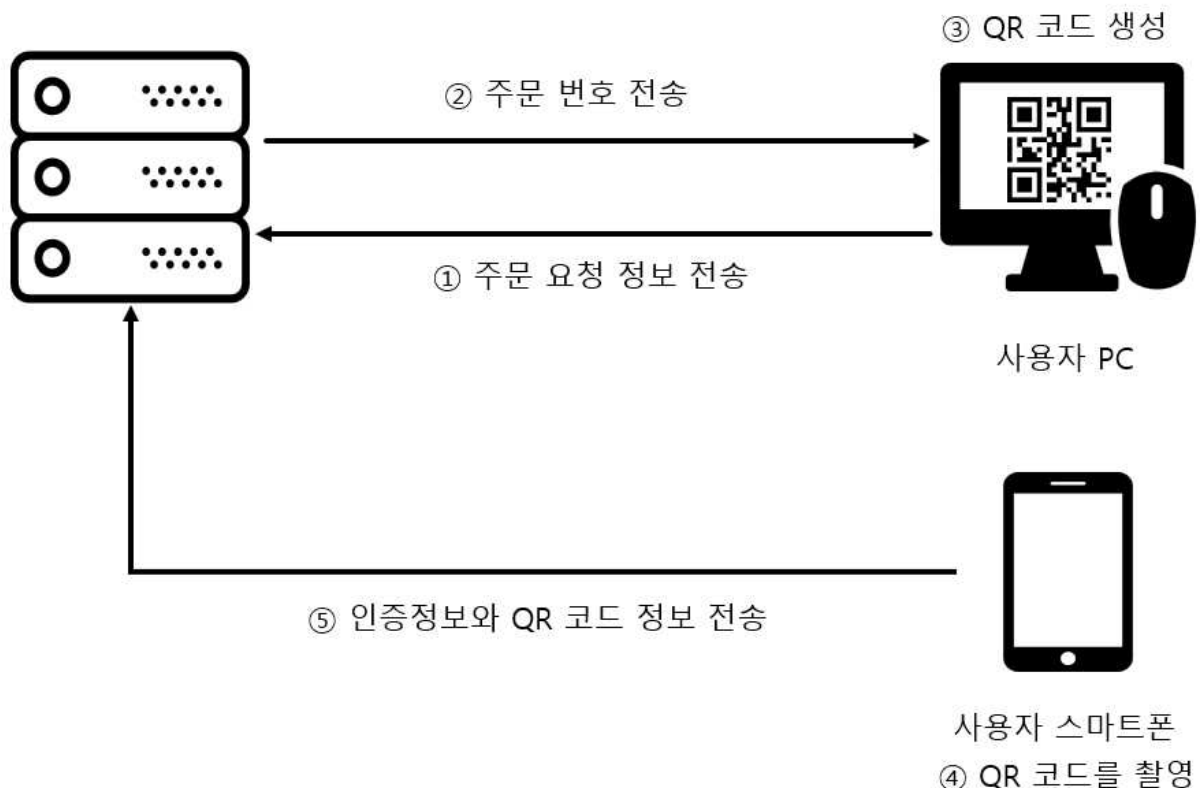
2.2.9 X.509 인증서

X.509 인증서는 암호학에서 공개키 인증서와 인증 알고리즘의 표준 가운데에서 공개키 기반의 ITU-T 표준이다. X.509 인증서는 1988년 X.500 표준안의 일환으로 시작되어 1993년에 인증기관 고유 식별자가 추가된 v2가 발표되었으며, 1996년에 확장 기능을 이용해 데이터를 추가할 수 있는 v3가 발표되어 현재 쓰이고 있다. PGP처럼 상호 신뢰를 기반으로 하는 웹 모델과 달리, X.509는 매우 엄격한 수직적 구조를 채택하였다. 따라서 하나의 인증 기관을 정점으로 하는 트리 구조를 갖게 된다. 이러한 형태의 불편함을 해소하기 위해, v3는 보다 유연한 구조를 채택하여 몇 개의 신용할만한 Root CA끼리는 상호 인증, 혹은 자가 인증을 허용하고 있다.

3. 본론

3.1 시스템 구성

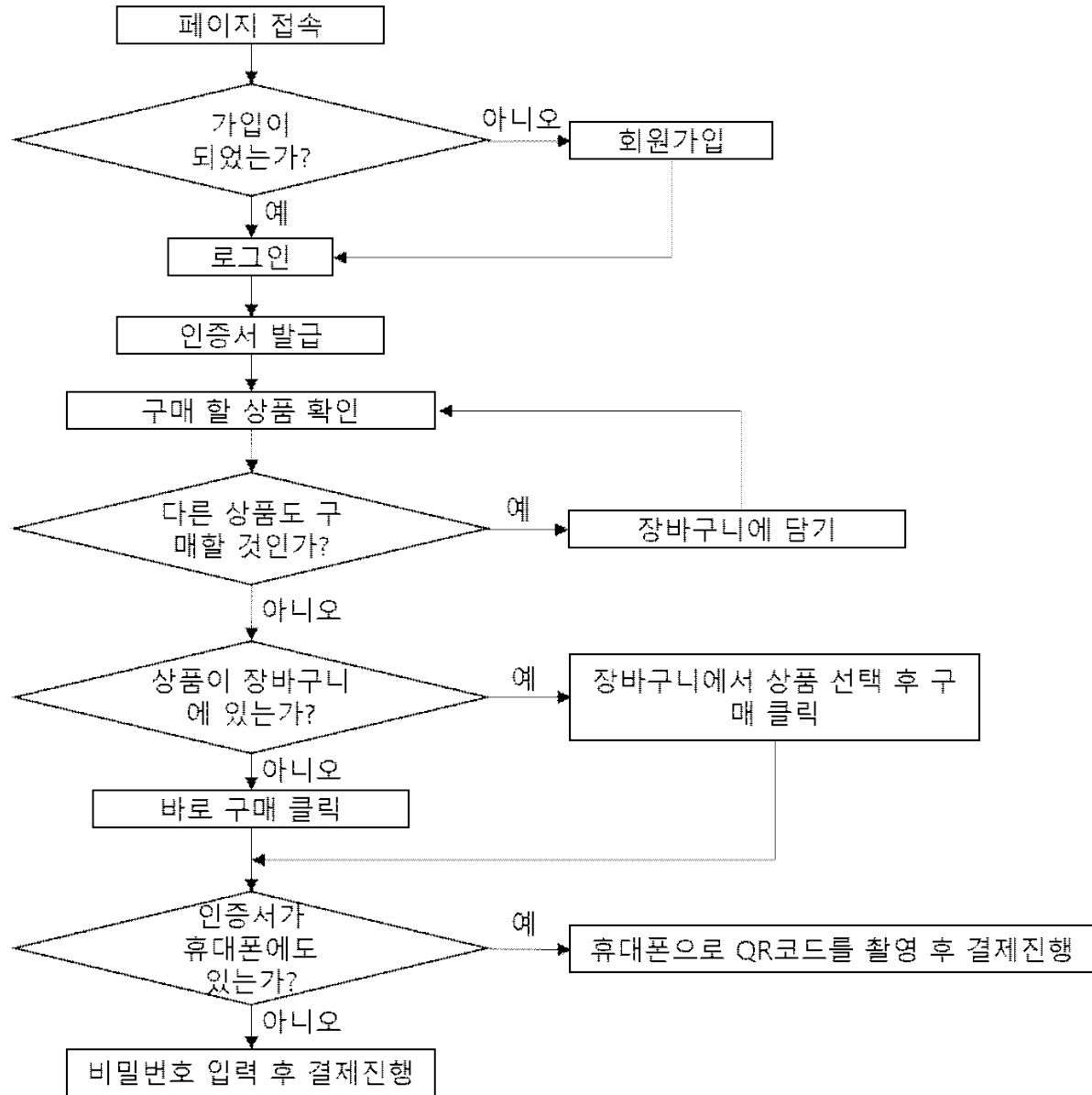
일반적인 쇼핑몰 사이트를 만들어 결제시스템을 만드는 것이 우선순위가므로 회원가입 및 로그인을 지원하는 쇼핑몰 사이트를 만들었다. 회원가입 후 로그인을 하면 각 사용자마다의 고유한 인증서를 저장할 수 있는 인증서 테이블과 사용자가 선택한 상품을 저장할 수 있는 장바구니 테이블이 만들어진다. 사용자는 회원가입을 하여 만든 계정을 이용하여 로그인 하면 인증 유지를 위해 서버에서는 현재시간과 의사난수 값을 이용하여 인증토큰을 만들어 사용자의 브라우저의 로컬스토리지에 저장하게 된다. 이 토큰은 1주일 간 유지되며 1주일이 지나서 검증을 하게 된다면 토큰을 폐기하게 된다. 사용자는 원하는 상품을 바로 결제 버튼을 누르거나 장바구니에 상품을 담아 차후에 구매할 상품을 선택하여 결제버튼을 누르면 결제페이지로 이동한다. 결제 버튼을 누르면 주문 요청 정보를 서버로 보내 서버에서는 데이터베이스에 주문정보를 만들고 결제여부에 false 값을 넣는다. 서버에서는 만든 주문 정보에서 주문번호를 프론트페이지로 보내 프론트페이지에서는 현재시간과 서버에서 받은 주문번호를 이용하여 QR Code를 만들게 된다. 사용자는 스마트폰으로 해당 쇼핑몰의 모바일 페이지의 결제 페이지로 가서 카메라를 활성화 시키고 QR Code를 찍으면 QR Code의 정보와 사용자의 인증 정보를 서버로 보내 서버에서는 해당 값들을 검증 후 정상적인 요청일 시 결제를 진행한다.



[그림 3-1] 시스템 시나리오

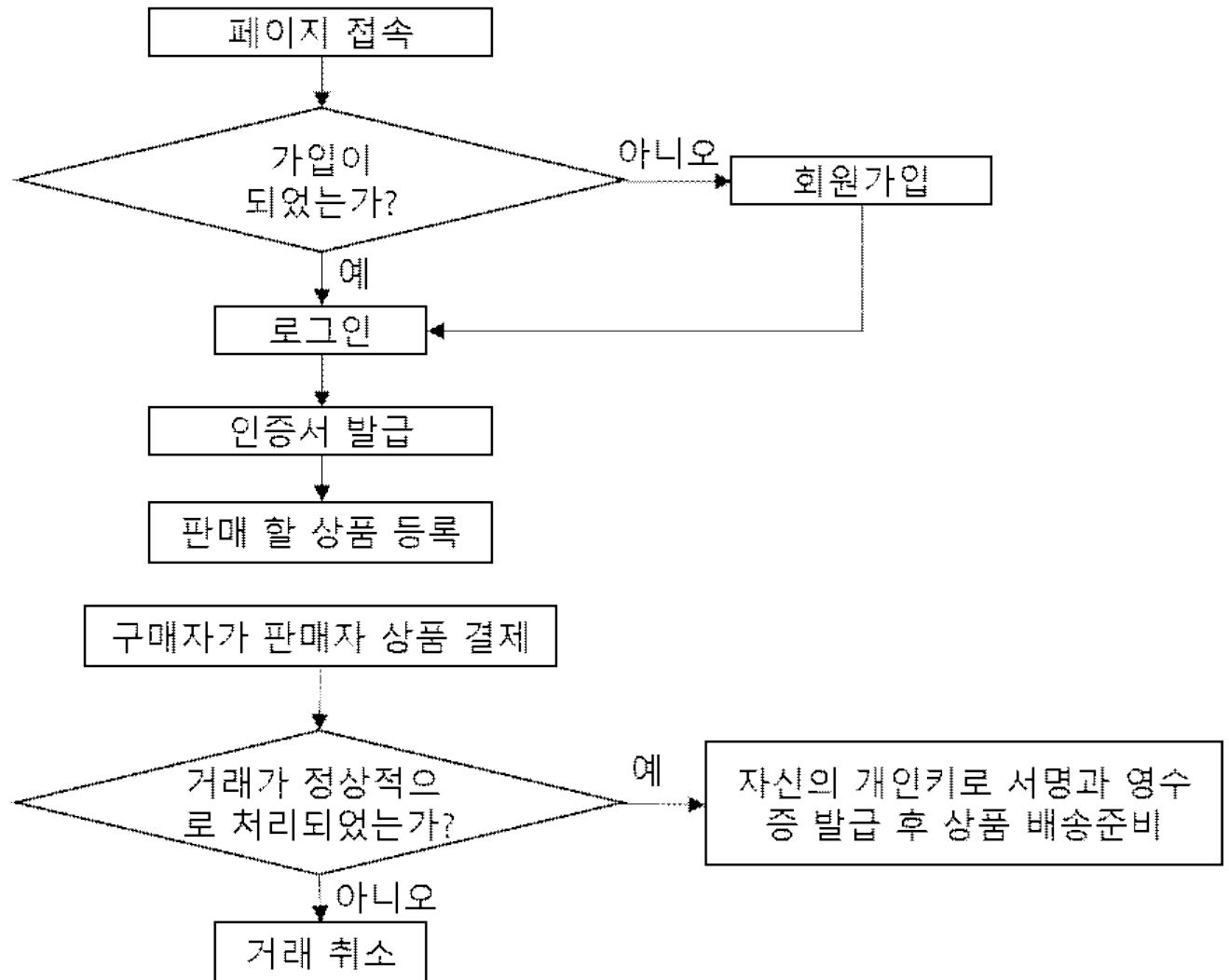
3.1.1 참여자 별 설계도

- 구매 과정 설계도



[그림 3-2] 구매과정 시나리오

-판매 과정 설계도



[그림3-3] 판매 과정 시나리오

3.1.2 기능 별 설계도

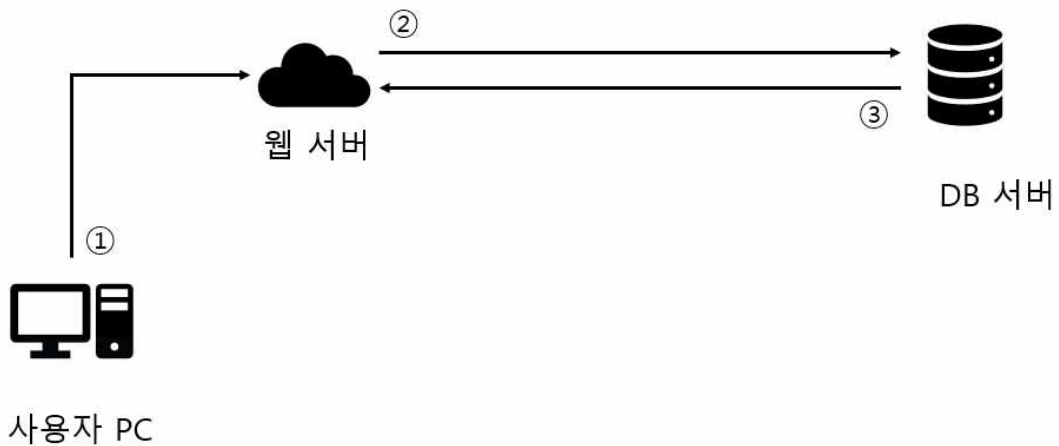
- 회원가입



[그림 3-4] 회원가입 시나리오

- ① 사용자는 회원가입 정보를 모두 입력 후 회원가입 버튼을 클릭하여 서버로 정보를 전송한다.
- ② 서버는 사용자의 정보를 쿼리문으로 작성하여 DB서버에 쿼리 요청을 전송한다.
- ③ DB서버는 요청받은 쿼리를 실행시킨 후 성공, 실패 여부를 서버로 전송한다.

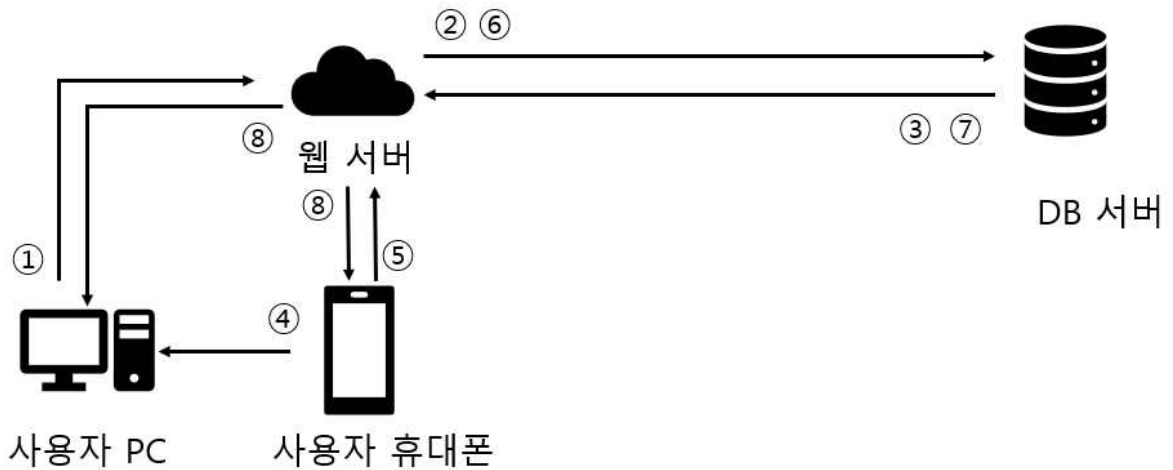
- 로그인



[그림 3-5] 로그인 시나리오

- ① 사용자는 회원가입할 때 입력한 정보를 로그인 페이지에서 입력 후 서버로 전송한다.
- ② 서버는 사용자의 로그인 정보를 쿼리문으로 작성하고 DB서버로 전송한다.
- ③ DB서버는 요청받은 쿼리를 실행시킨 후 성공, 실패 여부를 서버로 전송한다.

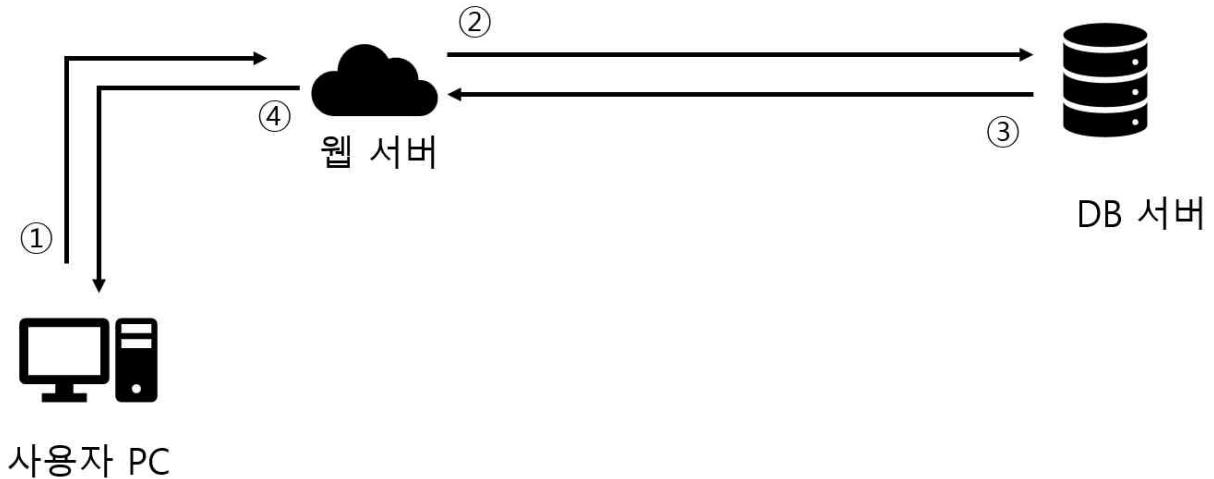
- 상품 구매



[그림 3-6] 상품 구매 시나리오

- ① 사용자는 구매할 상품을 선택하여 웹서버로 전송한다.
- ② 서버는 사용자가 전송한 상품 정보로 쿼리문을 작성하여 DB서버로 쿼리를 전송한다.
- ③ DB서버는 요청받은 쿼리문을 실행하여 처리결과를 서버로 전송한다.
- ④ 서버는 사용자 PC에 QR코드를 출력하고 사용자는 QR코드를 휴대폰으로 촬영한다.
- ⑤ 사용자는 휴대폰으로 결제 버튼을 누르고 개인키를 복호화 할 비밀번호를 입력하여 서버로 전송한다.
- ⑥ 서버는 사용자가 전송한 정보가 맞는지 검증을 한 후 결제를 진행할 쿼리문을 DB서버로 전송한다.
- ⑦ DB서버는 쿼리문을 실행시킨 후 결과를 서버로 전송한다.
- ⑧ 서버는 정상적으로 쿼리가 실행되었다면 사용자 휴대폰과 사용자 PC에서 결제완료 페이지로 리다이렉션한다.

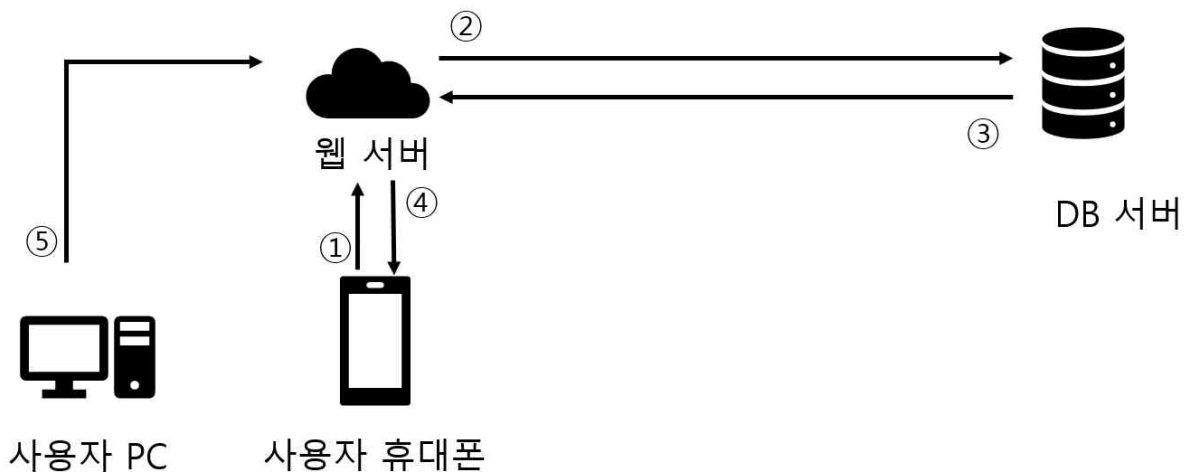
- 인증서 발급



[그림 3-7] 인증서 발급 시나리오

- ① 사용자는 PC로 인증서를 발급 시 개인키를 암호화 할 비밀번호를 입력하여 서버로 전송한다.
- ② 서버는 암호 라이브러리를 이용하여 공개키 쌍을 만들고 그 키 쌍중에 개인키는 사용자가 전송한 비밀번호를 대칭키로 사용하여 대칭키 암호화를 하고 공개키로는 인증서를 생성하여 인증서를 PEM 형식으로 DB에 저장하는 쿼리문을 만들어 DB서버로 전송한다.
- ③ DB서버는 전송받은 쿼리문을 실행하여 실행결과를 서버로 전송한다.
- ④ 서버는 인증서를 DB에 저장되었다면 암호화된 개인키와 인증서를 브라우저의 로컬스토리지에 저장한다. 이 인증서는 마스터 인증서가 된다.

- 마스터 인증서, 추가 인증서



[그림 3-8] 마스터, 추가 인증서 시나리오

- ① 사용자는 휴대폰으로 추가 인증서를 발급받기 위해 개인키를 암호화 할 비밀번호를 입력하여 서버로 전송한다.
- ② 서버는 공개키쌍을 생성하고 사용자가 입력한 비밀번호로 개인키를 암호화하고 공개키로는 인증서를 생성하여 인증서를 PEM 형식으로 저장할 쿼리문을 만들고 DB에 전송한다.

- ③DB는 전송받은 쿼리문을 실행하여 쿼리문 실행 결과를 서버로 전송한다.
- ④서버는 인증서를 DB에 저장되었다면 암호화된 개인키와 인증서를 브라우저의 로컬 스토리지에 저장한다. 이 인증서는 추가 인증서가 되며 마스터 인증서로부터 승인을 받지 못한다면 사용이 불가능하다.
- ⑤ 발급된 추가 인증서는 아직 승인을 받지 못했으므로 사용자는 마스터인증서가 저장되어있는 PC에서 해당 추가 인증서를 승인한다. 승인된 정보는 서버를 거쳐 DB로 전송이 된다.

3.2 DB 자료구조 설계

- 회원 테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
<input type="checkbox"/> 1	account	varchar(20)	utf8_general_ci		예	NULL		
<input type="checkbox"/> 2	addr	varchar(45)	utf8_general_ci		예	NULL		
<input type="checkbox"/> 3	email	varchar(45)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 4	id	varchar(15)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 5	indi	tinyint(4)			아니오	없음		
<input type="checkbox"/> 6	money	int(15)			아니오	0		
<input type="checkbox"/> 7	name	varchar(20)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 8	number	int(11)			아니오	없음		AUTO_INCREMENT
<input type="checkbox"/> 9	password	varchar(255)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 10	tel	varchar(11)	utf8_general_ci		아니오	없음		

[그림 3-9] user 테이블

사용자가 가입 시 입력하는 데이터이다. 각 사용자를 식별하기 위한 기본키인 number 속성은 자동으로 1이 더해지게 되는 AUTO_INCREMENT를 적용하였다. 기본키는 각 데이터 열을 식별할 수 있는 값으로, 중복이 불가능하고 Null 값을 가질 수 없다.

AUTO_INCREMENT는 테이블에 데이터가 삽입될 때 자동으로 1씩 증가하는 기능이다. 최초로 삽입된 데이터는 1이 되며 그 다음부터는 1씩 증가하여 자동으로 데이터가 삽입된다. number 속성은 다른 테이블의 기본키와 연결되는 외래키로도 활용이 되기 때문에 매우 중요한 속성이다. password는 해시값으로 저장되기 때문에 크기를 255로 설정했다. indi는 True와 False 데이터를 저장하는 속성이기 때문에 tinyint 타입을 사용한다.

- 기업 회원 테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
<input type="checkbox"/> 1	company	varchar(10)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 2	crn	int(11)			아니오	없음		
<input type="checkbox"/> 3	number	int(11)			아니오	없음		AUTO_INCREMENT
<input type="checkbox"/> 4	seller	tinyint(4)			아니오	없음		

[그림 3-10] ent 테이블

기업회원이 가입 시 입력하는 추가 데이터다. crn은 사업자 등록번호이다. 해당 테이블도 사용자를 관리하는 테이블이기 때문에 number를 기본키로 사용하고 AUTO_INCREMENT를 적용했다. seller는 기업 판매자인지 기업 구매자인지를 식별하기 위한 속성이다. 회원 테이블에 있는 indi 속성과 마찬가지로 True, False 데이터만 넣기 위해 tinyint 타입을 사용했다.

- 장바구니 테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
<input type="checkbox"/> 1	number	int(11)			아니오	없음		AUTO_INCREMENT
<input type="checkbox"/> 2	productcode	int(11)			아니오	없음		
<input type="checkbox"/> 3	productname	varchar(250)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 4	quantity	int(11)			아니오	없음		
<input type="checkbox"/> 5	price	int(11)			아니오	없음		
<input type="checkbox"/> 6	seller	varchar(45)	utf8_general_ci		아니오	없음		

[그림 3-11] shopping_cart 테이블

사용자가 회원가입 할 때 생성되는 사용자 고유의 장바구니 테이블이다. number는 장바구니에 들어간 상품의 순서를 나타내는 속성이기 때문에 AUTO_INCREMENT를 적용했고 각 데이터를 식별하기 위해 기본키로 설정했다.

- 주문내역 테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
<input type="checkbox"/> 1	buyer	int(11)			예	NULL		
<input type="checkbox"/> 2	delivery_address	varchar(255)	utf8_general_ci		예	NULL		
<input type="checkbox"/> 3	delivery_tel	varchar(45)	utf8_general_ci		예	NULL		
<input type="checkbox"/> 4	order_no	int(11)			아니오	없음		AUTO_INCREMENT
<input type="checkbox"/> 5	orderer	int(11)			아니오	없음		
<input type="checkbox"/> 6	paid	tinyint(4)			아니오	없음		
<input type="checkbox"/> 7	price	int(11)			아니오	없음		
<input type="checkbox"/> 8	product	varchar(255)	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 9	purchase_signature	varchar(5000)	utf8_general_ci		예	NULL		
<input type="checkbox"/> 10	receipt	mediumtext	utf8_general_ci		예	NULL		
<input type="checkbox"/> 11	trade_time	varchar(15)	utf8_general_ci		예	NULL		

[그림 3-12] trade_detail 테이블

사용자들이 주문 시 데이터가 삽입되는 주문내역 테이블이다. order_no는 주문내역으로, 고유값이 필요하고 각 데이터를 식별해야하므로 AUTO_INCREMENT를 적용하고 기본키로 설정했다. paid는 결제가 되었는지 여부를 넣기 때문에 True나 False 값을 삽입하기 위해 tinyint 타입을 사용하였다. purchase_signature는 구매자가 구매할 시 생성되는 구매자의 전자서명 값이다. pem 형식으로 저장되며 길이가 매우 길기 때문에 5000바이트의 길이를 할당하였다. receipt는 판매자의 전자서명 값이다. 마찬가지로 pem 형식으로 저장하는 필드이다.

- 인증서 테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
<input type="checkbox"/> 1	certnumber 	int(11)			아니오	없음		AUTO_INCREMENT
<input type="checkbox"/> 2	masterCert	tinyint(1)			아니오	0		
<input type="checkbox"/> 3	allowed	tinyint(1)			아니오	0		
<input type="checkbox"/> 4	disable	tinyint(1)			아니오	1		
<input type="checkbox"/> 5	cert	text	utf8_general_ci		아니오	없음		
<input type="checkbox"/> 6	deviceId	varchar(45)	utf8_general_ci		예	NULL		

[그림 3-13] cert 테이블

사용자의 인증서가 저장되는 테이블이다. 인증서의 순서를 알기 위해 certnumber 속성은 AUTO_INCREMENT를 적용하였고 고유값이고 각 인증서를 식별하기 위해 기본키로 설정하였다. masterCert는 마스터 인증서인지 확인하기 위한 필드이고 True, False 값을 넣기 위해 tinyint 타입을 사용하였다. allowed와 disable은 활성화 된 인증서인지 비활성화된 인증서인지 확인하는 필드이다. 인증서를 발급하면 마스터 인증서로 활성화를 시켜줘야하기 때문에 데이터가 삽입되면 disable 필드가 1이 되어 인증서가 비활성화된다.

3.2 프로그램 구성

3.2.1 웹페이지 기능

COCONUT

모든 상품 QR결제 회원가입 로그인

QR코드 간편결제 시스템



QR코드를 이용한 모바일 웹 애플리케이션
인증서를 사용한 안전한 거래
JSON Web Token기반의 무상태 서비스 제공

Express Backend

Express는 웹 및 모바일 애플리케이션을 위한 일련의 강력한 기능을 제공하는 간결하고 유연한 Node.js 웹 애플리케이션 프레임워크입니다.

자유롭게 활용할 수 있는 수많은 HTTP 유틸리티 메소드 및 미들웨어를 통해 쉽고 빠르게 강력한 API를 작성할 수 있습니다.

Vue JS

Vue.js는 뷰(View)에 최적화된 프론트엔드 프레임워크입니다. 컨트롤러 대신 뷰 모델을 가지는 MVVM(Model-View-ViewModel) 패턴을 기반으로 디자인되었으며, 컴포넌트를 사용하여 재사용이 가능한 UI들을 묶고 뷰 레이어를 정리하는 것이 가장 강력한 기능입니다.

JWT Tokens

JSON Web Token은 정보를 안전하게 전송하기 위해 정의된 공개된 표준(RFC 7519)입니다. JWT은 자체적으로 필요한 모든 정보를 포함합니다. 헤더 정보와, 실제 전달할 데이터, 검증할 수 있는 서명 데이터를 모두 포함하고 있습니다.

디지털 서명에 의해 검증할 수 있으며 신뢰할 수 있습니다. 비밀 값을 사용하는 HMAC 알고리즘이나 RDS or ECDSA와 같은 공개키, 개인키 쌍으로 서명될 수 있습니다.

X.509 인증서

X.509는 PKI에서 사용하는 표준 인증서 형식이다. PKI에서 사용하는 공개키, 개인키 같은 비대칭키를 X.509 인증서로 관리한다.

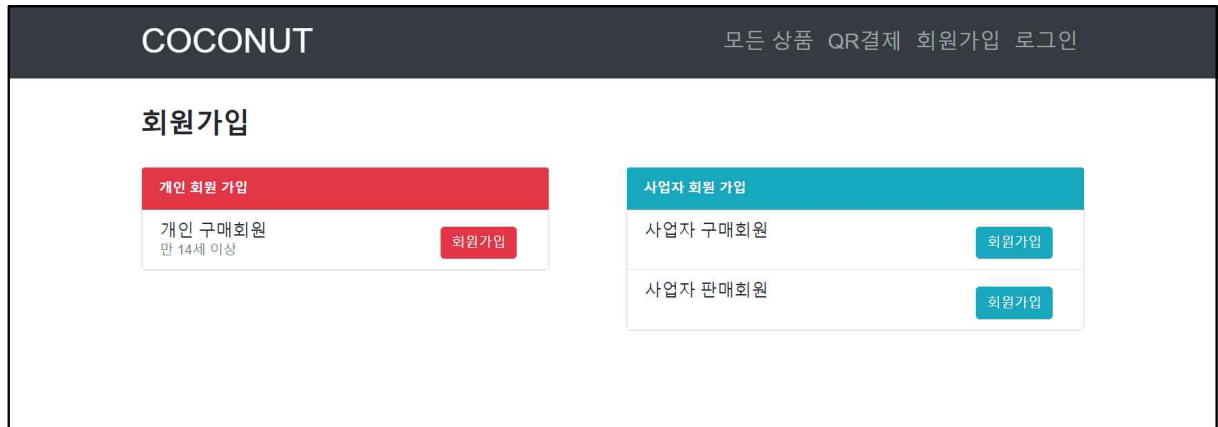
X.509(V1)은 1993년 디렉터리 접근 제어를 위한 두 가지 내용을 추가하기 위해 X.509(V2) 형식으로 개정되었다. 그리고 이메일의 보안 요소 등 새로운 개념이 포함된 X.509(V3)가 1996년에 발표되었다.

Vuex

Vuex는 Vue.js 애플리케이션에 대한 상태 관리 패턴 + 라이브러리입니다. 애플리케이션의 모든 컴포넌트에 대한 중앙 집중식 저장소 역할을 하며 예측 가능한 방식으로 상태를 변경할 수 있습니다. 또한 Vue의 공식 확장 프로그램과 통합되어 설정 시간이 필요 없는 디버깅 및 상태 스냅 샷과 같은 고급 기능을 제공합니다.

[그림 3-14] 홈페이지의 메인 페이지

웹페이지 메인에는 해당 웹페이지에 사용한 기술들을 나열하고 해당 웹페이지 주소를 QR코드화 하여 스마트폰의 카메라로 찍으면 접속할 수 있도록 나타내고 있다. 페이지의 상단에는 Navigation Bar를 넣어 탐색을 더 쉽게 만들었다.



COCONUT

모든 상품 QR결제 회원가입 로그인

회원가입

개인 회원 가입

개인 구매회원
만 14세 이상

회원가입

사업자 회원 가입

사업자 구매회원

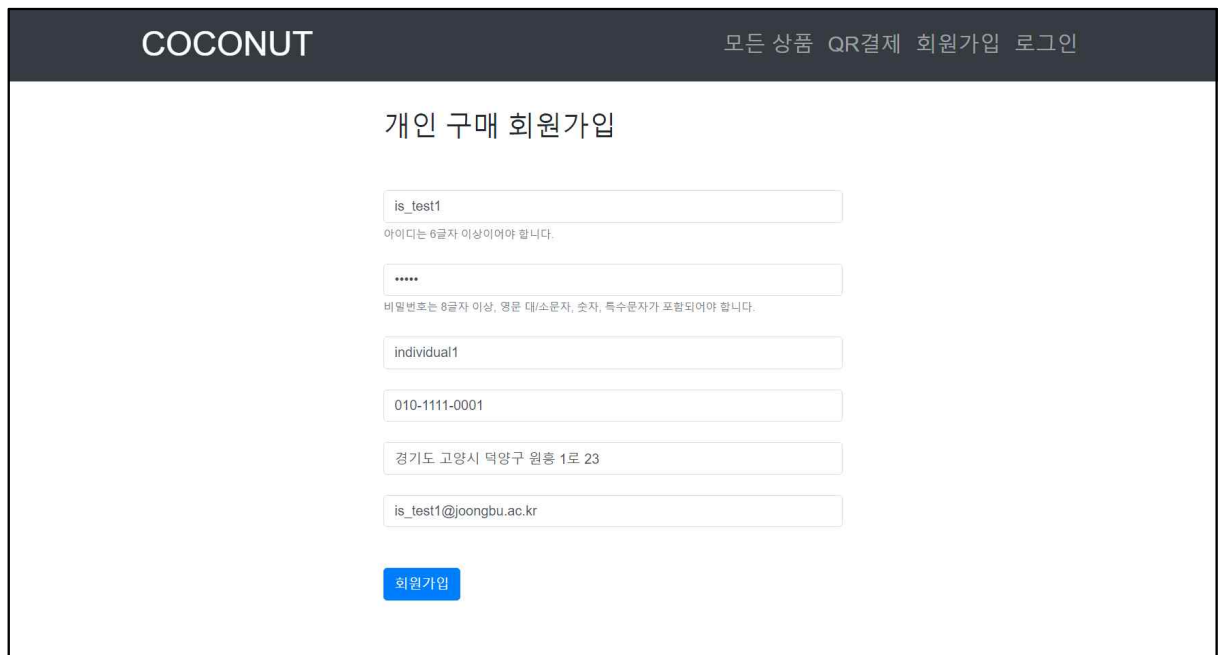
회원가입

사업자 판매회원

회원가입

[그림 3-15] 회원가입 페이지

회원가입은 개인과 사업자를 구분하여 가입하고 사업자는 구매회원과 판매회원으로 나누어 가입을 할 수 있다.



COCONUT

모든 상품 QR결제 회원가입 로그인

개인 구매 회원가입

is_test1

아이디는 6글자 이상이어야 합니다.

비밀번호는 8글자 이상, 영문 대/소문자, 숫자, 특수문자가 포함되어야 합니다.

individual1

010-1111-0001

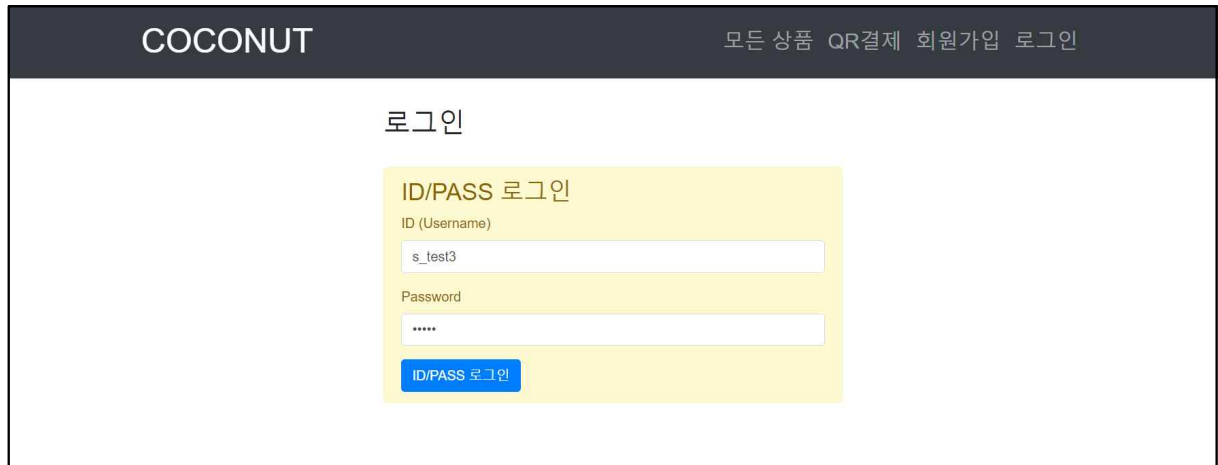
경기도 고양시 덕양구 원흥 1로 23

is_test1@joongbu.ac.kr

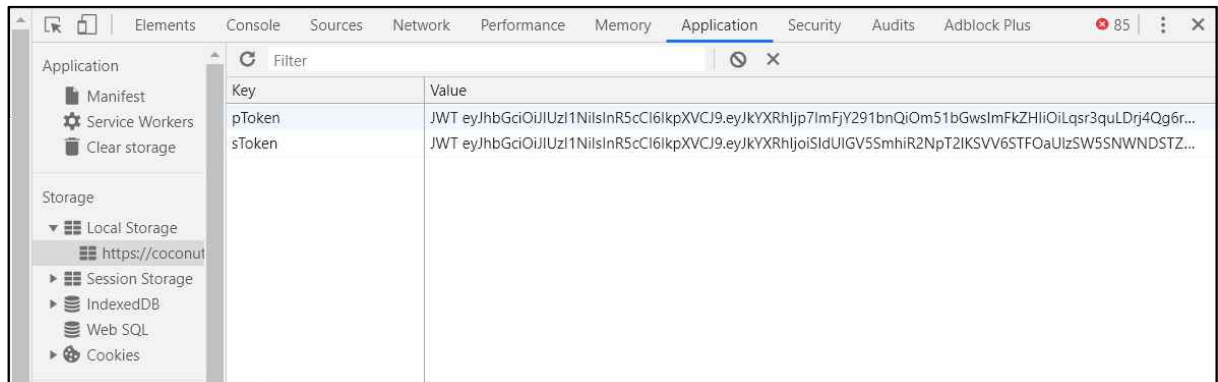
회원가입

[그림 3-16] 개인 구매 회원가입 페이지

개인 구매 회원가입 시 개인 구매 회원은 아이디, 비밀번호, 이름, 전화번호, 주소, 이메일 등을 입력받고 회원가입을 진행하게 된다. 사업자는 사업자 등록번호와 상표명 추가로 들어가게 된다. 회원가입 시 데이터베이스에 장바구니와 인증서 테이블을 생성하게 된다.



[그림 3-17] 로그인 페이지



[그림 3-18] Json Web Token 값

회원가입 시 입력한 ID와 Password를 이용해 로그인을 하게 된다. 로그인 시 서버에서 현재 시간 값과 의사 난수 값을 이용하여 토큰 값을 생성해 클라이언트로 전송한다. 클라이언트는 서버로부터 받은 토큰 값을 브라우저 로컬 스토리지에 저장하여 인증을 유지한다. 이 인증은 약 일주일 간 유효하다.

QR코드 간편결제 시스템



QR코드를 이용한 모바일 웹 애플리케이션
인증서를 사용한 안전한 거래
JSON Web Token기반의 무상태 서비스 제공

Express Backend

Express는 웹 및 모바일 애플리케이션을 위한 일련의 강력한 기능을 제공하는 간결하고 유연한 Node.js 웹 애플리케이션 프레임워크입니다.

자유롭게 활용할 수 있는 수많은 HTTP 유틸리티 메소드 및 미들웨어를 통해 쉽고 빠르게 강력한 API를 작성할 수 있습니다.

Vue JS

Vue.js는 뷰(View)에 최적화된 프론트엔드 프레임워크입니다. 컨트롤러 대신 뷰 모델을 가지는 MVVM(Model-View-ViewModel) 패턴을 기반으로 디자인되었으며, 컴포넌트를 사용하여 재사용이 가능한 UI들을 묶고 뷰 레이어를 정리하는 것이 가장 강력한 기능입니다.

JWT Tokens

JSON Web Token은 정보를 안전하게 전송하기 위해 정의된 공개된 표준(RFC 7519)입니다. JWT은 자체적으로 필요한 모든 정보를 포함합니다. 헤더 정보와, 실제 전달할 데이터, 검증할 수 있는 서명 데이터를 모두 포함하고 있습니다.

디지털 서명에 의해 검증할 수 있으며 신뢰할 수 있습니다. 비밀 값을 사용하는 HMAC 알고리즘이나 RDS or ECDSA와 같은 공개키, 개인키 쌍으로 서명될 수 있습니다.

X.509 인증서

X.509는 PKI에서 사용하는 표준 인증서 형식입니다. PKI에서 사용하는 공개키, 개인키 같은 비대칭 키를 X.509 인증서로 관리한다.

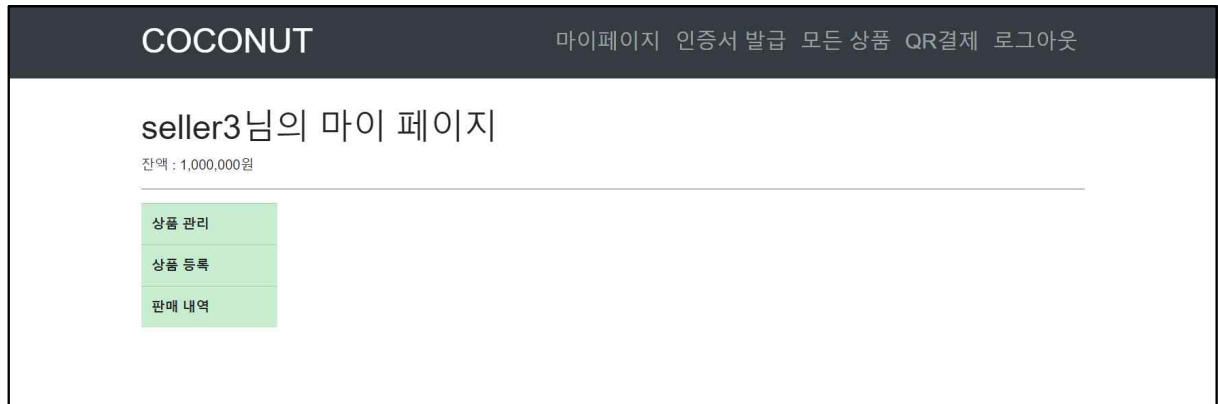
X.509(V1)은 1993년 디렉터리 접근 제어를 위한 두 가지 내용을 추가하기 위해 X.509(V2) 형식으로 개정되었다. 그리고 e메일의 보안 요소 등 새로운 개념이 포함된 X.509(V3)가 1996년에 발표되었다.

Vuex

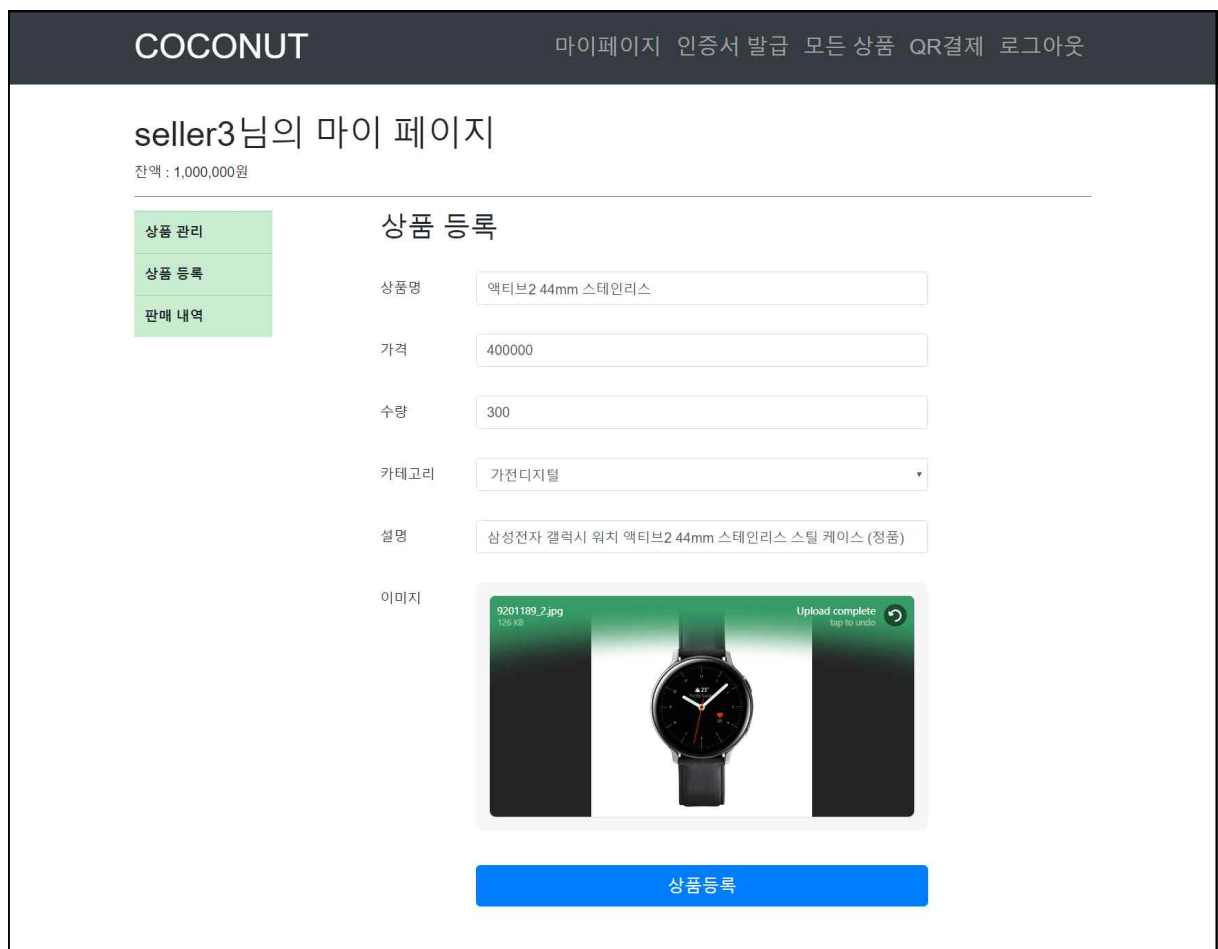
Vuex는 Vue.js 애플리케이션에 대한 상태 관리 패턴 + 라이브러리입니다. 애플리케이션의 모든 컴포넌트에 대한 중앙 집중식 저장소 역할을 하며 예측 가능한 방식으로 상태를 변경할 수 있습니다. 또한 Vue의 공식 확장 프로그램과 통합되어 설정 시간이 필요 없는 디버깅 및 상태 스냅 샷과 같은 고급 기능을 제공합니다.

[그림 3-19] 로그인 후의 메인 페이지

로그인을 하면 Navigation Bar에 로그인과 회원가입 버튼이 사라지고 로그아웃 버튼을 보여준다. 해당 기능은 Vue.js의 기능 중 하나인 조건부 렌더링을 이용해 구현하였다.



[그림 3-20] 기업판매 회원의 마이 페이지



[그림 3-21] 상품 등록

seller3님의 마이 페이지

잔액 : 1,000,000원

상품 관리

상품 등록

판매 내역

등록 상품 관리

액티브2 44mm 스테인리스

수량 : 300

가전디지털

삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품)

상품번호 : 10

400000원

갤럭시 워치 46mm

수량 : 200

가전디지털

삼성전자 갤럭시 워치 46mm (정품)

상품번호 : 11

290000원

어메이즈핏 뱀

수량 : 300

가전디지털

샤오미 화미 어메이즈핏 뱀 (해외구매)

상품번호 : 12

60000원

APPLE 워치 시리즈5

수량 : 100

가전디지털

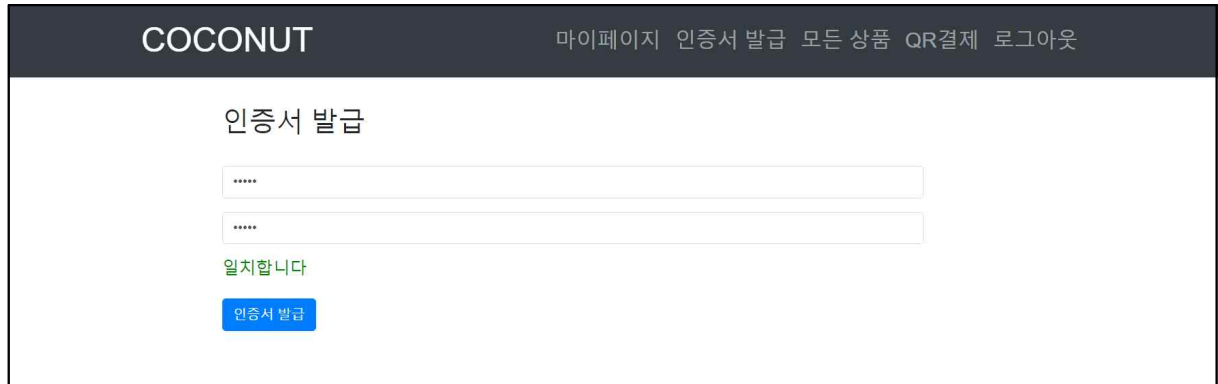
APPLE 워치 시리즈5 40mm 스페이스 그레이 알루미늄 케이스, 해외구매 (스포츠 밴드)

상품번호 : 13

540000원

[그림 3-22] 'seller3' 계정으로 등록한 상품

3.2.2 인증서 발급 기능



COCONUT 마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

인증서 발급

일치합니다

인증서 발급

[그림 3-23] 인증서 발급 페이지

해당 웹페이지에서 인증서를 사용하는 이유는 거래 시 전자 서명을 하기 위해서이다. 흔히 공개키 쌍을 사용하는데, 개인키는 전자 서명을 할 때 사용하고, 공개키는 해당 전자 서명을 검증할 때 사용한다. 공개키는 인증서에 같이 포함되어 있으므로 개인키를 안전하게 보관해야 하는데, 해당 웹페이지에서는 사용자의 비밀번호를 대칭키로 사용하여 암호화 하게 된다.



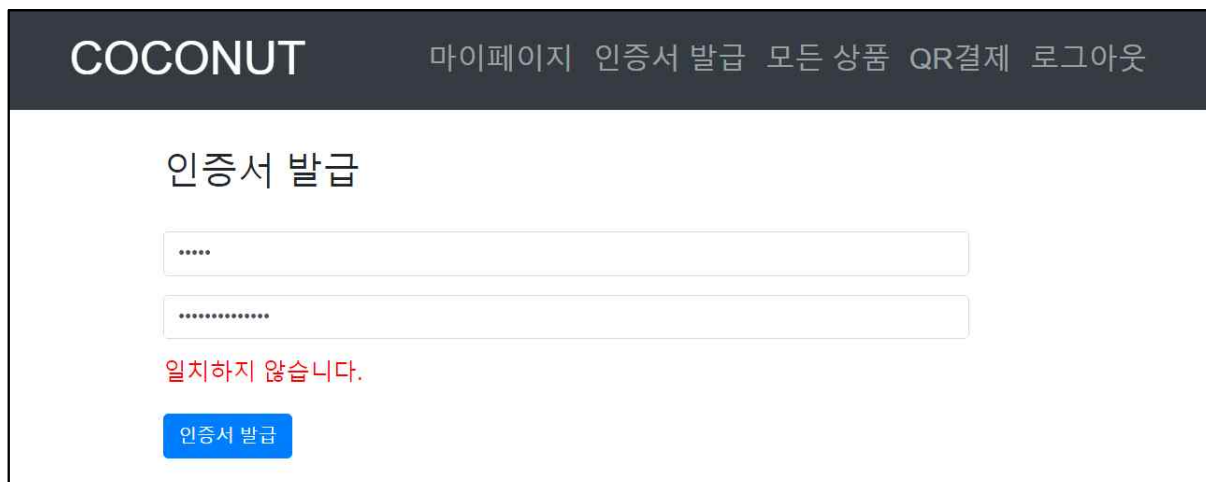
COCONUT 마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

인증서 발급

일치합니다

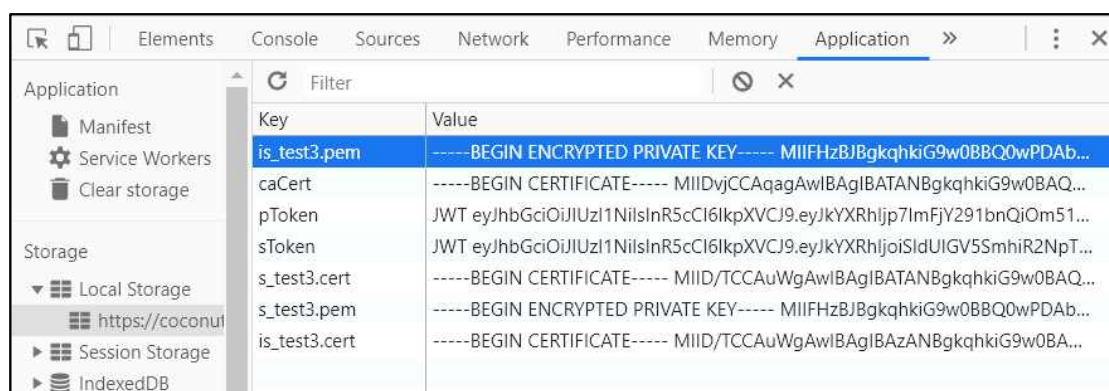
인증서 발급

[그림 3-24] 인증서 암호 비밀번호가 일치 시



[그림 3-25] 인증서 암호 비밀번호가 불일치 시

해당 웹페이지에서는 Vue.js의 장점 중 하나인 양방향 데이터 바인딩을 이용하여 비밀번호 일치, 불일치 여부를 판별한다. 양방향 데이터 바인딩은 AngularJS에 있던 기능이지만 AngularJS 개발자였던 Evan You가 AngularJS의 장점을 뽑아내 만든 것이 Vue.js 이기 때문에 가능한 것이다.



[그림 3-26] 개인키 및 인증서가 브라우저 로컬스토리지에 저장된 모습

인증서와 개인키를 발급받으면 브라우저의 로컬 스토리지에 저장되고 개인키는 사용자가 입력한 비밀번호에 의해 암호화되어있다.

3.2.3 QR Code 결제 기능

COCONUT

마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

모든 상품

의류

식품

생활용품

가전/디지털

스포츠/레저

자동차 용품

도서/음반/DVD

완구/취미


문구/오피스

반려동물용품

뷰티

출산/유아동


주방용품



액티브2 44mm 스테인리스

가전디지털


400,000원



갤럭시 워치 46mm

가전디지털


290,000원



어메이즈핏 빕

가전디지털


60,000원



APPLE 워치 시리즈5

가전디지털


540,000원



다용도 스위즈

생활용품


7,000원



깨끗한나라 30m

생활용품


11,000원



스타벅스 머그컵

생활용품


10,000원



다우니 초고농축

생활용품

3,000원



비모 휴대폰 케이스

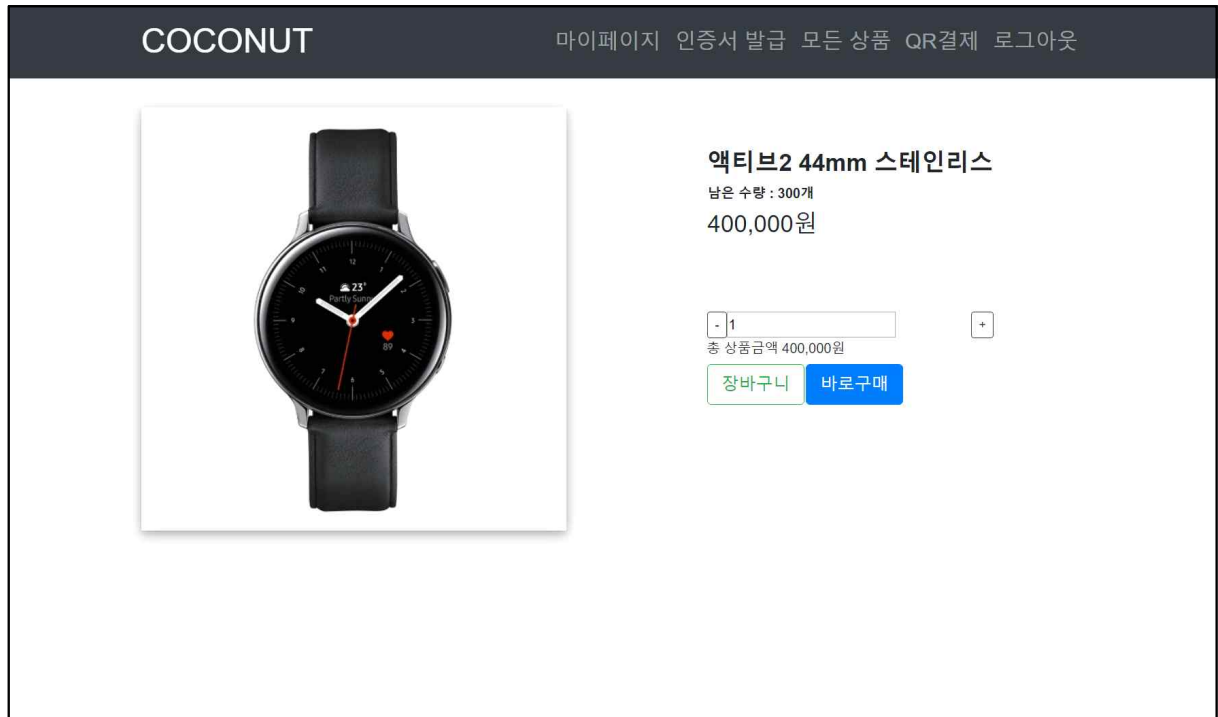
가전디지털

7,600원

[그림 3-27] 모든 상품 페이지

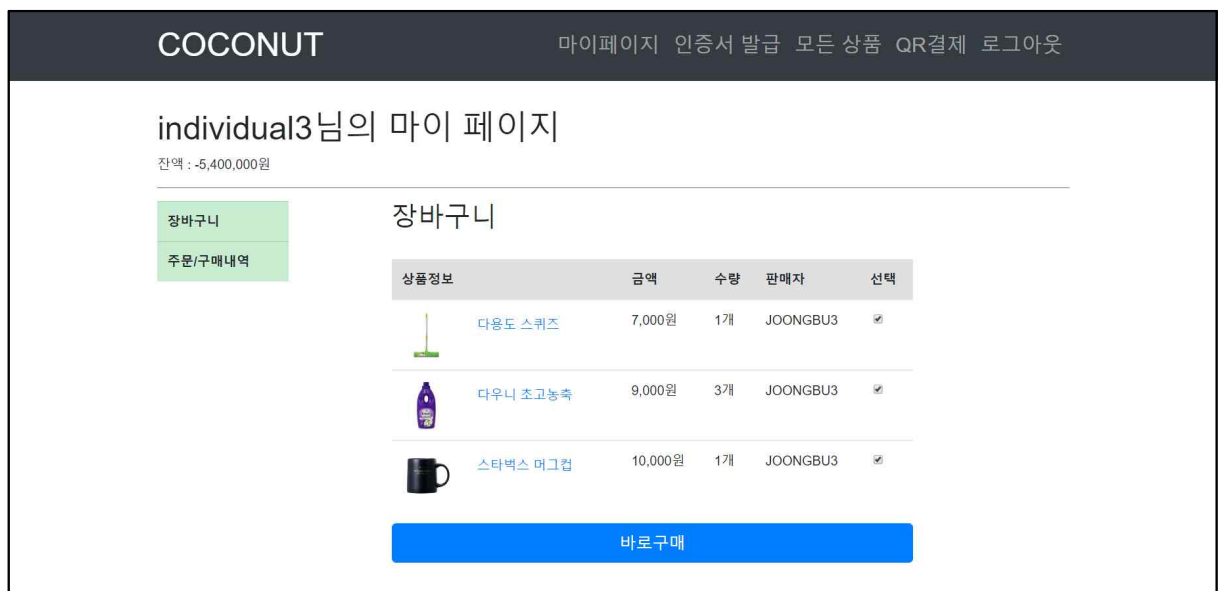
모든 상품 페이지로 들어가면 데이터베이스에 있는 모든 상품과 상품 정보를 보여준다. 해당 페이지는 자식 컴포넌트를 등록하여 상품을 보여준다. 즉, 해당 페이지의 모든 요소는 화면 좌측의 카테고리이고, 화면 중앙의 상품들은 자식 컴포넌트가 모든 상품 페이지 위에서 보여지는 것이다.

- 25 -



[그림 3-28] 상품 페이지

상품 페이지에서 사용자는 수량을 정하고 장바구니에 넣을 지, 바로 구매할지 선택한다. 장바구니에 넣을 시 회원가입 할 때 만들었던 장바구니 테이블에 값이 들어가므로 장바구니가 영구적으로 유지가 된다.




[그림 3-29] 장바구니 페이지

장바구니 페이지에서는 자신이 장바구니에 넣은 모든 상품들의 정보 및 수량을 볼 수 있고 원하는 상품만 선택하여 주문할 수 있다.

COCONUT
마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

결제

상품정보	금액	수량	판매자
 액티브2 44mm 스테인리스 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품) 가전디지털	800,000 원	2 개	JOONGBU3

총 주문 상품수

1종 2개

총 결제 예상 금액

800,000원

바로 결제

QR코드 결제

[그림 3-30] 상품 바로 결제

상품 정보 페이지에서 바로 결제를 클릭하면 배송지와 연락처를 입력하는 페이지를 출력하게 되고 배송지와 연락처를 입력하게 되면 결제를 할 수 있는 페이지를 출력하게 된다.

COCONUT
마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

individual3님의 주문서

1. 주문상품 확인

상품정보	금액	수량	판매자
다용도 스위치	7,000원	1개	JOONGBU3
다우니 초고농축	9,000원	3개	JOONGBU3
스타벅스 머그컵	10,000원	1개	JOONGBU3

2. 배송지 정보 입력

고객입력 상세주소	경기도 고양시 덕양구 원흥 1로 23
우편번호	10562
연락처	010-3333-3003

주문하기

[그림 3-31] 장바구니에서 결제

장바구니에서 원하는 상품을 선택해 결제하게 되면 해당 상품들만 주문서에 작성되고 배송지 정보를 입력하는 페이지를 출력한다.

결제

상품정보	금액	수량	판매자
 액티브2 44mm 스테인리스 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스텝 케이스 (정품) 가전디지털	800,000 원	2 개	JOONGBU3

총 주문 상품수 1종 2개

총 결제 예상 금액 800,000원

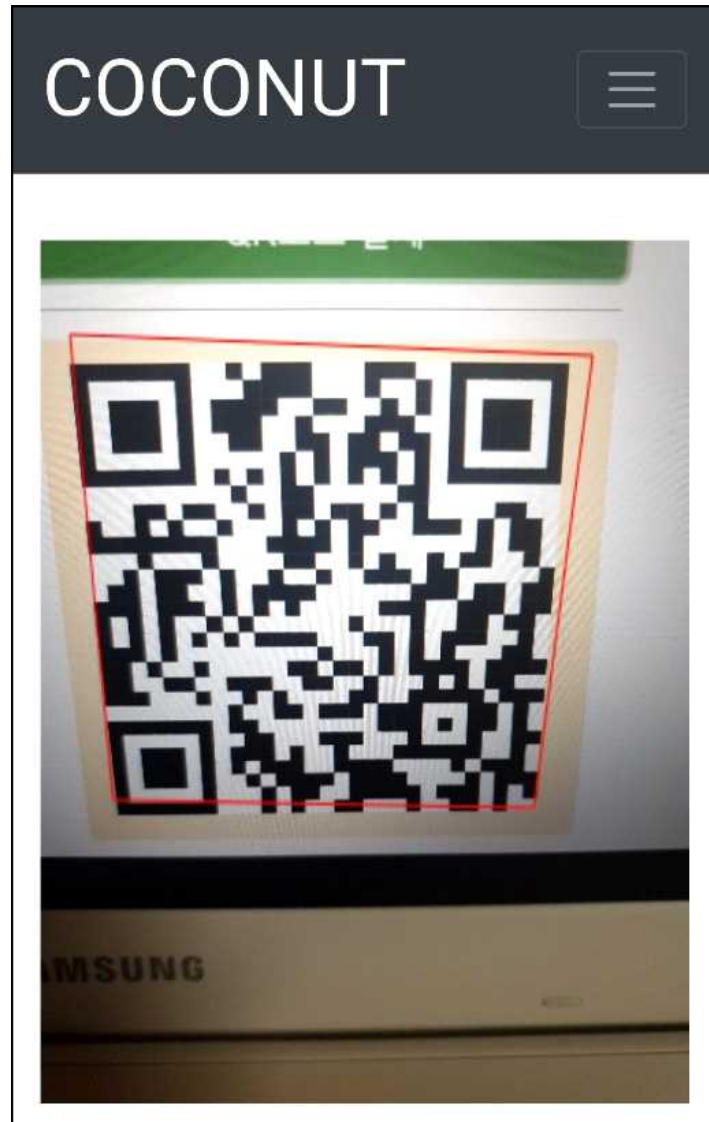
바로 결제

QR코드 결제



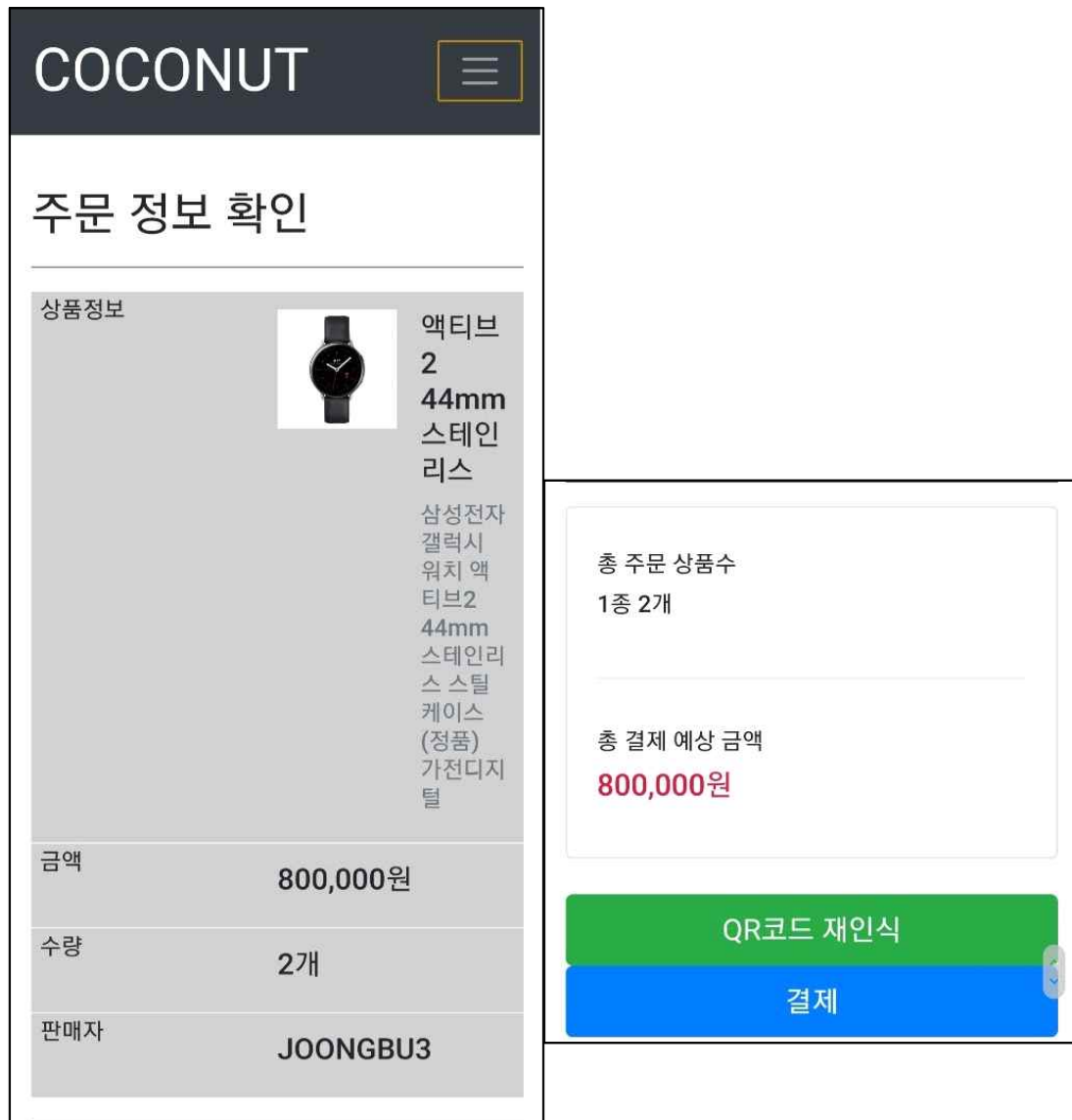
[그림3-32] 결제 페이지

결제페이지에서는 인증서를 이용해 바로 결제를 할지, 모바일 기기를 이용해 QR코드를 할지 결정한다. 바로 결제를 선택한다면 개인키 비밀번호를 입력하는 창이 뜨고 QR코드로 결제를 선택한다면 결제에 사용될 QR코드가 화면에 출력된다.



[그림 3-33] 모바일 페이지에서 QR Code 인식

모바일 기기에서 QR코드를 인식하면 [그림 3-21]처럼 인식되는 QR코드 외곽에 붉은 사각형을 출력하면서 인식을 하여 서버에 QR코드 정보와 자신의 인증정보를 전송하게 되어 서버에서는 결제가 처리된다.



[그림 3-34] 모바일 페이지에서 QR코드 인식 후 최종 결제 페이지

모바일 기기에서 QR코드를 정상적으로 인식하고 서버에 정보를 보내게 되면 서버에서는 정상적인 결제요청인지 확인 후 모바일 기기로 응답 값을 보내 모바일 기기는 서버로부터 받은 응답 값을 사용자에게 보여준다. 그 후 사용자는 자신의 주문 정보를 한번 더 확인하여 QR코드를 재인식시킬지, 결제를 진행할지 선택하게 된다. 결제 선택 시 인증서 비밀번호 필드를 활성화시켜 비밀번호를 입력받게 된다.

The image shows a mobile application interface for 'COCONUT'. At the top, the brand name 'COCONUT' is displayed in white on a dark grey background, next to a hamburger menu icon. Below this, a white box contains the order summary: '총 주문 상품수' (Total ordered items) as '1종 2개' (1 type, 2 items). A horizontal line separates this from the total payment amount, '총 결제 예상 금액' (Total payment estimated amount), which is '800,000원' (800,000 KRW) in red text. Below the summary box are two large, rounded rectangular buttons: a green one labeled 'QR코드 재인식' (QR code re-identification) and a blue one labeled '결제' (Payment). A horizontal line follows. Below the line is a yellow box titled '인증서 비밀번호 입력' (Certificate password input). Inside this box, the word 'Password' is shown above a white input field containing five dots. At the bottom of the yellow box is a blue button labeled '결제' (Payment).

[그림 3-35] 결제를 위한 인증서 입력

결제를 할 때 부인방지를 위해 서명을 할 필요가 있다. 서명을 하기 위해서는 개인키가 필요하지만 본 사이트에서는 개인키가 사용자의 비밀번호로 암호화되어있기 때문에 개인키를 평문으로 풀 비밀번호를 사용자로부터 입력받아 개인키를 복호화 하고 서명을 진행하여 서명값과 결제요청을 서버로 전송한다.

COCONUT

결제 완료

결제 금액

결제일

800,000원
2019-10-18 / 23:45:07

구매자

주문자

individual3
individual3

배송 주소

주문자 전화번호

경기도 고양시 덕양구 원흥 1로 23
010-3333-3303

주문번호

78

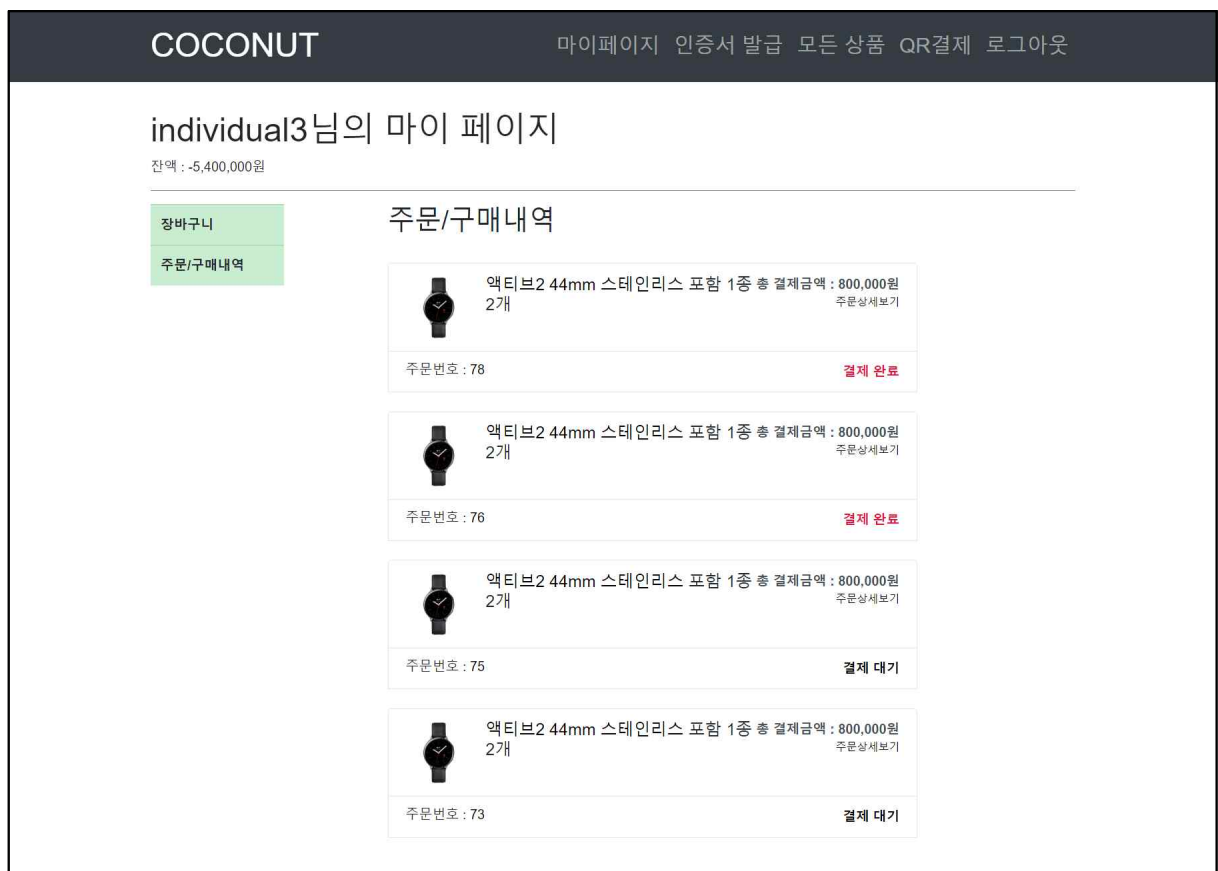
[그림 3-36] 결제가 완료된 후의 확인 페이지

결제가 완료되면 결제금액, 시간정보, 주소, 전화번호, 주문번호 등의 정보를 출력한다.



[그림 3-37] 결제가 완료되면 바뀌는 PC페이지

모바일 기기에서 결제를 완료하면 QR코드를 출력하던 PC에서는 결제완료가 되었다는 페이지로 리다이렉션하여 모바일과 동일한 정보를 출력하는 결제 결과를 출력한다.




[그림 3-38] 구매자의 주문/구매내역

결제 중 여러 가지의 오류로 인해 결제가 정상적으로 진행이 안될 시 결제가 되지 않았으므로 결제 대기 표시가 출력되고 정상적으로 결제가 완료된 주문은 결제 완료가 출력된다.

3.2.4 전자서명 기능

COCONUT
마이페이지 인증서 발급 모든 상품 QR결제 로그아웃

주문번호 : 78

상품정보	금액	수량	판매자	영수증
 액티브2 44mm 스테인리스 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품) 가전디지털	800,000 원	2 개	JOONGBU3	미발급

주문자	12	구매자	12
배송 주소	경기도 고양시 덕양구 원흥 1로 23	전화번호	010-3333-3303
결제 시간	2019-10-18 / 23:45:07		

총 주문 상품수	1종 2개
총 결제금액	800,000원

[그림 3-39] 구매자 주문 상세보기 페이지

구매자 계정으로 주문 상세보기 페이지로 들어가면 해당 주문 정보와 영수증을 출력한다. [그림 3-27]에서는 결제가 완료되었지만 영수증이 '미발급' 상태로 출력이 되는데, 결제가 완료되었더라도 판매자가 해당 결제정보를 확인하지 않았으므로 거래가 성사된 것이 아니다. 거래가 아직 성사되지 않았으므로 영수증도 출력이 되지 않고 영수증은 미발급 상태로 출력된다. 판매자가 해당 주문을 확인하고 영수증 발급을 하면 구매자의 해당 페이지에서는 영수증 필드에 '발급' 문자열이 출력된다.

seller3님의 마이 페이지

잔액 : 4,213,000원

상품 관리

상품 등록

판매 내역

판매내역



액티브2 44mm 스테인리스 2개

삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품)
가전디지털

상품 금액 : 800,000원

주문상세보기

주문번호 : 76

결제 완료



액티브2 44mm 스테인리스 2개

삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품)
가전디지털

상품 금액 : 800,000원

주문상세보기

주문번호 : 75

결제 대기



다우니 초고농축 2개

P&G 다우니 초고농축 화이트티와 릴리 향 1L (1개)
생활용품

상품 금액 : 6,000원

주문상세보기



다용도 스퀴즈 1개

3M 스카치브라이트 다용도 스퀴즈 (1개)
생활용품

상품 금액 : 7,000원

주문상세보기

주문번호 : 74

결제 완료



액티브2 44mm 스테인리스 2개

삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품)
가전디지털

상품 금액 : 800,000원

주문상세보기

주문번호 : 73

결제 대기

[그림 3-40] 판매자의 판매 내역 페이지

판매자 계정으로 판매내역 페이지로 들어가면 자신의 상품을 주문한 주문이 출력되고 해당 주문에 대한 간략한 주문정보들이 출력된다. 앞에서 판매자는 구매자의 주문을 확인하고 영수증을 발급해야 거래가 성사되었다고 했는데, 영수증 발급은 주문 상세보기 페이지로 들어가면 볼 수 있다.

주문번호 : 78

판매 상품정보	금액	수량	판매자	영수증
 액티브2 44mm 스테인리스 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품) 가전디지털	800,000 원	2 개	JOONGBU3	미 발 급

주문자	12	구매자	12
배송 주소	경기도 고양시 덕양구 원흥 1로 23	전화번호	010-3333-3303
결제 시간	2019-10-18 / 23:45:07		

주문 상품수	1종 2개
결제 금액	800,000원

영수증 발급

인증서 비밀번호 입력


Password

영수증 발급

[그림 3-41] 영수증 발급을 위한 인증서 입력

판매자는 해당 거래를 확인하고 거래를 승인한다. 이 때, 거래에 대한 부인방지를 위해 판매자도 서명을 해야한다. 판매자는 서명을 위해 결제가 완료된 주문의 상세정보 페이지에 들어가 영수증 발급 버튼을 클릭한다. 구매자의 서명은 결제를 할 때 이미 받았으므로 판매자는 거래승인을 위해 구매자처럼 암호화된 개인키를 자신만의 비밀번호로 복호화하여 서명한다. 판매자는 자신의 비밀번호를 입력하여 하단의 영수증 발급 버튼을 클릭하면 암호화된 개인키를 복호화하여 개인키로 서명한 서명값과 영수증 발급요청을 서버로 보내 서버에서는 영수증 발급을 처리한다.

주문번호 : 78

판매 상품정보	금액	수량	판매자	영수증
 액티브2 44mm 스테인리스 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품) 가전디지털	800,000 원	2 개	JOONGBU3	발급완료

주문자	12	구매자	12
배송 주소	경기도 고양시 덕양구 원흥 1로 23	전화번호	010-3333-3303
결제 시간	2019-10-18 / 23:45:07		

주문 상품수	1종 2개
결제 금액	800,000 원

영수증 확인

영수증

상 호 : JOONGBU3
 사업자 번호 : 2147483647
 주 소 : 경기도 고양시 덕양구 원흥 3로 23
 대표자 : seller3
 전화번호 : 010-3333-00
 매출일 : 2019-10-18 / 23:45:07
 번 호 : 78

상품명	단가	수량	금액
액티브2 44mm 스테인리스	400,000	2	800,000

영수증 서명값	2add54870a0c1469689bd18667a35870955bc ae57dbb4dba574bf7d7036c9bf7d5c5ca68557 15daad4f90bfbff620bdc245059c14685edbb0 6ad921cce0cd6a7695cad1ee62447d33606dc 699dd515469af8cb2bac9074220f4460d25d83 4aa5e48ff5b55a6461bf0305ff7c5e7bcaacca4 5ca1a120bd8b6d2e45e9499d5636dd57143b9 59138963436ce1f4b0bd1d2b81d6a15755446 07bd983965e91fc68186fc1ae81cec665b6fcfc 39c9f766c2db62645d3e9e22b57e9f9620d872 d135da92f8730be5023076137368ca3a729f11 5b7fae967a6d2c3c5e6c27ee424e93eeca91 bedd8a40f60cabdcb9fc6e16ac67f88242086 d4fc3cbac8d9044f997	영수증 서명 확인
---------	--	-----------

[그림 3-42] 영수증 발급과 서명값 확인

판매자가 영수증 발급을 하면 구매자는 자신의 주문 상세페이지에서 판매자가 발급한 영수증을 확인할 수 있다. 본 페이지에서 구매자는 판매자의 서명을 검증할 수 있다. 전자서명의 검증은 서명자의 공개키를 사용하면 검증할 수 있으므로 구매자는 판매자의 인증서에 포함된 공개키를 사용하여 판매자의 서명을 검증할 수 있으며 영수증 서명 확인 버튼을 누르면 서명을 검증하여 검증된 값을 브라우저에 출력한다.

4. 결론 및 향후 과제

4.1 결론

현존하는 결제체계는 여러 보안 및 결제 플러그인을 설치해야하고 그 중 일부는 설치 후 결제가 진행 중이던 브라우저를 재시작하는 번거로움이 있다. 해당 프로젝트에서 QR 코드를 카메라로 찍어 결제하는 시스템을 통해 결제 과정을 간소화 하여 그에 따른 보안 취약점을 암호 기술 표준인 X.509 인증서를 활용하여 보완하였다. 이는 DNS Poisoning에 대한 일종의 대안이 될 수 있을 것이고 더 이상 플러그인을 설치하지 않아 사용자에게는 쾌적한 PC환경을 제공할 수 있을 것이다.

4.2 기대효과

본 과제에서 처음 사용한 기술은 Node.js와 Vue.js이다. 두 기술 모두 현업에서 활발하게 사용중인 기술들이다. 특히 Vue.js는 유연하고, 성능이 우수하고, 유지보수와 테스트가 편리한 자바스크립트 프레임워크라는 특징으로 인해 현재 가장 많이 쓰이는 프론트엔드 프레임워크 중 하나가 되었다. 이러한 기술에 대한 숙련도를 익힘으로써 현업에서 바로 업무에 투입할 수 있는 것을 기대할 수 있다.

4.3 향후 과제

현재 결제시스템은 아직 정식버전이 아니고 데모버전이고 은행과 제휴도 맺지 않았기 때문에 은행 계좌에 연결해서 금액을 충전하는 기능이 없다. 그러므로 차후에 창업을 하게 된다면 국내 여러 은행과 제휴를 맺어 금액을 충전하여 실제로 사용할 수 있게 만드는 것이 아직 과제로 남아있다.

5. 별첨

5.1 참고자료

- (1) Vue.js 가이드 한국어 페이지 (<https://kr.vuejs.org/v2/guide/index.html>)
- (2) 더밸류뉴스 (<http://www.thevaluenews.co.kr/news/view.php?idx=156172>)
- (3) 이병천, 「OAuth 2.0 MAC 토큰인증의 효율성 개선을 위한 무상태 난수화 토큰인증」, 2018.

5.2 소스코드

GitHub 링크 : <https://github.com/lunijay2/coconut>


```

<app.js>
const express = require('express');
const path = require('path');
const bodyparser = require('body-parser');
const cors = require('cors');
const passport = require('passport');
const mysql = require('mysql');
const config = require('./config/database');
const users = require('./routes/users');
const forge = require('node-forge');
const fs = require('fs');
const jwt = require('jsonwebtoken');
const pay = require('./routes/Pay');
const stores = require('./routes/stores');
const cert = require('./routes/Cert');

const https = require('https');
const http = require('http');

const app = express();

// port number

// localhost port
//const port = 3000;

// cloud service port
const port = process.env.PORT || 6000;

// CORS Middleware
app.use(cors());

// Body Parser Middleware
app.use(bodyparser.json());

app.use(passport.initialize());
app.use(passport.session());

require('./config/passport')(passport);

//app.use('/user', passport.authenticate('jwt', {session: false}), users);
app.use('/users', users);
app.use('/stores', stores);
app.use('/Pay', pay);
app.use('/Cert', cert);

app.use(express.static(path.join(__dirname, 'public')));

app.get('/*', function(req, res) {
    res.sendFile(path.join(__dirname, 'public/index.html'), function(err) {
        if (err) {
            res.status(500).send(err)
        }
    })
});

// Start server
app.listen(port, function(){

```

```
        console.log("server started on port "+port);
    });

    /*
    https.createServer(app).listen(port, function(){
        console.log('Https server started on port '+port);
    });*/
```

```

<users.js>
const express = require('express');
const router = express.Router();
const passport = require('passport');
const jwt = require('jsonwebtoken');
const mysql = require('mysql');
const config = require('../config/database');
const bcrypt = require('bcryptjs');
const multer = require('multer');
const forge = require('node-forge');
const fs = require('fs');
const path = require('path');

//const upload = multer({dest : './public/images'});
const upload = multer({
  storage: multer.diskStorage({
    destination: function (req, file, cb) {
      cb(null, './public/img/');
    },
    filename: function (req, file, cb) {
      cb(null, new Date().valueOf() + '.jpg');
      //cb(null, new Date().valueOf() + file.originalname);
    }
  }),
});

router.post('/imgupload', upload.single('bin'), (req, res, next) => {
  console.log('req.body : ' + JSON.stringify(req.body));
  console.log('req.file : ' + JSON.stringify(req.file));
  res.json(req.file.filename);
});

router.delete('/imgupload', (req, res, next) => {
  console.log('IMAGE DELETE');
  res.status(204).send()
});

// Register
router.post('/register', (req, res, next) => {
  let newUser = {
    name: req.body.name,
    id: req.body.id,
    password: req.body.password,
    tel: req.body.tel,
    addr: req.body.addr,
    email: req.body.email,
    indi: req.body.indi
  };

  CreateSalt() // Salt값 생성 함수 호출
    .then( function (resSalt) { // CreateSalt 함수가 resSalt를 반환한 것을 받음
      return PasswordHash(resSalt, newUser.password); // PasswordHash 함수호출
    })
    .then(function (resulthash) { //PasswordHash함수가 resulthash값을 반환
      var newUserHash = { //newUserHash에 유저정보, 해쉬화한 비밀번호를 json
        형태로 담는다
          newUser: newUser,
          hash : resulthash
        };
      return CreateRegisterQuery(newUserHash); // CreateQuery함수에

```

```

newUserHash를 보내며 호출
    })
    .then( function(query) { // CreateQuery함수에서 쿼리문을 반환
        return PoolGetConnection(query); // PoolGetConnection에 쿼리문을 보냄
        // 커넥션을 얻는데 쿼리문은 필요가 없지
만 뒤에 사용될 함수가 커넥션을 사용하므로 다음 함수에 쿼리문을 전달하기 위해서 쿼리문을
보냄
    })
    .then(function (connectionQuery) { // PoolGetConnection에서 커넥션과 쿼리문을 받
음
        return ExecuteQuery(connectionQuery); // ExecuteQuery함수에 커넥션과 쿼리
문을 보냄
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("가입 결과 : " + JSON.stringify(rows));
        return CreateShoppingCart(newUser.id);
    })
    .then( function (cartQuery) {
        return PoolGetConnection(cartQuery);
    })
    .then(function (connectionQuery) {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("장바구니 생성 결과 : " + JSON.stringify(rows));
        return CreateCertTable(newUser.id);
    })
    .then( function (certQuery) {
        return PoolGetConnection(certQuery);
    })
    .then(function (connectionQuery) {
        return ExecuteQuery(connectionQuery);
    })
    .then(function (rows) {
        console.log("인증서 테이블 생성 결과 : " + JSON.stringify(rows));
        return RegComplete(res); // RegComplete에 res를 보냄. res.json을 실행하기
위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Excute Query err : "+err);
        res.json({success: false, msg: 'Failed to register user 1'});
    })
    .catch( function (err) { // 전체적으로 에러를 캐치한다
        console.log("Catch err : "+err);
        res.json({success: false, msg: 'Failed to register user 2'}); // 에러 캐치시 false반
환
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    });
});

// RegisterEnt
router.post('/registerEnt', (req, res, next) => {
    let newUser = {
        name: req.body.name,
        id: req.body.id,
        password: req.body.password,
        tel: req.body.tel,
        addr: req.body.addr,
        email: req.body.email,
        indi: req.body.indi,

```

```

        crn: req.body.crn,
        company: req.body.company,
        seller: req.body.seller
    };
    console.log(newUser);
    let connection1;
    CreateSalt() // Salt값 생성 함수 호출
        .then(function(resSalt) { // CreateSalt 함수가 resSalt를 반환한 것을 받음
            return PasswordHash(resSalt, newUser.password); // PasswordHash 함수호출
resSalt, 패스워드를 같이 보냄
        })
        .then(function(resulthash) { //PasswordHash함수가 resulthash값을 반환
            var newUserHash = { //newUserHash에 유저정보, 해쉬화한 비밀번호를 json형태
로 담는다
                newUser: newUser,
                hash: resulthash
            };
            return CreateRegisterQuery(newUserHash); // CreateQuery함수에 newUserHash를
보내며 호출
        })
        .then(function(query) { // CreateQuery함수에서 쿼리문을 반환
            return PoolGetConnection(query); // PoolGetConnection에 쿼리문을 보냄
            // 커넥션을 얻는데 쿼리문은 필요가 없지만 뒤에 사용될 함수가 커넥션을 사용
하므로 다음 함수에 쿼리문을 전달하기 위해서 쿼리문을 보냄
        })
        .then(function(connectionQuery) {
            return ExecuteQuery(connectionQuery)
        })
        .then(function(rows) {
            var entuser = {
                ent_num: rows,
                newUser: newUser
            };
            return CreateQueryEnt(entuser);
        })
        .then(function(query) { // CreateQuery함수에서 쿼리문을 반환
            return PoolGetConnection(query); // PoolGetConnection에 쿼리문을 보냄
            // 커넥션을 얻는데 쿼리문은 필요가 없지만 뒤에 사용될 함수가 커넥션을 사용
하므로 다음 함수에 쿼리문을 전달하기 위해서 쿼리문을 보냄
        })
        .then(function(connectionQuery) {
            return ExecuteQuery(connectionQuery)
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return CreateShoppingCart(newUser.id);
        })
        .then( function (cartQuery) {
            return PoolGetConnection(cartQuery);
        })
        .then(function (connectionQuery) {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return CreateCertTable(newUser.id);
        })
        .then( function (certQuery) {
            return PoolGetConnection(certQuery);
        })
        .then(function (connectionQuery) {

```

```

        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return RegComplete(res); // RegComplete에 res를 보냄. res.json을 실행하기 위
해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Excute Query err : " + err);
        return Rollback(connection1); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .then(function() {
        return ReleaseConnection(connection1); // 결과값이 어떻든 커넥션은 반환되어야
한다
    })
    .catch(function(err) { //마지막으로 에러를 캐치
        console.log(err);
        res.json({ success: false, msg: 'Failed to register user' }); // 에러 캐치시 false반
환
    })
});

// Authenticate
router.post('/authenticate', (req, res, next) => {
    const id = req.body.id;
    const password = req.body.password;
    console.log("입력받은 id : " + id);
    console.log("입력받은 password : " + password);

    CreateUserFoundQuery(id)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(rows => {
            return BcryptCompare(password, rows[0]);
        })
        .then( isMatch => {
            return CreateAuthToken(isMatch.user);
        })
        .then( AuthToken => {
            return LoginComplete( res, AuthToken );
        })
        .catch(err => {
            console.log(err);
            res.json({success: false, msg: err });
        })
});

// Profile
router.get('/profile', passport.authenticate("jwt", {session: false}), function(req, res) {

    const ptoken = req.headers.authorization;
    const currT = req.headers.ctime;
    const auth = req.headers.auth;
    delete req.user.password;

    const stoken = 'JWT '+jwt.sign({data: ptoken}, config.secret, {
        noTimestamp: true
    });

```

```

});

var md = forge.md.sha256.create();
md.update(currT+token);
const auth2 = md.digest().toHex();
const serverTime = new Date().getTime();
const diff = serverTime - currT;
console.log('수신한 일회용 인증 : '+auth);
console.log('계산한 일회용 인증 : '+auth2);
console.log('시간 차이 : '+diff);
if(auth == auth2 && diff<100000){
    res.json({user: req.user});
}
});

router.post('/addBasket', (req, res, next) => {
    let product = {
        userid : req.body.userid,
        number: req.body.number,
        price: req.body.price,
        productcode: req.body.productcode,
        productname: req.body.productname,
        quantity: req.body.quantity,
        seller: req.body.seller
    };

    addBasketQuery(product)          // Salt값 생성 함수 호출
    .then( function(query) {
        console.log("query : " + query);
        return PoolGetConnection(query);
    })
    .then(function (connectionQuery) {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) {
        console.log("This Solutions is : " + JSON.stringify(rows));
        return Complete(res);
    }, function(err) {
        console.log("err 1 : "+err);
        res.json({success: false, msg : err });
    })
    .catch(function (err) {
        console.log(err);
        res.json({success: false, msg : err });
    })
});

router.post('/FindUsername', (req, res, next) => {
    number = req.body.number;

    UsernameFoundQuery(number)          // Salt값 생성 함수 호출
    .then( function(query) {
        console.log("query : " + query);
        return PoolGetConnection(query);
    })
    .then(function (connectionQuery) {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) {
        console.log("This Solutions is : " + JSON.stringify(rows));
    })
});

```

```

        return Complete(res, rows);
    }, function(err) {
        console.log("err 1 : "+err);
        res.json({success: false, msg : err });
    })
    .catch(function (err) {
        console.log(err);
        res.json({success: false, msg : err });
    })
});

var pool = mysql.createPool(config); //연결에 대한 풀을 만든다. 기본값은 10개

function CreateShoppingCart(id) {
    return new Promise( function ( resolve ) {
        console.log(id);
        var aa = "CREATE TABLE `shoppingcart_"+ id + "` ( `number` INT NOT NULL
        AUTO_INCREMENT, `productcode` INT NOT NULL, `productname` VARCHAR(250) NOT NULL,
        `quantity` INT NOT NULL, `price` INT NOT NULL, `seller` VARCHAR(45) NOT NULL, PRIMARY
        KEY (`number`));";
        resolve(aa);
    })
}

function CreateCertTable(id) {
    return new Promise( function ( resolve ) {
        console.log(id);
        var aa = "CREATE TABLE `cert_"+ id + "` ( `certnumber` INT NOT NULL
        AUTO_INCREMENT, `masterCert` BOOLEAN NOT NULL DEFAULT FALSE, `allowed` BOOLEAN
        NOT NULL DEFAULT FALSE, `cert` TEXT(2000) NOT NULL, PRIMARY KEY (`certnumber`));";
        //var aa = "CREATE TABLE `cert_"+ id + "` ( `certnumber` INT NOT NULL
        AUTO_INCREMENT, `masterCert` BOOLEAN NOT NULL DEFAULT FALSE, `cert` TEXT(2000)
        NOT NULL, PRIMARY KEY (`certnumber`), UNIQUE (`cert`));";
        resolve(aa);
    })
}

function CreateSalt() {    //salt값을 생성하는 Promise 함수
    return new Promise( function (resolve, reject) {
        bcrypt.genSalt(10, function (err, salt) {
            if (salt) {
                console.log("CreateSalt : "+salt);
                resolve(salt);
            } else {
                console.log("CreateSalt err : "+err);
                reject(err);
            }
        });
    });
}

function PasswordHash(salt, pass) {    //비밀번호를 해쉬화하는 Promise 함수
    return new Promise( function (resolve, reject) {
        bcrypt.hash(pass, salt, function (err, hash) {
            if (hash) {
                console.log("PasswordHash : "+hash);
                resolve(hash);
            } else {
                console.log("PasswordHash err : "+err);
                reject(err);
            }
        })
    })
}

```



```

    });
  });
}

function CreateRegisterQuery(newUserHash) { //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을 작성하는 Promise 함수
  return new Promise( function (resolve, reject) {
    if(newUserHash) {
      let statement = "INSERT INTO user (name, id, password, tel, addr, email, indi, money) VALUES ('" + newUserHash.newUser.name + "', '" + newUserHash.newUser.id + "', '" + newUserHash.hash + "', '" + newUserHash.newUser.tel + "', '" + newUserHash.newUser.addr + "', '" + newUserHash.newUser.email + "', '" + newUserHash.newUser.indi + "', " + 1000000 + "');";
      resolve(statement);
    } else {
      console.log("CreateRegisterQuery err : "+err);
      reject(err);
    }
  });
}

function CreateQueryEnt(entuser) { //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을 작성하는 Promise 함수
  return new Promise(function(resolve, reject) {
    if (entuser) {
      let statement = "INSERT INTO ent (number ,crn, company, seller) VALUES ('" + entuser.ent_num.insertId + "', '" + entuser.newUser.crn + "', '" + entuser.newUser.company + "', '" + entuser.newUser.seller + "');";
      console.log(statement);
      resolve(statement, entuser.connection);
    } else {
      console.log("CreateQuery err : " + err);
      reject(err);
    }
  });
}

function CreateUserFoundQuery(Userid) { //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을 작성하는 Promise 함수
  return new Promise( function (resolve, reject) {
    if(Userid) {
      let statement = "SELECT * FROM user WHERE id='" + Userid + "';";
      console.log("CreateUserFoundQuery : "+statement);
      resolve(statement);
    } else {
      console.log("CreateUserFoundQuery err : "+err);
      reject(err);
    }
  });
}

function UsernameFoundQuery(number) { //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을 작성하는 Promise 함수
  return new Promise( function (resolve, reject) {
    if(number) {
      let statement = "SELECT name FROM user WHERE number=" + number + "';";
      console.log("CreateUserFoundQuery : "+statement);
      resolve(statement);
    } else {
      console.log("UsernameFoundQuery err : "+err);
    }
  });
}

```

```

        reject(err);
    }
    });
}

function addBasketQuery(product) {    //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을
작성하는 Promise 함수
    return new Promise( function (resolve, reject) {
        if(product) {
            let statement = "INSERT INTO shoppingcart_"+product.userid+" (price,
productcode, productname, quantity, seller) VALUES ('" + product.price + "', '" +
product.productcode + "', '" + product.productname + "', '" + product.quantity + "', '" +
product.seller + "');"
            console.log("addBasketQuery : "+statement);
            resolve(statement);
        } else {
            console.log("addBasketQuery err : "+err);
            reject(err);
        }
    });
}

function PoolGetConnection(query) {    //Pool에서 Connection을 가져오는 Promise 함수
return new Promise( function (resolve, reject) {
    pool.getConnection(function (err, connection) {
        if(connection){
            var connectionQuery = {
                connection : connection,
                query : query
            };
            resolve(connectionQuery);
        } else {
            console.log("PoolGetConnection err : "+err);
            reject(err);
        }
    });
});
}

function ExecuteQuery(ConQue) {    // Connection과 쿼리문을 받아와서 실행하는 Promise
함수
    return new Promise( function (resolve, reject) {
        ConQue.connection.query(ConQue.query, function(err, rows, fields) {
            if (!err) {
                console.log("query 실행 결과 : "+ JSON.stringify(rows));
                resolve(rows);
            } else {
                console.log("query 실행 err : "+err);
                reject(err);
            }
        });
        ConQue.connection.release();
    });
});
}

function BcryptCompare ( password, User ) {
    return new Promise( function (resolve, reject) {
        bcrypt.compare(password, User.password, function(err, isMatch) {
            if (isMatch === true) {
                console.log("패스워드 일치 : "+ isMatch);
                let isMatchUser = {

```

```

        result : isMatch,
        user : User
    };
    resolve(isMatchUser);
} else {
    console.log("BcryptCompare err : "+ err);
    reject(err);
}
});
});
}

function CreateAuthToken(User) {
    return new Promise( function ( resolve ) {
        const ptoken = 'JWT '+jwt.sign(
            { data : User },
            config.secret,
            { expiresIn : 86400 * 7 } //유효기간 7일
        );
        console.log("공개 토큰값 : ", ptoken);

        const stoken = 'JWT '+jwt.sign(
            { data : ptoken },
            config.secret,
            { noTimestamp : true } //유효기간 무제한
        );
        console.log("비밀 토큰값 : ", stoken);

        let AuthToken = {
            ptoken : ptoken,
            stoken : stoken,
            user : User
        };
        resolve(AuthToken);
    })
}

function LoginComplete( res, AuthToken ) {
    return new Promise( function () {
        res.json({
            success : true,
            ptoken : AuthToken.ptoken,
            stoken : AuthToken.stoken
        });
        console.log("로그인 성공");
    })
}

function RegComplete(res) { // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        res.json({success: true, msg: 'User registd'});
    });
}

function Complete(res, rows) { // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        res.json({ success: true, result : rows });
    });
}

```

```

function Rollback(connection) {      // 쿼리문 에러시 롤백을 실행하는 Promise 함수
    return new Promise( function () {
        connection.rollback(function () {
            console.error('rollback error');
        });
    });
}

function ReleaseConnection(connection) {    // 쿼리문을 다 실행한 후 Connection을 반환하
는 Promise 함수
    return new Promise( function () {
        connection.release();
    });
}

module.exports = router;

```

```

<stores.js>
const express = require('express');
const router = express.Router();
const passport = require('passport');
const jwt = require('jsonwebtoken');
const mysql = require('mysql');
const config = require('../config/database');
const bcrypt = require('bcryptjs');

const forge = require('node-forge');
const fs = require('fs');

router.post('/newStore', (req, res, next) => {
    let newStore = {
        seller: req.body.seller,
        name: req.body.name,
        price: req.body.price,
        quantity: req.body.quantity,
        category: req.body.category,
        description: req.body.description,
        number : req.body.number,
        thumbnail : req.body.image
    };

    newStore.thumbnail = (newStore.thumbnail).replace(/"/g, "");

    CreateStoreQuery(newStore)          // Salt값 생성 함수 호출
    .then( function(query) {
        console.log("query : " + query)// CreateQuery함수에서 쿼리문을 반환
        return PoolGetConnection(query);    // PoolGetConnection에 쿼리문을 보냄
        // 커넥션을 얻는데 쿼리문은 필요가 없지
        만 뒤에 사용될 함수가 커넥션을 사용하므로 다음 함수에 쿼리문을 전달하기 위해서 쿼리문을
        보냄
    })
    .then(function (connectionQuery) { // PoolGetConnection에서 커넥션과 쿼리문을 받
    음
        return ExecuteQuery(connectionQuery);    // ExecuteQuery함수에 커넥션과 쿼리
        문을 보냄
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return StoreComplete(res);    // RegComplete에 res를 보냄. res.json을 실행하기
        위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    })

```

```

    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("err 1 : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떠한
        커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
        res.json({success: false, msg: 'Failed to register user'});
    })
});

/*-----상품 보기 & 바로 구매-----*/
router.post('/GetProductDetail', (req, res, next) => {

    const productcode = req.body.productcode;

    CreateFindProductCodeQuery(productcode)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return Complete(res, rows); // RegComplete에 res를 보냄. res.json을 실행하
            기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
            false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떠한
            커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
});

router.post('/GetProductDetail2', (req, res, next) => {

    const productcode = req.body.productcode;
    console.log(productcode);

    var p1 = productcode.split(',');
    var p2 = new Array;
    var p3 = new Array;
    for (var i=0; i<p1.length; i++) {
        p2.push(p1[i].split('/'));
        p3.push(p2[i][0]);
    }
    console.log('p2 : '+JSON.stringify(p2));

```

```

console.log('p3 : '+JSON.stringify(p3));

CreateFindProductCodeQuery2(p3)
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return Complete(res, rows); // RegComplete에 res를 보냄. res.json을 실행하
기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Query Excute err : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .catch( function (err) { // 전체적으로 에러를 캐치한다
        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
false반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떻든
커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/GetProductDetail3', (req, res, next) => {

    const productcode = req.body.productcode;
    console.log(productcode);
    var p1 = productcode.split('/');

    CreateFindProductCodeQuery2(p1)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return Complete(res, rows); // RegComplete에 res를 보냄. res.json을 실행하
기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떻든
커넥션은 반환되어야 한다
        })

```

```

        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

router.post('/GetProductDetail4', (req, res, next) => {

    const products = req.body.products;
    console.log(products);

    CreateFindProductCodeQuery2(products)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return Complete(res, rows); // RegComplete에 res를 보냄. res.json을 실행하
기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어땠든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

/*-----장바구니 구매-----*/
router.post('/GetProductOder', (req, res, next) => {

    const product = req.body.product;
    const id = req.body.id;
    console.log('product1 : ' + JSON.stringify(product));
    console.log('id : ' + JSON.stringify(id));

    var b = "";
    var c = "";

    for (var i=0; i<(product.length); i++) {
        console.log('product2 : ' + JSON.stringify(product[i].productcode));
        var a = product[i].productcode;
        console.log(a);

        c = c+b.concat(a+', '+0);

        console.log('c :'+c);
        console.log('b' +b);
    }
}

```

```

console.log(c);

CreateFindCartProductQuery(id, c)
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return Complete(res, rows); // RegComplete에 res를 보냄. res.json을 실행하
기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Query Excute err : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .catch( function (err) { // 전체적으로 에러를 캐치한다
        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
false반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떤든
커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

function CreateFindCartProductQuery(id, c) {
    return new Promise( function (resolve) {
        console.log('product2 : '+JSON.stringify(c));
        let statement = "SELECT * FROM shoppingcart_"+id+" where productcode IN
("+c+") ";
        console.log("CreateFindProductCodeQuery : "+statement);
        resolve(statement);
    });
}
/*-----*/

router.post('/Product', (req, res, next) => {

    CreateProductFoundQuery()
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다

```



```

        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떨든
        커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/Store', (req, res, next) => {

    CreateStoreFoundQuery()
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return GetStoreComplete(res, rows);    // RegComplete에 res를 보냄. res.json을
        실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Query Excute err : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .catch( function (err) {    // 전체적으로 에러를 캐치한다
        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떨든
        커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/FoundEnt', (req, res, next) => {

    const number = req.body.number;

    CreateFoundEntQuery(number)
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows[0]));
        return GetStoreComplete(res, rows[0]);    // RegComplete에 res를 보냄. res.json
        을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("Query Excute err : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
});

```

```

        .catch( function (err) {    // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떨든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

router.post('/FindProduct', (req, res, next) => {

    const store = req.body.store;
    console.log("This Solutions is : " + store);

    CreateFindProductQuery(store)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows);    // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) {    // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Find ALL Product'}); // 에러 캐치시 false
반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떨든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

router.post('/FindCategory', (req, res, next) => {

    const category = req.body.category;
    console.log("category is : " + category);

    CreateFindCategoryQuery(category)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
    });

```

```

        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Found Category'}); // 에러 캐치시 false반
환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떻든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

router.post('/MyProduct', (req, res, next) => {

    const number = req.body.number;

    console.log("This number is : " + number);

    MyCreateProductFoundQuery(number)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("나누기 1 err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떻든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
    });

function MyCreateProductFoundQuery(number) {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM product where user_number='"+number+"'";
        console.log("CreateStoreFoundQuery : "+statement);
        resolve(statement);
    });
}

```

```

    });
}

router.post('/MyProduct2', (req, res, next) => {

    const number = req.body.number;

    console.log("This number is : " + number);

    MyCreateProductFoundQuery2(number)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("나누기 1 err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떠한
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
});

function MyCreateProductFoundQuery2(number) {
    return new Promise( function (resolve) {
        let statement = "SELECT productcode, productname, description, category,
user_number, seller, price, thumbnail FROM product where user_number='"+number+"'";
        console.log("CreateStoreFoundQuery : "+statement);
        resolve(statement);
    });
}
/*-----*/

router.post('/GetCart', (req, res, next) => {

    const id = req.body.number;

    CreateGetCartQuery(id)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetStoreComplete(res, rows); // RegComplete에 res를 보냄. res.json을

```

```

실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("나누기 1 err : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .catch( function (err) { // 전체적으로 에러를 캐치한다
        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection ); // 결과값이 어떨든
        커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

function CreateGetCartQuery(id) {
    return new Promise( function (resolve) {
        console.log('id : '+id);
        let statement = "SELECT * FROM shoppingcart_"+id+"";
        console.log("CreateGetCartQuery : "+statement);
        resolve(statement);
    });
}
/*-----*/

function CreateFindCategoryQuery(category) {
    return new Promise( function (resolve) {
        console.log('category : '+category);
        let statement = "SELECT * FROM product where category='"+category+"'";
        console.log("CreateFindCategoryQuery : "+statement);
        resolve(statement);
    });
}

function CreateFindProductCodeQuery(productcode) {
    return new Promise( function (resolve) {
        console.log('productcode : '+productcode);
        let statement = "SELECT * FROM product where productcode='"+productcode+"'";
        console.log("CreateFindProductCodeQuery : "+statement);
        resolve(statement);
    });
}

function CreateFindProductCodeQuery2(productcode) {
    return new Promise( function (resolve) {
        console.log('productcode : '+productcode);
        let statement = "SELECT * FROM product WHERE productcode IN
        ('"+productcode+"')";
        console.log("CreateFindProductCodeQuery2 : "+statement);
        resolve(statement);
    });
}

function CreateProductFoundQuery() {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM product";
        console.log("ALLProductFoundQuery : "+statement);
        resolve(statement);
    });
}

```

```

    });
}

function CreateStoreFoundQuery() {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM ent WHERE seller = 1";
        console.log("CreateStoreFoundQuery : "+statement);
        resolve(statement);
    });
}

function CreateFoundEntQuery(number) {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM ent where number = " + number + ";";
        console.log("CreateFoundEntQuery : "+statement);
        resolve(statement);
    });
}

function CreateFindProductQuery(store) {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM product where user_number = " + store + ";";
        console.log("CreateFindProductQuery : "+statement);
        resolve(statement);
    });
}

function CreateStoreQuery(newStore) { //유저 정보, 해쉬화된 비밀번호를 받아서 쿼리문을
    //작성하는 Promise 함수
    return new Promise( function (resolve, reject) {
        if(newStore) {
            let statement = "INSERT INTO product (user_number, seller, productname,
            price, allquantity, category, description, thumbnail) VALUES ('" + newStore.number + "', '" +
            newStore.seller + "', '" + newStore.name + "', '" + newStore.price + "', '" +
            newStore.quantity + "', '" + newStore.category + "', '" + newStore.description + "', '" +
            newStore.thumbnail + "');";
            resolve(statement);
        } else {
            console.log("CreateRegisterQuery err : "+err);
            reject(err);
        }
    });
}

function PoolGetConnection(query) { //Pool에서 Connection을 가져오는 Promise 함수
    return new Promise( function (resolve, reject) {
        console.log("PoolGetConnection 1");
        pool.getConnection(function (err, connection) {
            if(connection){
                var connectionQuery = {
                    connection : connection,
                    query : query
                };
                console.log("PoolGetConnection 2");
                resolve(connectionQuery);
            } else {
                console.log("PoolGetConnection err : "+err);
                reject(err);
            }
        });
    });
}

```

```

    });
}

function ExecuteQuery(ConQue) {    // Connection과 쿼리문을 받아와서 실행하는 Promise 함수
    return new Promise( function (resolve, reject) {
        ConQue.connection.query(ConQue.query, function(err, rows, fields) {
            if (!err) {
                console.log("ExecuteQuery : "+ JSON.stringify(rows));
                resolve(rows);
            } else {
                console.log("ExecuteQuery err : "+err);
                reject(err);
            }
        });
        ConQue.connection.release();
    });
}

function Complete(res, rows) {    // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        console.log('성공 : '+ JSON.stringify(rows));
        res.json({
            success: true,
            result : rows
        });
    });
}

function GetStoreComplete(res, rows) {    // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        console.log('마지막 : '+ JSON.stringify(rows));
        res.json({
            success: true,
            store : rows
        });
    });
}

function GetProductComplete(res, rows) {    // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        res.json({
            success: true,
            Product : rows
        });
    });
}

function StoreComplete(res) {    // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        res.json({success: true, msg: 'User registd'});
    });
}

function Rollback(connection) {    // 쿼리문 에러시 롤백을 실행하는 Promise 함수
    return new Promise( function () {
        connection.rollback(function () { //쿼리가 에러로 실패하면 롤백해야 함

```

```

        console.error('rollback error1');
    });
}

function ReleaseConnection(connection) {    // 쿼리문을 다 실행한 후 Connection을 반환하
    // 는 Promise 함수
    return new Promise( function () {
        connection.release();
    });
}

var pool = mysql.createPool(config); //연결에 대한 풀을 만든다. 기본값은 10개
module.exports = router;

```

```

<Pay.js>
const express = require('express');
const router = express.Router();
const passport = require('passport');
const jwt = require('jsonwebtoken');
const mysql = require('mysql');
const config = require('../config/database');
const passport_policy = require('../config/passport');
const bcrypt = require('bcryptjs');
const forge = require('node-forge');
const fs = require('fs');
const pki = forge.pki;

const caCertPem = fs.readFileSync('caCert.pem', 'utf8');
const caPrivateKeyPem = fs.readFileSync('caPrivateKey.pem', 'utf8');
const caCert = pki.certificateFromPem(caCertPem);
const caPrivateKey = pki.privateKeyFromPem(caPrivateKeyPem);

router.post('/procpay',(req, res, next) => {
    let number = req.body.Payinfo.order_no;
    let jwt = req.body.token;
    let Statement;
    procpay(number, jwt)
});

router.post('/GetOrder',(req, res, next) => {
    let order_no = req.body.orderno;
    console.log('order_number : '+order_no);

    //var order_no = order_no.split('/');
    //console.log('o : '+JSON.stringify(order_no));

    OrderFoundQuery(order_no)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            return Complete( res, rows );
        })
        .catch(err => {
            console.log(err);
        })
    }

```



```

        res.json({success: false});
    })

});

router.post('/GetOrder_2',(req, res, next) => {

    let order_no = req.body.order_no;
    let t1 = req.body.time;
    console.log('order_number : '+order_no);
    console.log('time1 : '+t1);
    let t2 = new Date().getTime();
    console.log('time2 : '+t2);

    if((t2 - t1) > (1000 * 60 * 5)) {
//if((t2 - t1) > (1000 * 5)) {
        console.log('유효시간 초과');
        res.json({
            success: false,
            msg : '유효시간 초과'
        });
    } else {
        OrderFoundQuery(order_no)
            .then( query => {
                return PoolGetConnection(query);
            })
            .then(connectionQuery => {
                return ExecuteQuery(connectionQuery);
            })
            .then( rows => {
                return Complete( res, rows );
            })
            .catch(err => {
                console.log(err);
                res.json({success: false});
            })
    }
});

router.post('/GetOrder_3',(req, res, next) => {
    let number = req.body.number;

    UserOrderFoundQuery(number)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            return Complete( res, rows );
        })
        .catch(err => {
            console.log(err);
            res.json({success: false, msg : err });
        })
});

function UserOrderFoundQuery(number) {
    return new Promise( function (resolve, reject) {
        if(number) {

```

```

        let statement = "SELECT * FROM trade_detail WHERE orderer=" + number + "
OR buyer=" + number + ",";
        console.log("UserOrderFoundQuery : "+statement);
        resolve(statement);
    } else {
        console.log("UserOrderFoundQuery err : "+err);
        reject(err);
    }
    });
}

router.post('/GetOrder_4',(req, res, next) => {
    let products = req.body.products;

    console.log('aaa : '+products);

    let bbb = [];

    for (let i = 0; i < products.length; i++) {
        let ccc = {
            productcode1 : products[i]+"W/%",
            productcode2 : "%,"+products[i]+"W/%"
        };

        bbb.push(ccc);
        //ccc.productcode1 = "";
        //ccc.productcode2 = "";
    }

    console.log("bbb : " + JSON.stringify(bbb));

    UserOrderFoundQuery2(bbb)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery2(connectionQuery);
        })
        .then( rows => {
            return Complete( res, rows );
        })
        .catch(err => {
            console.log(err);
            res.json({success: false, msg : err });
        })
    });

function UserOrderFoundQuery2(bbb) {
    return new Promise( function (resolve, reject) {
        if(bbb) {

            let statement = new Array;
            for (var i=0; i<bbb.length; i++){
                statement.push("SELECT order_no, product, orderer, buyer, price, paid
FROM trade_detail WHERE product LIKE '"+ bbb[i].productcode1 +" OR product LIKE '"+
bbb[i].productcode2 +"");
            }
            console.log('statement 2 : '+JSON.stringify(statement));

            //let statement = "SELECT * FROM trade_detail WHERE product LIKE ? OR
product LIKE ?";

```

```

        //let statement = "SELECT * FROM trade_detail WHERE product IN('5W/', '7W/',
        '5W/', '7W/';";
        console.log("UserOrderFoundQuery2 : "+statement);
        resolve(statement);
    } else {
        console.log("UserOrderFoundQuery2 err : "+err);
        reject(err);
    }
    });
}

router.post('/newOrder',(req, res, next) => {

    console.log(JSON.stringify(req.body));
    let newOrder = {
        //product: req.body.product[0].product,
        //price: req.body.product[0].price,
        product : "",
        price : 0,
        orderer: req.body.orderer,
        delivery_address: req.body.delivery_address,
        delivery_tel: req.body.delivery_tel
    };

    for(var i=0; i< req.body.product.length; i++ ) {
        if(i==0){
            newOrder.product = req.body.product[i].product;
            newOrder.price = req.body.product[i].price;
        } else {
            newOrder.product = newOrder.product+', '+req.body.product[i].product;
            newOrder.price = newOrder.price + req.body.product[i].price;
        }
        console.log('new order product : '+newOrder.product);
        console.log('new order price : '+newOrder.price);
    }

    console.log('newOrder : '+JSON.stringify(newOrder));

    CreateOrderQuery(newOrder)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            return Complete( res, rows );
        })
        .catch(err => {
            console.log(err);
            res.json({success: false});
        })
    );

    router.post('/Trade',(req, res, next) => {
        let signatureHex = req.body.signature;
        let Request = req.body.Request;
        let currentTime1 = req.body.currentT;
    }

```

```

console.log('Request : '+JSON.stringify(Request));

const cert = pki.certificateFromPem(req.body.cert);
const signature = forge.util.hexToBytes(signatureHex);
const publicKey = cert.publicKey;
const serialNumber = cert.serialNumber;
const commonCert = cert.subject.getField('CN').value;
let DbCertPem;

var pss;
var md;
var verify0;
var verify1;
var verify2;
var verify3;
var verify4;
var p;
var p1;
var p2;
var p3;
var p4;
var p5;

FindCertQuery(serialNumber, Request.id)
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then( rows => {
        //console.log(rows);
        DbCertPem = rows[0].cert;
        //console.log("DbCertPem : "+DbCertPem);

        const DbCert = pki.certificateFromPem(DbCertPem);
        const DbPublicKey = DbCert.publicKey;
        const DbCommonCert = DbCert.subject.getField('CN').value;
        //받은 인증서가 DB 인증서 테이블에 존재 하는지 확인
        //그리고 DB의 인증서와 받은 인증서가 같은지 확인

        //console.log("Cert : "+JSON.stringify(cert));
        //console.log("DbCert : "+JSON.stringify(DbCert));
        //console.log(JSON.stringify(cert) == JSON.stringify(DbCert));

        if( JSON.stringify(cert) == JSON.stringify(DbCert) ) {
            verify0 = true;
            console.log('Signature Verify0 : '+verify0);

            const currentTime2 = new Date().getTime();
            const diff = currentTime2 - currentTime1;

            // verify RSASSA-PSS signature
            pss = forge.pss.create({
                md: forge.md.sha1.create(),
                mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
                saltLength: 20
                // optionally pass 'prng' with a custom PRNG implementation
            });
            md = forge.md.sha1.create();
            md.update(Request, 'utf8');

```

```

md.update(currentTime1, 'utf8');
verify1 = DbPublicKey.verify(md.digest().getBytes(), signature, pss);
console.log('Signature Verify1 : '+verify1);
verify2 = caCert.verify(DbCert);
console.log('Signature Verify2 : '+verify2);

if(diff < 1000 * 30) { //30초
    verify3 = true;
    console.log('Signature Verify3 : '+verify3);
}
}
버) 비교
if ( Request.unum == DbCommonCert ) { //인증서 내부의 CN 필드(유저넘
    verify4 = true;
    console.log('Signature Verify4 : '+verify4);
}

if( verify0==true && verify1==true && verify2==true && verify3==true
&& verify4==true) {

    p = Request.order.product;

    console.log('p : '+p);

    p1 = p.split(',');

    /*
    if(p.length >= 1) {
        p1 = p.split(',');
    } else {
        p1 = p;
    }
    */

    p2 = new Array;
    p3 = new Array;
    p4 = new Array;
    for (var i=0; i<p1.length; i++) {
        p2.push(p1[i].split('/'));
        p3.push(p2[i][0]);
        p4.push(p2[i][1]);
    }

    for (var j=0; j<p3.length; j++){
        //p3[j] *= 1;
        //p4[j] *= 1;

        for (var k=0; k<p4.length; k++){
            //p2[j][k] *= 1;
        }
    }
    console.log('p2 (1) : '+p2);
    console.log('p2 (2) : '+JSON.stringify(p2));
    console.log('p3 : '+JSON.stringify(p3)); // 코드
    console.log('p4 : '+JSON.stringify(p4)); // 수량

    // 1. 구매한 상품들에서 수량 빼고
    let statement = new Array;
    for (var i=0; i<p2.length; i++){
        statement.push("UPDATE
                                product
                                SET
allquantity=allquantity-"+p2[i][1]+" WHERE productcode="+p2[i][0]+"";");
    }

```

```

        console.log('statement : '+JSON.stringify(statement));
        return PoolGetConnection(statement);
        // 실패 시 모두 롤백
    } else {
        console.log("Signature Verify err");
        return res.json({success:false, msg: 'Signature Verify Err'});
    }

    } else {
        return res.json({success:false, msg: 'Certificate Validate Err'});
    }
})
.then(connectionQuery => {
    return ExecuteQuery2(connectionQuery);
})
.then( rows => {
    console.log("rows : "+JSON.stringify(rows));

    let statement = new Array;
    for (var i=0; i<p3.length; i++){
        statement.push("SELECT * FROM product WHERE productcode=" + p3[i] +
";");
    }
    console.log('statement 2 : '+JSON.stringify(statement));
    //return res.json({success:true, msg: '111'});

    return PoolGetConnection(statement);
})
.then(connectionQuery => {
    return ExecuteQuery2(connectionQuery);
})
.then( rows => {
    // 2. 판매자한테는 돈 넣고
    console.log("rows : "+JSON.stringify(rows));
    //console.log("rows user_number 0 : "+JSON.stringify(rows[0][0].user_number));
    //console.log("rows user_number 1 : "+JSON.stringify(rows[1][0].user_number));

    let statement = new Array;
    console.log("rows.length : "+rows.length);
    for (var i=0; i<rows.length; i++){
        for (var j=0; j<rows.length; j++){
            console.log("rows["+i+"]"+[0].productcode : "+rows[i][0].productcode);
            console.log("p3["+j+"]" : "+p3[j]);
            if (rows[i][0].productcode == p3[j]) {
                statement.push("UPDATE user SET money=money+" +
rows[i][0].price * p4[j] + " WHERE number="+rows[i][0].user_number+";");
            }
        }
    }
    console.log('statement 3 : '+JSON.stringify(statement));

    return PoolGetConnection(statement);
})
.then(connectionQuery => {
    return ExecuteQuery2(connectionQuery);
})
.then( rows => {
    // 3. 구매자한테서 돈 빼고
    console.log('rows : '+JSON.stringify(rows));

    let statement = new Array;

```

```

        statement.push("UPDATE user SET money=money-"+ Request.order.price + "
WHERE id='"+ Request.id + "','");

        console.log('statement 4 : '+JSON.stringify(statement));
        return PoolGetConnection(statement);
    })
    .then(connectionQuery => {
        return ExecuteQuery2(connectionQuery);
    })
    .then( rows => {
        // 4. 주문정보 테이블에 결제 정보 업데이트
        console.log('rows : '+JSON.stringify(rows));

        var time = new Date().getTime();

        let statement = new Array;
        statement.push("UPDATE trade_detail SET purchase_signature='"+ signatureHex
+ "", paid="+ 1 +", trade_time='"+ time +", buyer="+Request.unum+" WHERE order_no="+
Request.order_no + "','");

        console.log('statement 5 : '+JSON.stringify(statement));

        return PoolGetConnection(statement);
    })
    .then(connectionQuery => {
        return ExecuteQuery2(connectionQuery);
    })
    .then( rows => {
        console.log("rows : "+JSON.stringify(rows));

        let statement = new Array;
        statement.push("SELECT * FROM user WHERE number='"+ Request.order.orderer
+ "','");

        console.log('statement 6 : '+JSON.stringify(statement));

        return PoolGetConnection(statement);
    })
    .then(connectionQuery => {
        return ExecuteQuery2(connectionQuery);
    })
    .then( rows => {
        // 5. 장바구니에서 삭제
        console.log("rows : "+JSON.stringify(rows));

        let statement = new Array;
        for (var i=0; i<p3.length; i++){
            statement.push("DELETE FROM shoppingcart_"+rows[0][0].id+" WHERE
productcode="+ p3[i] + "','");
        }
        console.log('statement 7 : '+JSON.stringify(statement));

        return PoolGetConnection(statement);
    })
    .then(connectionQuery => {
        return ExecuteQuery2(connectionQuery);
    })
    .then( rows => {
        console.log("rows : "+JSON.stringify(rows));
        console.log("결제 성공");
        return res.json({success:true, order: Request.order_no });
    })

```

```

        .catch( function (err, connection) {
            console.log(err);
            //connection.rollback();
        });

    /*
    TradeQuery()
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            return Complete( res, rows );
        })
        .catch(err => {
            console.log(err);
            res.json({success: false});
        })
    */
});

function ExecuteQuery2(ConQue) {    // Connection과 쿼리문을 받아와서 실행하는 Promise
함수
    return new Promise( function (resolve, reject) {
        let aaa = new Array;
        console.log("ConQue : "+ConQue.query.length);
        let bbb = ConQue.query.length - 1;
        console.log('bbb : '+bbb);
        for (var i=0; i<(ConQue.query.length); i++) {
            ConQue.connection.query(ConQue.query[i], function(err, rows, fields) {
                if (!err) {
                    console.log("query 실행 결과 : " + JSON.stringify(rows));
                    aaa.push(rows);
                    console.log("query 실행 중 : " + JSON.stringify(aaa));
                    console.log('i : '+i);
                    if (i === aaa.length ) {
                        console.log("query 실행 끝 : " + JSON.stringify(aaa));
                        resolve(aaa);
                        ConQue.connection.release();
                    }
                } else {
                    console.log("query 실행 err : "+err);
                    reject(err);
                }
            });
        }
    });
}

/* ----- 영수증 발급 ----- */
router.post('/Receipt',(req, res, next) => {
    let signatureHex = req.body.signature;
    let Request = req.body.Request;
    let currentTime1 = req.body.currentT;

```



```

console.log('Request : '+JSON.stringify(Request));

const cert = pki.certificateFromPem(req.body.cert);
const signature = forge.util.hexToBytes(signatureHex);
const publicKey = cert.publicKey;
const serialNumber = cert.serialNumber;
const commonCert = cert.subject.getField('CN').value;
let DbCertPem;

var pss;
var md;
var verify0;
var verify1;
var verify2;
var verify3;
var verify4;
var p;
var p1;
var p2;
var p3;
var p4;
var p5;
let statementParams;

FindCertQuery(serialNumber, Request.user.id)
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then( rows => {
        //console.log(rows);
        DbCertPem = rows[0].cert;
        //console.log("DbCertPem : "+DbCertPem);

        const DbCert = pki.certificateFromPem(DbCertPem);
        const DbPublicKey = DbCert.publicKey;
        const DbCommonCert = DbCert.subject.getField('CN').value;
        //받은 인증서가 DB 인증서 테이블에 존재 하는지 확인
        //그리고 DB의 인증서와 받은 인증서가 같은지 확인

        //console.log("Cert : "+JSON.stringify(cert));
        //console.log("DbCert : "+JSON.stringify(DbCert));
        //console.log(JSON.stringify(cert) == JSON.stringify(DbCert));

        if( JSON.stringify(cert) == JSON.stringify(DbCert) ) {
            verify0 = true;
            console.log('Signature Verify0 : ' + verify0);

            const currentTime2 = new Date().getTime();
            const diff = currentTime2 - currentTime1;

            // verify RSASSA-PSS signature
            pss = forge.pss.create({
                md: forge.md.sha1.create(),
                mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
                saltLength: 20
            });
            // optionally pass 'prng' with a custom PRNG implementation
        }
    });

```

```

md = forge.md.sha1.create();
md.update(Request, 'utf8');
//md.update(currentTime1, 'utf8');
verify1 = DbPublicKey.verify(md.digest().getBytes(), signature, pss);
console.log('Signature Verify1 : ' + verify1);
verify2 = caCert.verify(DbCert);
console.log('Signature Verify2 : ' + verify2);

if (diff < 1000 * 30) { //30초
    verify3 = true;
    console.log('Signature Verify3 : ' + verify3);
}
저번호) 비교
if (Request.user.number == DbCommonCert) { //인증서 내부의 CN 필드(유

    verify4 = true;
    console.log('Signature Verify4 : ' + verify4);
}

if (verify0 == true && verify1 == true && verify2 == true && verify3 ==
true && verify4 == true) {
    console.log("서명 검증 완료");

    //여기서 서명값 문자열 만들어서 쿼리 생성 후 실행
    console.log('Request : '+JSON.stringify(Request));

    let sig01 = "";

    if ( Request.order.receipt == null ) { //아직 영수증이 하나도 없으면

        sig01 = Request.product + '+' + signatureHex;
        console.log('sig01 : '+sig01);

    } else {    // 먼저 발급된 영수증이 하나라도 있으면

        //sig01 = Request.order.receipt;
        //sig01 = sig01 + ';' + Request.product + '+' + signatureHex;
        sig01 =',' + Request.product + '+' + signatureHex;
        console.log('sig01 : '+sig01);
    }

    let statement = "UPDATE trade_detail SET receipt=CONCAT(receipt,?)
WHERE order_no=?;";
    statementParams = [sig01, Request.order.order_no];

    console.log('statement : '+statement);
    console.log('statementParams : '+statementParams);

    return PoolGetConnection(statement);
    // 실패 시 모두 롤백
} else {
    console.log("Signature Verify err");
    return res.json({success:false, msg: 'Receipt Verify Err'});
}

} else {
    return res.json({success:false, msg: 'Receipt Validate Err'});
}
})
.then(connectionQuery => {
    return ExecuteQuery3(connectionQuery, statementParams);
})

```

```

        .then( rows => {
            console.log("rows : "+JSON.stringify(rows));
            console.log("영수증 성공");
            return res.json({success:true, order: Request.order.order_no });
        })
        .catch( function (err, connection) {
            console.log(err);
            //connection.rollback();
        });

/*
TradeQuery()
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then( rows => {
        return Complete( res, rows );
    })
    .catch(err => {
        console.log(err);
        res.json({success: false});
    })
    */
});

// ----- 영수증 서명 검증 -----
router.post('/ReceiptValidate',(req, res, next) => {
    let products = req.body.product;
    console.log('products : '+products);

    let p = new Array;
    let p1 = new Array;
    p.push(products.split('-'));
    console.log('p : '+JSON.stringify(p));

    for (var i=0; i<p[0].length; i++) {
        console.log('p[0][i] : '+JSON.stringify(p[0][i]));
        p1.push(p[0][i].split('/'));
    }
    console.log('p1 : '+JSON.stringify(p1));

    FindProductCodeQuery(p1[0][0])
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            console.log('rows1 : '+JSON.stringify(rows));

            return ReceiptValidateQuery(rows[0].seller);
        })
        .then( query => {
            return PoolGetConnection(query);
        })

```

```

        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then( rows => {
            console.log('rows2 : '+JSON.stringify(rows));

            res.json({
                success: true,
                Cert : rows
            })
        })
        .catch(err => {
            console.log(err);
            res.json({success: false});
        })
    }

    /*
    ReceiptValidateQuery()
    .then( query => {
        return PoolGetConnection(query);
    })
    .then(connectionQuery => {
        return ExecuteQuery(connectionQuery);
    })
    .then( rows => {
        return Complete( res, rows );
    })
    .catch(err => {
        console.log(err);
        res.json({success: false});
    })
    */
});

function ReceiptValidateQuery(id) {
    return new Promise( function (resolve, reject) {
        if(id) {
            let statement = "SELECT cert FROM cert_"+ id +"";
            console.log("ReceiptValidateQuery : "+statement);
            resolve(statement);
        } else {
            console.log("ReceiptValidateQuery err : "+err);
            reject(err);
        }
    });
}

function FindProductCodeQuery(productcode) {
    return new Promise( function (resolve) {
        console.log('productcode : '+productcode);
        let statement = "SELECT * FROM product where productcode='"+productcode+"'";
        console.log("CreateFindProductCodeQuery : "+statement);
        resolve(statement);
    });
}

function procpay(number, jwt) {
    return new Promise( function (resolve, reject) {
        let temp = passport_policy(jwt);
        console.log(temp);
    })
}

```

```

}

function TradeQuery(ordernumber) {
    return new Promise( function (resolve, reject) {
        if(ordernumber) {
            let statement = "SELECT * FROM trade_detail WHERE order_no=" +
ordernumber + ",";
            console.log("TradeQuery : "+statement);
            resolve(statement);
        } else {
            console.log("TradeQuery err : "+err);
            reject(err);
        }
    });
}

function FindCertQuery(serial, id) {
    return new Promise( function (resolve, reject) {
        if(serial && id) {
            let statement = "SELECT * FROM cert_"+id+" WHERE certnumber=" + serial
+",";

            //statement = statement.replace(/\n/g, ""); //행바꿈제거
            // = statement.replace(/\r/g, ""); //엔터제거
            console.log("FindCertQuery : "+statement);
            resolve(statement);
        } else {
            console.log("FindCertQuery err : "+err);
            reject(err);
        }
    });
}

function OrderFoundQuery(ordernumber) {
    return new Promise( function (resolve, reject) {
        if(ordernumber) {
            let statement = "SELECT * FROM trade_detail WHERE order_no=" +
ordernumber + ",";
            console.log("OrderFoundQuery : "+statement);
            resolve(statement);
        } else {
            console.log("OrderFoundQuery err : "+err);
            reject(err);
        }
    });
}

function CreateOrderQuery(newOrder) {
    return new Promise( function (resolve) {
        let statement = "INSERT INTO trade_detail (product, price, orderer, paid,
delivery_address, delivery_tel) VALUES ('" + newOrder.product + "', '" + newOrder.price + "',
'" + newOrder.orderer + "', '" + 0 + "', '" + newOrder.delivery_address + "', '" +
newOrder.delivery_tel + "');";
        console.log("newOrderQuery : "+statement);
        resolve(statement);
    });
}

function PoolGetConnection(query) { //Pool에서 Connection을 가져오는 Promise 함수
    return new Promise( function (resolve, reject) {
        pool.getConnection(function (err, connection) {

```

```

        if(connection){
            var connectionQuery = {
                connection : connection,
                query : query
            };
            resolve(connectionQuery);
        } else {
            console.log("PoolGetConnection err : "+err);
            reject(err);
        }
    });
}

function ExecuteQuery(ConQue) {    // Connection과 쿼리문을 받아와서 실행하는 Promise 함수
    return new Promise( function (resolve, reject) {
        ConQue.connection.query(ConQue.query, function(err, rows, fields) {
            if (!err) {
                console.log("query 실행 결과 : "+ JSON.stringify(rows));
                resolve(rows);
            } else {
                console.log("query 실행 err : "+err);
                reject(err);
            }
        })
        ConQue.connection.release();
    });
}

function ExecuteQuery3(ConQue, Params) {    // Connection과 쿼리문을 받아와서 실행하는 Promise 함수
    return new Promise( function (resolve, reject) {
        ConQue.connection.query(ConQue.query, Params, function(err, rows, fields) {
            if (!err) {
                console.log("query3 실행 결과 : "+ JSON.stringify(rows));
                resolve(rows);
            } else {
                console.log("query3 실행 err : "+err);
                reject(err);
            }
        })
        ConQue.connection.release();
    });
}

function Complete(res, rows) {
    return new Promise( function () {
        console.log("Success");
        res.json({success: true, order: rows});
    });
}

function Rollback(connection) {    // 쿼리문 에러시 롤백을 실행하는 Promise 함수
    return new Promise( function () {
        connection.rollback(function () {
            console.error('rollback error');
        });
    });
}

```

```

function ReleaseConnection(connection) {    // 쿼리문을 다 실행한 후 Connection을 반환하
는 Promise 함수
    return new Promise( function () {
        connection.release();
    });
}

var pool = mysql.createPool(config); //연결에 대한 풀을 만든다. 기본값은 10개

module.exports = router;

```

```

<Cert.js>
const express = require('express');
const router = express.Router();
const passport = require('passport');
const jwt = require('jsonwebtoken');
const mysql = require('mysql');
const config = require('./config/database');
const bcrypt = require('bcryptjs');

const forge = require('node-forge');
const fs = require('fs');
const pki = forge.pki;
const caCertPem = fs.readFileSync('caCert.pem', 'utf8');
const caPrivateKeyPem = fs.readFileSync('caPrivateKey.pem', 'utf8');
const caCert = pki.certificateFromPem(caCertPem);
const caPrivateKey = pki.privateKeyFromPem(caPrivateKeyPem);

function pad(n, width) {
    n = n + "";
    return n.length >= width ? n : new Array(width - n.length + 1).join('0') + n;
}

router.post('/newCert', (req, res, next) => {
    console.log(JSON.stringify(req.body));
    let certinfo;

    let highestCertNumber;
    FindCertNumberHighestQuery(req.body.user.id)          // Salt값 생성 함수 호출
        .then( function(query) {
            //console.log("masterCert query : " + query);
            return PoolGetConnection(query);
        })
        .then(function (connectionQuery) {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) {
            console.log("FindCertNumberHighest is : " + JSON.stringify(rows));
            highestCertNumber = rows[0].max_certnumber;
            console.log('highestCertNumber 1 : '+highestCertNumber);

            highestCertNumber = pad(highestCertNumber+1, 2);
            console.log('highestCertNumber 2 : '+highestCertNumber);

            var cert = pki.createCertificate();
            cert.publicKey = pki.publicKeyFromPem(req.body.publicKey);
            cert.serialNumber = highestCertNumber;
            cert.validity.notBefore = new Date();
            cert.validity.notAfter.setFullYear(cert.validity.notBefore.getFullYear() + 5);

```

```

console.log('cert serial : '+cert.serialNumber);

var userAttrs = [
  {
    name: 'commonName',
    value: req.body.user.number
  }, {
    name: 'countryName',
    value: 'kr'
  }, {
    name: 'organizationName',
    value: 'Coconut'
  }, {
    shortName: 'OU',
    value: 'Master'
  }
];
cert.setSubject(userAttrs);

var caAttrs = [
  {
    name: 'commonName',
    value: caCert.subject.getField('CN').value
  }, {
    name: 'countryName',
    value: caCert.subject.getField('C').value
  }, {
    shortName: 'ST',
    value: caCert.subject.getField('ST').value
  }, {
    name: 'localityName',
    value: caCert.subject.getField('L').value
  }, {
    name: 'organizationName',
    value: caCert.subject.getField('O').value
  }, {
    shortName: 'OU',
    value: caCert.subject.getField('OU').value
  }
];
cert.setIssuer(caAttrs);

cert.setExtensions([
  {
    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
        }, {
            name: 'subjectAltName',
            altNames: [{
                type: 6, // URI
                value: 'http://coconutpay.herokuapp.com/'
            }]
        }, {
            name: 'subjectKeyIdentifier'
        }
    ]);

    cert.sign(caPrivateKey);

    certinfo = {
        cert : pki.certificateToPem(cert),
        caCert: caCertPem,
        user : req.body.user
    };
    return addMasterCertQuery(certinfo);
}, function(err) {
    console.log("err 1 : "+err);
})
.then( function(query) {
    console.log("masterCert query : " + query);
    return PoolGetConnection(query);
})
.then(function (connectionQuery) {
    return ExecuteQuery(connectionQuery);
})
.then(function(rows) {
    console.log("This Solutions is : " + JSON.stringify(rows));
    return MasterCertComplete(res, certinfo);
}, function(err) {
    console.log("err 1 : "+err);
    res.json({success: false, msg : err });
})
.catch(function (err) {
    console.log(err);
    res.json({success: false, msg : err });
});

});

function FindCertNumberHighestQuery(id) {
    return new Promise( function (resolve) {
        let statement = "SELECT MAX(certnumber) AS max_certnumber FROM cert_"+id+"";
        console.log("FindCertNumberHighestQuery : "+statement);
        resolve(statement);
    });
}

router.post('/AddCert', (req, res, next) => {

```

```

console.log(JSON.stringify(req.body));
var cert = pki.createCertificate();
cert.publicKey = pki.publicKeyFromPem(req.body.publicKey);
cert.serialNumber = '01'; // 이 부분은 시리얼 넘버가 점점 올라가도록 해야 함
cert.validity.notBefore = new Date();
cert.validity.notAfter.setFullYear(cert.validity.notBefore.getFullYear() + 1);

var userAttrs = [
  {
    name: 'commonName',
    value: req.body.user.number
  }, {
    name: 'countryName',
    value: 'kr'
  }, {
    name: 'organizationName',
    value: 'Coconut'
  }, {
    shortName: 'OU',
    value: req.body.deviceId //이 부분은 나중에 사용자가 입력한 기기 식별명으로
    변경
  }
];
cert.setSubject(userAttrs);

var caAttrs = [
  {
    name: 'commonName',
    value: caCert.subject.getField('CN').value
  }, {
    name: 'countryName',
    value: caCert.subject.getField('C').value
  }, {
    shortName: 'ST',
    value: caCert.subject.getField('ST').value
  }, {
    name: 'localityName',
    value: caCert.subject.getField('L').value
  }, {
    name: 'organizationName',
    value: caCert.subject.getField('O').value
  }, {
    shortName: 'OU',
    value: caCert.subject.getField('OU').value
  }
];
cert.setIssuer(caAttrs);

cert.setExtensions([
  {
    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
    }, {
        name: 'subjectAltName',
        altNames: [{
            type: 6, // URI
            value: 'http://coconutpay.herokuapp.com/'
        }]
    }, {
        name: 'subjectKeyIdentifier'
    }
    ]);

cert.sign(caPrivateKey);

let certinfo = {
    cert : pki.certificateToPem(cert),
    caCert: caCertPem,
    user : req.body.user
};

addMasterCertQuery(certinfo) // Salt값 생성 함수 호출
    .then( function(query) {
        console.log("masterCert query : " + query);
        return PoolGetConnection(query);
    })
    .then(function (connectionQuery) {
        return ExecuteQuery(connectionQuery);
    })
    .then(function(rows) {
        console.log("This Solutions is : " + JSON.stringify(rows));
        return MasterCertComplete(res, certinfo);
    }, function(err) {
        console.log("err 1 : "+err);
        res.json({success: false, msg : err });
    })
    .catch(function (err) {
        console.log(err);
        res.json({success: false, msg : err });
    });
});

router.post('/newStore', (req, res, next) => {
    let newStore = {
        seller: req.body.seller,
        name: req.body.name,
        price: req.body.price,
        quantity: req.body.quantity,

```

```

        category: req.body.category,
        description: req.body.description,
        number : req.body.number
    };

    CreateStoreQuery(newStore)          // Salt값 생성 함수 호출
    .then( function(query) {
        console.log("query : " + query)// CreateQuery함수에서 쿼리문을 반환
        return PoolGetConnection(query); // PoolGetConnection에 쿼리문을 보냄
        // 커넥션을 얻는데 쿼리문은 필요가 없지
        // 만 뒤에 사용될 함수가 커넥션을 사용하므로 다음 함수에 쿼리문을 전달하기 위해서 쿼리문을
        // 보냄
    })
    .then(function (connectionQuery) { // PoolGetConnection에서 커넥션과 쿼리문을 받
        // 음
        return ExecuteQuery(connectionQuery); // ExecuteQuery함수에 커넥션과 쿼리
        // 문을 보냄
    })
    .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
        console.log("This Solutions is : " + JSON.stringify(rows));
        return StoreComplete(res); // RegComplete에 res를 보냄. res.json을 실행하기
        // 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
    }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
        console.log("err 1 : "+err);
        return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection ); // 결과값이 어땠든
        // 커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
        res.json({success: false, msg: 'Failed to register user'});
    })
    });

    router.post('/GetProductDetail', (req, res, next) => {

        const productcode = req.body.productcode;

        CreateFindProductCodeQuery(productcode)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows[0]));
            return Complete(res, rows[0]); // RegComplete에 res를 보냄. res.json을 실행
            // 하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Get Product Detail'}); // 에러 캐치시
            // false반환
        })
        .then( function () {

```

```

        return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떻든
        커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/Product', (req, res, next) => {

    CreateProductFoundQuery()
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows);    // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) {    // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떻든
            커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
});

router.post('/Store', (req, res, next) => {

    CreateStoreFoundQuery()
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetStoreComplete(res, rows);    // RegComplete에 res를 보냄. res.json을
            실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) {    // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떻든

```

```

커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/FoundEnt', (req, res, next) => {

    const number = req.body.number;

    CreateFoundEntQuery(number)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows[0]));
            return GetStoreComplete(res, rows[0]); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to register user'}); // 에러 캐치시 false반환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection ); // 결과값이 어땠든
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
});

router.post('/FindProduct', (req, res, next) => {

    const store = req.body.store;
    console.log("This Solutions is : " + store);

    CreateFindProductQuery(store)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows); // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            console.log("Query Excute err : "+err);
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) { // 전체적으로 에러를 캐치한다

```

```

        console.log("Catch 1 err : "+err);
        res.json({success: false, msg: 'Failed to Find ALL Product'}); // 에러 캐치시 false
반환
    })
    .then( function () {
        return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떠한
커넥션은 반환되어야 한다
    })
    .catch(function (err) { //마지막으로 에러를 캐치
        console.log(err);
    })
});

router.post('/FindCategory', (req, res, next) => {

    const category = req.body.category;
    console.log("category is : " + category);

    CreateFindCategoryQuery(category)
        .then( query => {
            return PoolGetConnection(query);
        })
        .then(connectionQuery => {
            return ExecuteQuery(connectionQuery);
        })
        .then(function(rows) { // ExecuteQuery가 쿼리문을 사용한 결과값을 받음
            console.log("This Solutions is : " + JSON.stringify(rows));
            return GetProductComplete(res, rows);    // RegComplete에 res를 보냄. res.json
을 실행하기 위해서는 res값이 필요하기 때문에 res를 인자값으로 보냄
        }, function(err) { // ExecuteQuery가 쿼리문을 실행한 결과로 에러가 온 경우
            return Rollback(connection); // 쿼리문 실행 중 에러가 나면 롤백을 실행해야 함
        })
        .catch( function (err) {    // 전체적으로 에러를 캐치한다
            console.log("Catch 1 err : "+err);
            res.json({success: false, msg: 'Failed to Found Category'}); // 에러 캐치시 false반
환
        })
        .then( function () {
            return ReleaseConnection( connectionQuery.connection );    // 결과값이 어떠한
커넥션은 반환되어야 한다
        })
        .catch(function (err) { //마지막으로 에러를 캐치
            console.log(err);
        })
});

function CreateFindCategoryQuery(category) {
    return new Promise( function (resolve) {
        console.log('category : '+category);
        let statement = "SELECT * FROM product where category='"+category+"'";
        console.log("CreateFindCategoryQuery : "+statement);
        resolve(statement);
    });
}

function addMasterCertQuery(certinfo) {
    return new Promise( function (resolve) {
        let statement = "INSERT INTO cert_"+certinfo.user.id+" (masterCert, cert) VALUES ("
+ 1 + ", " + certinfo.cert + ")";

```

```

        console.log("addMasterCertQuery : "+statement);
        resolve(statement);
    });
}

function CreateProductFoundQuery() {
    return new Promise( function (resolve) {
        let statement = "SELECT * FROM product";
        console.log("ALLProductFoundQuery : "+statement);
        resolve(statement);
    });
}

function PoolGetConnection(query) {    //Pool에서 Connection을 가져오는 Promise 함수
    return new Promise( function (resolve, reject) {
        console.log("PoolGetConnection 1");
        pool.getConnection(function (err, connection) {
            if(connection){
                var connectionQuery = {
                    connection : connection,
                    query : query
                };
                console.log("PoolGetConnection 2");
                resolve(connectionQuery);
            } else {
                console.log("PoolGetConnection err : "+err);
                reject(err);
            }
        });
    });
}

function ExecuteQuery(ConQue) {    // Connection과 쿼리문을 받아와서 실행하는 Promise
함수
    return new Promise( function (resolve, reject) {
        ConQue.connection.query(ConQue.query, function(err, rows, fields) {
            if (!err) {
                console.log("ExecuteQuery : "+ JSON.stringify(rows));
                resolve(rows);
            } else {
                console.log("ExecuteQuery err : "+err);
                reject(err);
            }
        });
        ConQue.connection.release();
    });
}

function CompleteResult(res, rows) {
    return new Promise( function () {
        console.log('성공 : '+ JSON.stringify(rows));
        res.json({
            success: true,
            result : rows
        });
    });
}

function MasterCertComplete(res, certinfo) {
    return new Promise( function () {
        console.log('성공 : '+ JSON.stringify(certinfo));
    });
}

```



```

        res.json({
            success: true,
            Mcert : certinfo.cert,
            caCert : certinfo.caCert
        });
    });
}

function Complete(res) {    // 프론트 엔드에 Success : true값을 반환하는 Promise 함수
    return new Promise( function () {
        res.json({success: true, msg: 'Success'});
    });
}

function Rollback(connection) {
    return new Promise( function () {
        connection.rollback(function () {
            console.error('rollback error1');
        });
    });
}

function ReleaseConnection(connection) {
    return new Promise( function () {
        connection.release();
    });
}

var pool = mysql.createPool(config);

module.exports = router;

```

```

<database.js>
//Localhost Database
/*
module.exports = {
    host      : 'localhost',
    user       : 'dba',
    password   : 'jbis2019',
    port       : 3306,
    database   : 'coconut',
    secret: 'mysupersecret',
    connectionLimit : 100
}*/

//DB4Free Database
/*
module.exports = {
    host : 'db4free.net',
    user : 'coconut',
    password : 'jbis2019',
    port : 3306,
    database : 'coconut',
    secret : 'mysupersecret',
    connectionLimit : 100,
    timeout : 60 * 60 * 1000
}
*/
//Personal Database
module.exports = {

```

```

host : 'gkszm.com',
user : 'dba',
password : 'jbis2019',
port : 3307,
database : 'coconut',
secret : 'mysupersecret',
connectionLimit : 100,
timeout : 60 * 60 * 1000
}

```

```

<passport.js>
const JwtStrategy = require('passport-jwt').Strategy;
const ExtractJwt = require('passport-jwt').ExtractJwt;
const config = require("./database");
const mysql = require('mysql');

module.exports = function(passport){
  let opts = {};
  opts.jwtFromRequest = ExtractJwt.fromAuthHeaderWithScheme( 'jwt');
  opts.secretOrKey = config.secret;

  //console.log('secretOrKey : ' + opts.secretOrKey);

  passport.use(new JwtStrategy( opts, (jwt_payload, done) => {
    //console.log('jwt_payload : '+JSON.stringify(jwt_payload));
    let statement = "SELECT * FROM user WHERE id='" + jwt_payload.data.id + "'";
    var pool = mysql.createPool(config);

    pool.getConnection(function (err, connection) {
      if(!err){
        connection.query(statement, function(err, rows, fields) {
          if (err) {
            console.log('Error while performing Query.', err);
            throw err;
            return done(err, false);
          } else if (rows){
            //console.log('The solution is: ', JSON.stringify(rows[0]));
            return done(null, rows[0]);
          } else {
            return done(null, false);
          }
        });
        connection.release(); //쿼리가 성공하던 실패하던 커넥션을 반환해야 함
      }
    });

    /*
    pool.getConnection((connection, err) => {
      if (err) {
        console.log(err);
        throw err;
      }
      if (connection) {
        connection.query(statement, (err, rows, fields) => {
          if (err) {
            return done(err, false);
          }
          if (rows) {
            return done(null, rows[0]);
          } else {

```

```

        return done(null, false);
    }
    })
  }
  });
  */
  });
};

```

```

<caCert.js>
var forge = require('node-forge');
var fs = require('fs');
var pki = forge.pki;
// 1. CA 인증서 생성
// generate a keypair and create an X.509v3 certificate
var caKeys = pki.rsa.generateKeyPair(2048);
var caCert = pki.createCertificate();
// CA 개인키 파일 저장
console.log(pki.privateKeyToPem(caKeys.privateKey));
fs.writeFileSync("caPrivateKey.pem", pki.privateKeyToPem(caKeys.privateKey));
console.log('CA개인키 저장 - caPrivateKey.pem \n\n');
caCert.publicKey = caKeys.publicKey;
caCert.serialNumber = '01';
caCert.validity.notBefore = new Date();
caCert.validity.notAfter = new Date();
caCert.validity.notAfter.setFullYear(caCert.validity.notBefore.getFullYear() + 1);
var caAttrs = [{
  //name: 'commonName', // CN
  shortName: 'CN',
  value: 'Coconut'
}, {
  //name: 'countryName', // C
  shortName: 'C',
  value: 'KR'
}, {
  //name: 'stateOrProvinceName', // ST
  shortName: 'ST',
  value: 'Gyeonggi-do'
}, {
  //name: 'localityName', // L
  shortName: 'L',
  value: 'Goyang-si'
}, {
  //name: 'organizationName', // O
  shortName: 'O',
  value: 'Joongbu Univ.'
}, {
  //name: 'organizationalUnitName',
  shortName: 'OU',
  value: 'Dept. of Information Security'
}];
caCert.setSubject(caAttrs);
caCert.setIssuer(caAttrs);

caCert.setExtensions([
  {
    name: 'subjectAltName',
    altNames: [
      {
        type: 6, // URI
        value: 'http://coconutpay.herokuapp.com/'
      }
    ]
  }
]);

```

```

});

// self-sign certificate
caCert.sign(caKeys.privateKey);
console.log('CA 자체서명인증서 생성');
console.log(pki.certificateToPem(caCert));
var verified = caCert.verify(caCert);
console.log('CA인증서 생성 후 검증: ' + verified);
console.log();
// CA 인증서 저장
fs.writeFileSync("caCert.pem", pki.certificateToPem(caCert));
console.log('CA인증서 저장 - caCert.pem');

```

```

<App.vue>
<template>
  <div id="app" class="container">
    <form>
      <header>
        <Navbar> </Navbar>
      </header>

      <section>
        <router-view />
      </section>
    </form>
  </div>
</template>

<script>
import Navbar from "./components/Navbar";
export default {
  name: "app",
  components: {
    Navbar
  },
  created() {
    this.$store.commit("GET_TOKENS");
  }
};
</script>

<style>
#app {
  font-family: "Avenir", Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  /*text-align: center;*/
  position: relative;
  width: 100%;
}
body{
  margin-top:120px; /*메뉴바와 본문의 간격 설정*/
}
@media(max-width: 840px){
  body{
    margin-top:120px;
  }
}
</style>

```

```

<main.js>
import Vue from 'vue'
import VueRouter from 'vue-router'
import axios from 'axios'
import store from './store'
import VueQrcodeReader from 'vue-qrcode-reader'

Vue.use(VueQrcodeReader);

Vue.prototype.$axios = axios;
Vue.config.productionTip = false;
Vue.prototype.$EventBus = new Vue();

Vue.use(VueRouter);

import App from './App.vue'
import Home from './components/Home';
import Navbar from './components/Navbar';
import Login from './components/Login';
import ChoiceMemberType from './components/ChoiceMemberType';
import RegisterIndividual from './components/RegisterIndividual';
import RegisterEnterpriseBuyer from './components/RegisterEnterpriseBuyer';
import RegisterEnterpriseSeller from './components/RegisterEnterpriseSeller';
import MyPage from './components/MyPage';
import Pay from './components/Pay';
import Test from './components/Test';
import NotFound from './components/NotFound';
import ImageUpload from './components/ImageUpload';
import CreateStore from './components/CreateStore';
import AllProduct from './components/AllProduct';
import AllStore from './components/AllStore';
import Category from './components/Category';
import DetailProduct from './components/DetailProduct';
import CreateOrder from './components/CreateOrder';
import Cert from './components/Cert';
import Cart from './components/Cart';
import CertTest from './components/CertTest';
import AdditionalCert from './components/AdditionalCert';
import ChangePassword from './components/ChangePassword';
import MyCategory from './components/MyCategory';
import ChoicePaytype from './components/ChoicePayType';
import OrderView from './components/OrderView';
import PurchaseSuccess from './components/PurchaseSuccess';
import DetailOrder from './components/DetailOrder';
import DetailOrderSell from './components/DetailOrderSell';
import SalesHistory from './components/SalesHistory';
//import jusoPopup from './popup/jusoPopup.jsp';

const router = new VueRouter({
  mode: 'history',
  routes : [
    {path: '/', component : Home },
    {path: '/Login', component : Login },
    {path: '/Navbar', component : Navbar },
    {path: '/ChoiceMemberType', component : ChoiceMemberType},
    {path: '/RegisterIndividual', component : RegisterIndividual },
    {path: '/RegisterEnterpriseBuyer', component : RegisterEnterpriseBuyer },
    {path: '/RegisterEnterpriseSeller', component : RegisterEnterpriseSeller },
    {path: '/MyPage', component : MyPage },
  ]
})

```

```

    {path: '/Pay', component : Pay },
    {path: '/Test', component : Test },
    {path: '/ImageUpload', component : ImageUpload },
    {path: '/CreateStore', component : CreateStore },
    {path: '/AllProduct', component : AllProduct },
    {path: '/AllStore', component : AllStore },
    {path: '/Category', component : Category },
    {path: '/DetailProduct/:product', component : DetailProduct },
    {path: '/Cert', component : Cert },
    {path: '/Cart', component : Cart },
    {path: '/CertTest', component : CertTest },
    {path: '/AdditionalCert', component : AdditionalCert },
    {path: '/MyCategory', component : MyCategory },
    {path: '/ChangePassword', component : ChangePassword },
    {path: '/CreateOrder/:number/:product', component : CreateOrder },
    {path: '/ChoicePayType/:order', component : ChoicePaytype },
    {path: '/OrderView', component : OrderView },
    {path: '/PurchaseSuccess/:order', component : PurchaseSuccess },
    {path: '/DetailOrder/:order', component : DetailOrder },
    {path: '/DetailOrderSell/:order', component : DetailOrderSell },
    {path: '/SalesHistory', component : SalesHistory },
    {path: '*', component: NotFound }

    //{path: 'jusoPopup', component : jusoPopup }
  ]
});
new Vue({
  store,
  router,
  render: h => h(App),
}).$mount('#app')

```

```

<store.js>
import Vue from 'vue';
import Vuex from 'vuex';
import * as forge from 'node-forge';
var pki = forge.pki;
//import fs from 'browserify-fs';
import * as fs from 'browserify-fs';
//import fs from 'fs';
//var fs = require('fs');
//var fs = require('browserify-fs');
import axios from 'axios';
import main from './main';

Vue.use(Vuex);

//const resourceHost = 'http://localhost:3000';
const resourceHost = '';

export default new Vuex.Store({
  state: {
    sToken: null,
    pToken: null,
  },
  mutations: {
    GET_TOKENS : function(state, payload) {
      state.sToken = localStorage.getItem('sToken');
      state.pToken = localStorage.getItem('pToken');
    },
  },
});

```

```

    LOGIN : function (state, payload) {
        state.sToken = payload.data.stoken;
        state.pToken = payload.data.ptoken;
        localStorage.setItem('pToken', payload.data.ptoken);
        localStorage.setItem('sToken', payload.data.stoken);
    },
    LOGOUT : function (state) {
        state.sToken = null;
        state.pToken = null;
        localStorage.removeItem('sToken');
        localStorage.removeItem('pToken');
    }
},
actions: {
    LOGIN : function (context, payload) {
        return axios.post( resourceHost+'/users/authenticate', payload);
        //return axios.post( '/users/authenticate', payload);
    },
    LOGOUT : function (context) {
        context.commit('LOGOUT');
    },
    REGISTER : function (context, payload) {
        return axios.post( resourceHost+payload.path, payload.user);
        //return axios.post( payload.path, payload.user);
    },
    GetProfile : function (context) {
        let currTime = new Date().getTime();
        let pt = localStorage.getItem('pToken');
        let st = localStorage.getItem('sToken');

        var md = forge.md.sha256.create();
        md.update(currTime + st);
        let auth = md.digest().toHex();

        return axios.get(
            resourceHost+'/users/profile',
            //'/users/profile',
            { headers: {
                "Authorization" : pt,
                "Ctime" : currTime,
                "Auth" : auth,
                "Content-Type" : 'application/json'
            }
        });
    },
    PAY : function (context, payload) {
        return axios.post( resourceHost+'/Pay/procpay', payload);
        //return axios.post( '/Pay/procpay', payload);
    },
    NewProduct : function (context, payload) {
        return axios.post(resourceHost+'/stores/newStore', payload);
        //return axios.post('/stores/newStore', payload);
    },
    GetProduct : function (context) {
        return axios.post(resourceHost+'/stores/Product');
        //return axios.post('/stores/Product');
    },
    GetStore : function (context) {
        return axios.post(resourceHost+'/stores/Store');
        //return axios.post('/stores/Store');
    },
}

```

```

FoundEnt : function (context, payload) {
    return axios.post(resourceHost+'/stores/FoundEnt', payload);
    //return axios.post('/stores/FoundEnt', payload);
},
FindCategory : function (context, payload) {
    return axios.post(resourceHost+'/stores/FindCategory', payload);
    //return axios.post('/stores/FindCategory', payload);
},
MyGetProduct : function (context, payload) {
    return axios.post(resourceHost+'/stores/MyProduct', payload);
},
MyGetProduct2 : function (context, payload) {
    return axios.post(resourceHost+'/stores/MyProduct2', payload);
},
MyFindCategory : function (context, payload) {
    return axios.post(resourceHost+'/stores/MyFindCategory', payload);
},
ChangePass: function (context, payload) {
    return axios.post(resourceHost+'/users/ChangePass', payload);
},
FindUsername: function (context, payload) {
    return axios.post(resourceHost+'/users/FindUsername', payload);
},
GetOrder : function (context, payload) {
    let auth = localStorage.getItem("pToken");
    return axios.post(
        resourceHost+'/Pay/GetOrder',
        payload,
        {headers : {
            "Authorization" : auth
        }});
    //return axios.post( '/Pay/GetOrder', payload);
},
GetOrder_2 : function (context, payload) {
    let a = (payload.orderno).split('/');
    let order = {
        order_no: a[0],
        time : a[1]
    };
    return axios.post( resourceHost+'/Pay/GetOrder_2', order);
    //return axios.post( '/Pay/GetOrder', payload);
},
GetOrder_3 : function (context, payload) { //사용자의 주문 조회
    let number = {
        number : payload.number
    };
    return axios.post( resourceHost+'/Pay/GetOrder_3', number);
    //return axios.post( '/Pay/GetOrder', payload);
},
GetOrder_4 : function (context, payload) { //판매자의 판매상품 주문 조회
    return axios.post( resourceHost+'/Pay/GetOrder_4', payload);
    //return axios.post( '/Pay/GetOrder', payload);
},
GetProductOder : function (context, payload) {
    console.log('product payload'+payload);
    return axios.post( resourceHost+'/stores/GetProductOder', payload);
    //return axios.post( '/stores/GetProductDetail', payload);
},
GetCart : function (context, payload) {
    console.log(JSON.stringify(payload));
    return axios.post(resourceHost+'/stores/GetCart', payload);
}

```



```

},
GetProductDetail : function (context, payload) {
    return axios.post( resourceHost+'/stores/GetProductDetail', payload);
    //return axios.post( '/stores/GetProductDetail', payload);
},
GetProductDetail2 : function (context, payload) {
    return axios.post( resourceHost+'/stores/GetProductDetail2', payload);
    //return axios.post( '/stores/GetProductDetail', payload);
},
GetProductDetail3 : function (context, payload) {
    return axios.post( resourceHost+'/stores/GetProductDetail3', payload);
    //return axios.post( '/stores/GetProductDetail', payload);
},
GetProductDetail4 : function (context, payload) {
    return axios.post( resourceHost+'/stores/GetProductDetail4', payload);
    //return axios.post( '/stores/GetProductDetail', payload);
},
addBasket : function (context, payload) {
    return axios.post( resourceHost+'/users/addBasket', payload);
    //return axios.post( '/users/addBasket', payload);
},
newOrder : function (context, payload) {
    return axios.post( resourceHost+'/Pay/newOrder', payload);
    //return axios.post( '/Pay/newOrder', payload);
},
certRequest : function(context, payload) {

    var keypair = pki.rsa.generateKeyPair(2048);
    var publicKey = keypair.publicKey;
    var privateKey = keypair.privateKey;
    var privateKeyPem = pki.privateKeyToPem(privateKey);
    var publicKeyPem = pki.publicKeyToPem(publicKey);

    console.log('public key : '+JSON.stringify(publicKey));
    console.log('private Key : '+JSON.stringify(privateKey));
    console.log('public key PEM : '+JSON.stringify(publicKeyPem));
    console.log('private Key PEM : '+JSON.stringify(privateKeyPem));

    //2. 개인키 포맷변환 Asn1
    var rsaPrivateKey = pki.privateKeyToAsn1(privateKey);
    console.log('rsaPrivateKey : '+JSON.stringify(rsaPrivateKey));

    //3. 개인키 정보 생성
    // 개인키를 RSA ASN.1 오브젝트 형식의 정보???
    var privateKeyInfo = pki.wrapRsaPrivateKey(rsaPrivateKey);
    console.log('privateKeyInfo : '+JSON.stringify(privateKeyInfo));

    //4. 개인키 정보를 암호화(비밀번호 사용)
    var encryptedPrivateKeyInfo = pki.encryptPrivateKeyInfo(
        privateKeyInfo, payload.pa, {
            algorithm: 'aes256', // 'aes128', 'aes192', 'aes256', '3des'
        });
    console.log('encryptedPrivateKeyInfo : '+JSON.stringify(encryptedPrivateKeyInfo));

    //5. pem 형식으로 만들기 / pem을 fs 사용해서 파일로 저장
    var finalpem = pki.encryptedPrivateKeyToPem(encryptedPrivateKeyInfo);
    console.log('encryptedPrivateKeyInfo PEM : '+finalpem);

    /*
    fs.mkdir('/home', function() {
        fs.writeFile('/home/hello-world.txt', 'Hello world!\n', function() {

```

```

        fs.readFile('/home/hello-world.txt', 'utf-8', function(err, data) {
            console.log(data);
        });
    });
    var filename = './'+payload.user.id+'.pem.txt';
    fs.writeFile(filename, pki.encryptedPrivateKeyToPem(encryptedPrivateKeyInfo),
function (err) {
    if(err)console.log(err);
    else {
        console.log("key save success");
        fs.readFile(filename, 'utf-8', function(err, ma11) {
            if(err)console.log(err);
            else console.log( filename+' : '+ma11);
        });
    }
});
*/

localStorage.setItem(payload.user.id+'.pem', finalpem);

const req = {
    user : payload.user,
    publicKey: publicKeyPem
};

return axios.post( resourceHost+'/Cert/newCert', req);
//return axios.post( '/Cert/newCert', req);
},
CertValidate : function(context, payload) {
    console.log(payload.id+'.pem');
    var pem = localStorage.getItem(payload.id+'.pem');
    console.log('pem : '+pem);
    var encryptedPrivateKeyInfo = pki.encryptedPrivateKeyFromPem(pem);
    console.log('encryptedPrivateKeyInfo : '+JSON.stringify(encryptedPrivateKeyInfo));
    var privateKeyInfo = pki.decryptPrivateKeyInfo(
        encryptedPrivateKeyInfo, payload.pa);
    console.log('privateKeyInfo : '+JSON.stringify(privateKeyInfo));
    var privateKey = pki.privateKeyFromAsn1(privateKeyInfo);
    console.log('privateKey : '+JSON.stringify(privateKey));

    var currentTime = new Date().getTime();

    // sign data using RSASSA-PSS where PSS uses a SHA-1 hash, a SHA-1 based
    // masking function MGF1, and a 20 byte salt
    var md1 = forge.md.sha1.create();
    md1.update(payload.id, 'utf8');
    md1.update(currentTime, 'utf8');
    var pss1 = forge.pss.create({
        md: forge.md.sha1.create(),
        mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
        saltLength: 20
        // optionally pass 'prng' with a custom PRNG implementation
        // optional pass 'salt' with a forge.util.ByteBuffer w/custom salt
    });
    var signature1 = privateKey.sign(md1, pss1);
    var signatureHex1 = forge.util.bytesToHex(signature1);
    console.log('signature 1 : '+signature1);
    console.log('signature Hex 1 : '+signatureHex1);

    const certPem = localStorage.getItem(payload.id+'.cert');

```

```

const cert = pki.certificateFromPem(certPem);
const publicKey = cert.publicKey;
// verify RSASSA-PSS signature
var pss2 = forge.pss.create({
  md: forge.md.sha1.create(),
  mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
  saltLength: 20
  // optionally pass 'prng' with a custom PRNG implementation
});
var md2 = forge.md.sha1.create();
md2.update(payload.id, 'utf8');
md2.update(currentTime, 'utf8');
var verifySignature = publicKey.verify(md2.digest().getBytes(), signature1, pss2);
console.log('Signature Verify : '+verifySignature);

},
AddCertRequest : function(context, payload) {
  var keypair = pki.rsa.generateKeyPair(2048);
  var publicKey = keypair.publicKey;
  var privateKey = keypair.privateKey;
  var privateKeyPem = pki.privateKeyToPem(privateKey);
  var publicKeyPem = pki.publicKeyToPem(publicKey);

  console.log('Add public key : '+JSON.stringify(publicKey));
  console.log('Add private Key : '+JSON.stringify(privateKey));
  console.log('Add public key PEM : '+JSON.stringify(publicKeyPem));
  console.log('Add private Key PEM : '+JSON.stringify(privateKeyPem));

  var rsaPrivateKey = pki.privateKeyToAsn1(privateKey);
  console.log('Add rsaPrivateKey : '+JSON.stringify(rsaPrivateKey));

  var privateKeyInfo = pki.wrapRsaPrivateKey(rsaPrivateKey);
  console.log('Add privateKeyInfo : '+JSON.stringify(privateKeyInfo));

  var encryptedPrivateKeyInfo = pki.encryptPrivateKeyInfo(
    privateKeyInfo, payload.pa, {
      algorithm: 'aes256', // 'aes128', 'aes192', 'aes256', '3des'
    });
  console.log('Add encryptedPrivateKeyInfo : '+JSON.stringify(encryptedPrivateKeyInfo));

  var finalpem = pki.encryptedPrivateKeyToPem(encryptedPrivateKeyInfo);
  console.log('Add encryptedPrivateKeyInfo PEM : '+finalpem);

  localStorage.setItem(payload.user.id+'.pem', finalpem);

  const req = {
    user : payload.user,
    deviceId : payload.deviceId,
    publicKey: publicKeyPem
  };
  console.log("req : "+JSON.stringify(req));

  return axios.post( resourceHost+'/Cert/AddCert', req);
  //return axios.post( '/Cert/AddCert', req);
},
storeMCert : function(context, payload) {
  localStorage.setItem(payload.id+'.cert', payload.response.Mcert);
  localStorage.setItem('caCert', payload.response.caCert);
},
storeCert : function(context, payload) {

```

```

        localStorage.setItem('cert', payload.cert);
        localStorage.setItem('MCert', payload.MCert);
    },
    deletePem : function(context, payload) {
        localStorage.removeItem(payload.user.id+'.pem');
    },
    TradeRequest : function(context, payload) {
        console.log(payload.id+'.pem');
        var pem = localStorage.getItem(payload.id+'.pem');
        console.log('pem : '+pem);
        var encryptedPrivateKeyInfo = pki.encryptedPrivateKeyFromPem(pem);
        console.log('encryptedPrivateKeyInfo : '+JSON.stringify(encryptedPrivateKeyInfo));
        var privateKeyInfo = pki.decryptPrivateKeyInfo(
            encryptedPrivateKeyInfo, payload.pa);
        console.log('privateKeyInfo : '+JSON.stringify(privateKeyInfo));
        var privateKey = pki.privateKeyFromAsn1(privateKeyInfo);
        console.log('privateKey : '+JSON.stringify(privateKey));

        var currentTime = new Date().getTime();

        console.log('payload : '+JSON.stringify(payload));
        console.log('currentTime : '+currentTime);

        const sign = {
            id : payload.id,
            unum : payload.unum,
            order : payload.order,
            order_no : payload.order_no,
        };

        // sign data using RSASSA-PSS where PSS uses a SHA-1 hash, a SHA-1 based
        // masking function MGF1, and a 20 byte salt
        var md1 = forge.md.sha1.create();
        md1.update(sign, 'utf8');
        md1.update(currentTime, 'utf8');
        var pss1 = forge.pss.create({
            md: forge.md.sha1.create(),
            mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
            saltLength: 20
            // optionally pass 'prng' with a custom PRNG implementation
            // optionally pass 'salt' with a forge.util.ByteBuffer w/custom salt
        });
        var signature1 = privateKey.sign(md1, pss1);
        var signatureHex1 = forge.util.bytesToHex(signature1);
        console.log('signature 1 : '+signature1);
        console.log('signature Hex 1 : '+signatureHex1);
        //여기까지 서명 생성

        const certPem = localStorage.getItem(payload.id+'.cert');

        const Trade = {
            Request : {
                id : payload.id,
                unum : payload.unum,
                order : payload.order,
                order_no : payload.order_no
            },
            cert : certPem,
            currentT : currentTime,
            signature : signatureHex1
        };
    }

```

```

return axios.post( resourceHost+ '/Pay/Trade', Trade);
/*
//여기부터 서명 검증
const certPem = localStorage.getItem(payload.id+'.cert');
const cert = pki.certificateFromPem(certPem);
const publicKey = cert.publicKey;
// verify RSASSA-PSS signature
var pss2 = forge.pss.create({
  md: forge.md.sha1.create(),
  mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
  saltLength: 20
  // optionally pass 'prng' with a custom PRNG implementation
});
var md2 = forge.md.sha1.create();
md2.update(payload, 'utf8');
md2.update(currentTime, 'utf8');
var verifySignature = publicKey.verify(md2.digest().getBytes(), signature1, pss2);
console.log('Signature Verify : '+verifySignature);
*/
},
ReceiptRequest : function(context, payload) {
  console.log(payload.user.id+'.pem');
  var pem = localStorage.getItem(payload.user.id+'.pem');
  console.log('pem : '+pem);
  var encryptedPrivateKeyInfo = pki.encryptedPrivateKeyFromPem(pem);
  console.log('encryptedPrivateKeyInfo : '+JSON.stringify(encryptedPrivateKeyInfo));
  var privateKeyInfo = pki.decryptPrivateKeyInfo(
    encryptedPrivateKeyInfo, payload.pa);
  console.log('privateKeyInfo : '+JSON.stringify(privateKeyInfo));
  var privateKey = pki.privateKeyFromAsn1(privateKeyInfo);
  console.log('privateKey : '+JSON.stringify(privateKey));

  var currentTime = new Date().getTime();

  console.log('payload : '+JSON.stringify(payload));
  console.log('currentTime : '+currentTime);

  const sign = {
    user : payload.user,
    order : payload.order,
    product : payload.product,
    pprice : payload.pprice,
  };

  // sign data using RSASSA-PSS where PSS uses a SHA-1 hash, a SHA-1 based
  // masking function MGF1, and a 20 byte salt
  var md1 = forge.md.sha1.create();
  md1.update(sign, 'utf8');
  //md1.update(currentTime, 'utf8');
  var pss1 = forge.pss.create({
    md: forge.md.sha1.create(),
    mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
    saltLength: 20
    // optionally pass 'prng' with a custom PRNG implementation
    // optionalls pass 'salt' with a forge.util.ByteBuffer w/custom salt
  });
  var signature1 = privateKey.sign(md1, pss1);
  var signatureHex1 = forge.util.bytesToHex(signature1);
  console.log('signature 1 : '+signature1);
  console.log('signature Hex 1 : '+signatureHex1);

```

```

//여기까지 서명 생성

const certPem = localStorage.getItem(payload.user.id+'.cert');

const Receipt = {
  Request : {
    user : payload.user,
    order : payload.order,
    product : payload.product,
    pprice : payload.pprice,
  },
  cert : certPem,
  currentT : currentTime,
  signature : signatureHex1
};

return axios.post( resourceHost+'/Pay/Receipt', Receipt);
/*
//여기부터 서명 검증
const certPem = localStorage.getItem(payload.id+'.cert');
const cert = pki.certificateFromPem(certPem);
const publicKey = cert.publicKey;
// verify RSASSA-PSS signature
var pss2 = forge.pss.create({
  md: forge.md.sha1.create(),
  mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
  saltLength: 20
  // optionally pass 'prng' with a custom PRNG implementation
});
var md2 = forge.md.sha1.create();
md2.update(payload, 'utf8');
md2.update(currentTime, 'utf8');
var verifySignature = publicKey.verify(md2.digest().getBytes(), signature1, pss2);
console.log('Signature Verify : '+verifySignature);
*/
},
ReceiptValidateRequest : function(context, payload) {

  const sign = {
    user : payload.user,
    order : payload.order,
    product : payload.product,
    pprice : payload.pprice,
  };
  let res;

  /*
const Receipt = {
  Request : {
    user : payload.user,
    order : payload.order,
    product : payload.product,
    pprice : payload.pprice,
  },
  cert : certPem,
  currentT : currentTime,
  signature : signatureHex1
};
*/

  axios.post( resourceHost+'/Pay/ReceiptValidate', sign)

```

```

.then( response => {
    //console.log('response : ' +JSON.stringify(response));

    let certPem = response.data.Cert;
    //console.log('certPem : ' +JSON.stringify(certPem));

    //여기부터 서명 검증
    //const certPem = localStorage.getItem(payload.id+'.cert');

    for (var i=0; i<certPem.length; i++){

        let cert = pki.certificateFromPem(certPem[i].cert);
        console.log('cert : ' + JSON.stringify(cert));
        let publicKey = cert.publicKey;
        console.log('publicKey : ' + JSON.stringify(publicKey));
        // verify RSASSA-PSS signature

        console.log("중간02");

        var pss2 = forge.pss.create({
            md: forge.md.sha1.create(),
            mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),
            saltLength: 20
            // optionally pass 'prng' with a custom PRNG implementation
        });

        console.log("중간03");

        var md2 = forge.md.sha1.create();
        md2.update(sign, 'utf8');
        //md2.update(currentTime, 'utf8');

        console.log("중간04");

        var verifySignature = publicKey.verify(md2.digest().getBytes(),
payload.signature[0], pss2);

        console.log("중간05");

        console.log('Signature Verify : '+verifySignature);

        if (verifySignature == true) {
            res = {
                cert : certPem[i].cert,
                verifySignature : verifySignature
            };
            console.log("맞음");
            return res;
        } else {
            if (i == (certPem.length - 1)) {
                res = {
                    verifySignature : false
                };
                console.log("틀림");
                return res1;
            }
        }
    }
}

```

```

        }
    }
}

})
.catch( err => {
    res = {
        verifySignature : err
    };
    return res;
});
},
}
})

```

```

<AdditionalCert.vue>
<template>
  <div class="row" v-if="user.id">
    <div class="col-md-3"></div>
    <div class="col-md-6">
      <h2>추가 인증서 발급</h2><br>
      <input type="text" v-model="deviceId" aria-describedby="DeviceHelp"
class="form-control" placeholder="기기 이름">
      <h6><small id="DeviceHelp" class="form-text text-muted float-left">현재 사용
하는 기기의 별명이나 명칭을 입력하세요. 다른 기기와의 구분에 사용됩니다.</small></h6>
      <br>
      <input type="password" id="staticpass" v-model="p" class="form-control"
placeholder="추가 인증서 비밀번호">
      <p></p>
      <input type="password" v-model="c" class="form-control" placeholder="추가 인
증서 비밀번호 재입력">
      <p></p>
      <h5 v-if="m.co == true" style="color:green;" class="float-left">{{m.me}}</h5>
      <h5 v-if="m.co == false" style="color:red;" class="float-left">{{m.me}}</h5>
      <br>
      <button @click="AddCertSubmit" type="button" class="btn btn-primary">추가
인증서 발급 요청</button>
    </div>
    <div class="col-md-3"></div>
  </div>
</template>

<script>
  export default {
    name: "AdditionalCert",
    data(){
      return {
        deviceId : "",
        p : "",
        c : "",
        m : {
          me : "",
          co : Boolean
        },
        user : {}
      }
    },
  },

```



```

created() {
  this.$store.dispatch('GetProfile')
    .then( response => {
      //console.log('토큰검증 성공'+JSON.stringify(response.data.user));
      console.log('토큰검증 성공');
      this.user = response.data.user;
    }, err => {
      console.log('검증 실패' + err);
      this.$store.dispatch('LOGOUT');
      this.$router.replace({path : '/Login'});
    })
    .catch( err => {
      console.log('검증 에러' + err);
      this.$store.dispatch('LOGOUT');
      this.$router.replace({path : '/Login'});
    });
},
methods : {
  AddCertSubmit : function () {
    if ( (this.p == this.c ) && (this.m.co == true) ) {
      let certR = {
        pa : this.p,
        user : this.user,
        deviceId: this.deviceId
      };
      this.$store.dispatch('AddCertRequest', certR)
        .then( response => {
          if(response.data.success == true) {
            console.log('Cert Request 1 : '+JSON.stringify(response));
            var re = {
              response : response.data,
              id : this.user.id
            };
            this.$store.dispatch('storeMCert', re);
            alert('Cert Request Success');
            console.log('Cert Request Success');
          } else {
            this.$store.dispatch('deletePem', certR);
            console.log('Cert Request Failure');
            alert('Cert Request Failure');
          }
        })
        .catch( err => {
          console.log('Cert Request Err : '+ err);
        });
    }
    else {
      alert('인증서 비밀번호를 확인해주세요');
    }
  }
},
watch : {
  p : function (pa) {
    if ( this.c == "" ) {
    }
    else if ( pa == this.c ) {
      this.m.co = true;
      this.m.me = '일치합니다';
    } else {
      this.m.co = false;
      this.m.me = '일치하지 않습니다.';
    }
  }
}

```

```

    }
  },
  c : function (confirm) {
    if ( confirm == this.p ) {
      this.m.co = true;
      this.m.me = '일치합니다';
    } else {
      this.m.co = false;
      this.m.me = '일치하지 않습니다.';
    }
  }
}
}
</script>

<style scoped>

</style>

```

```

<AllProduct>
<template>
  <div class="AllProduct">
    <div class="row">
      <div class="col-md-2">
        <table class="table table-hover">
          <tbody>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('all')">
                모든 상품
              </th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('의류')">의류</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('식품')">식품</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('생활용품')">생활용품</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('가전디지털')">가전/디지털
</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('스포츠/레저')">스포츠/레저
</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('자동차용품')">자동차 용품
</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('도서/음반/DVD')">도서/음반
/DVD</th>
            </tr>
            <tr class="table-success">
              <th scope="row" v-on:click="onCategory('완구/취미')">완구/취미</th>
            </tr>
            <tr class="table-success">

```

```

        <th scope="row" v-on:click="onCategory('문구/오피스')">문구/오피스
    </th>
    </tr>
    <tr class="table-success">
        <th scope="row" v-on:click="onCategory('반려동물용품')">반려동물용품
    </th>
    </tr>
    <tr class="table-success">
        <th scope="row" v-on:click="onCategory('뷰티')">뷰티 </th>
    </tr>
    <tr class="table-success">
        <th scope="row" v-on:click="onCategory('출산/유아동')">출산/유아동
    </th>
    </tr>
    <tr class="table-success">
        <th scope="row" v-on:click="onCategory('주방용품')">주방용품 </th>
    </tr>
</tbody>
</table>
</div>
<div class="col-md-1"></div>
<div class="col-md-9">
    <Category v-bind:choice="choiceCategory"> </Category>
</div>
</div>
</div>
</template>

<script>
    import Category from "./Category";

    export default {
        name: "AllProduct",
        components : {
            Category
        },
        data () {
            return {
                choiceCategory : ""
            }
        },
        methods : {
            onCategory : function ( select ) {
                this.choiceCategory = select;
                console.log(this.choiceCategory);
            }
        }
    }
</script>

<style scoped>

</style>

```

```

<AllStore.vue>
<template>
    <div class="row">

```

```

<div class="col-md-3" v-for="store in stores">
  <div v-if="stores">

    <br><br>
    <div class="card border-primary mb-3" style="max-width: 20rem;">
      <div class="card-header">{{store.company }}네 상점</div>
      <div class="card-body">
        <!-- <h4 class="card-title">{{ store.seller }}</h4>-->
        
        <button @click="FindProductSubmit(store.number)" type="button"
class="btn btn-primary">상품보러가기</button>
      </div>
    </div>
  </div>
</div>
</template>

<script>
export default {
  name: "AllStore",
  data () {
    return {
      stores : {},
      sendText: ""
    }
  },
  methods : {
    FindProductSubmit : function (store) {

      this.$EventBus.$emit('select', store);
      console.log('선택 상점 전송');
      store = "";
      this.$router.push({path: '/AllProduct'});

    }
  },
  created() {
    this.$store.dispatch('GetStore')
    .then( response => {
      //alert('상점목록 성공 : '+JSON.stringify(response.data.store));
      console.log('상점목록 성공');
      this.stores = response.data.store;
    })
    .catch( err => {
      console.log('상점목록 실패' + err);
      //alert(err);
      this.$router.replace({path : '/Login'});
    })
  }
}
</script>

<style scoped>

</style>

```

```

<Cart.vue>
<template>
  <from>
    <div class="list-group" v-if="choice == 'cart'">
      <h2>장바구니</h2>
      <br>
      <table class="table">
        <thead>
          <tr class="table-active">
            <th scope="col">상품정보</th>
            <th scope="col">금액</th>
            <th scope="col">수량</th>
            <th scope="col">판매자</th>
            <th scope="col">선택</th>
          </tr>
        </thead>
        <tbody v-for="cart in carts">
          <tr>
            <td>
              <div class="media">
                <img v-bind:src='cart.thumbnail' class="align-self-start mr-3
widthSet heightSet" />
                <div class="media-body">
                  <h6 class="mt-0">
                    <router-link :to='"/DetailProduct/"+cart.productcode"
class="nav-link">{{cart.productname}}</router-link>
                  </h6>
                </div>
              </div>
            <td>{{{(cart.price*cart.quantity).toLocaleString()}}원}</td>
            <td>{{cart.quantity}}개</td>
            <td>{{cart.seller}}</td>
            <td>
              <div class="custom-control custom-checkbox">
                <input class="form-check-input" v-model="cart.check"
type="checkbox" value="1">
              </div>
            </td>
          </tr>
        </tbody>
      </table>
      <div>
        <button @click="CreateOrderSubmit" type="button" class="btn btn-block
btn-lg btn-primary">바로구매</button>
      </div>
      <!--
        <div v-for="cart in carts">
          <div class="custom-control custom-checkbox">
            <input class="form-check-input" v-model="cart.check"
type="checkbox" value="1" checked="">
          </div>
          <a class="list-group-item list-group-item-action flex-column
align-items-start">
            <div class="d-flex w-100 justify-content-between">
              <h5 class="mb-1">
                <router-link :to='"/DetailProduct/"+cart.productcode" class="nav-link">
                  {{cart.productname}}
                </router-link>
              </h5>
            </div>
          </a>
        </div>
      </div>
    </div>
  </from>
</template>

```

```

                <small class="text-muted">{{cart.quantity}}개 </small>
            </div>
            <p class="mb-1">{{cart.seller}}</p>
            <small class="text-muted">그냥{{cart.price}}원      합계
{{cart.quantity*cart.price}}</small>
        </a>
        <br>
        <p>{{cart.number}}</p>
        <div>
            <td width="200px">
                <button @click="CreateOrderSubmit" type="button"
class="btn btn-block btn-lg btn-primary">바로구매</button>
            </td>
        </div>
    </div>-->
</div>
</from>
</template>

<script>
export default {
  name: "Cart",
  data() {
    return {
      user : {},
      carts : {},
      temp : {},
      //lnk : 'http://localhost:3000/img/',
      lnk : "/img/"
    }
  },
  props: {
    choice : ""
  },
  created() {
    this.$store.dispatch('GetProfile')
      .then( response => {
        console.log('토큰검증 성공');
        this.user = response.data.user;});
  },
  watch : {
    choice : function (category) {
      if ( category == 'cart') {
        let UserNumber = {
          number : this.user.id
        };
        console.log(JSON.stringify(UserNumber));
        this.$store.dispatch('GetCart', UserNumber)
          .then( response => {
            console.log("가지고 온거 : "+JSON.stringify(response.data.store));
            this.temp = response.data.store;

            let pp = "";
            for (let i=0; i<response.data.store.length; i++) {
              if(i==0){
                pp = response.data.store[i].productcode;
              } else {
                pp = pp+'/'+response.data.store[i].productcode;
              }
            }
          })
      }
    }
  }
}

```

```

        console.log('pp: '+pp);
        let p = {
            productcode : pp
        };
        if ( (JSON.stringify(p).indexOf('/') !== -1) ) {
            return this.$store.dispatch('GetProductDetail3', p);
        } else {
            return this.$store.dispatch('GetProductDetail', p);
        }
    })
    .then( response => {
        console.log('res : '+JSON.stringify(response.data.result));
        this.imglnk(response.data.result);
        this.carts = this.temp;
        console.log('carts : '+JSON.stringify(this.carts));
    });
    }
    },
    methods : {
        imglnk : function (res) {
            for (let i=0; i<this.temp.length; i++) {
                for (let j=0; j<res.length; j++) {
                    if(this.temp[i].productname == res[j].productname) {
                        this.temp[i].thumbnail = this.lnk+res[j].thumbnail;
                    }
                }
                console.log(i +' '+j);
            }
        },
        CreateOrderSubmit : function () {
            //['1+1+1+1', '2+4+5+0']
            try {
                var arr = [];
                for(var io=0; io<100; io++) {
                    console.log('carts : '+JSON.stringify(this.carts[io]));
                    var a = String(this.carts[io].number);
                    var b = String(this.carts[io].productcode);
                    var c = String(this.carts[io].quantity);
                    var d = String(this.carts[io].check);

                    if((this.carts[io].check == true) || (this.carts[io].check == 1)) {
                        console.log('ab : '+JSON.stringify(arr));
                        var ab = String(b+'&'+c);
                        arr.push(ab);
                        console.log('ab2 : '+JSON.stringify(arr));
                    } else {
                        a = ",";
                        b = ",";
                        c = ",";
                        d = ",";
                    }
                }
            } catch (e) {
                console.log(e);
            } finally {
                this.$router.push({ path : '/CreateOrder/'+0+'/' +arr});
                console.log('arr : '+JSON.stringify(arr));
                //JSON.stringify(this.carts));
            }
        }
    }
}

```

```

    }

    }
  }
</script>

<style scoped>
  .widthSet {
    max-width: 60px;
  }
  .heightSet {
    max-height: 60px;
  }

  @media
  only screen
  and (max-width: 768px), (min-device-width: 768px)
  and (max-device-width: 1024px) {

    /* Force table to not be like tables anymore */
    table, thead, tbody, th, td, tr {
      display: block;
    }

    /* Hide table headers (but not display: none;, for accessibility) */
    thead tr {
      position: absolute;
      top: -9999px;
      left: -9999px;
    }

    tr {
      margin: 0 0 1rem 0;
    }

    tr:nth-child(odd) {
      background: #ccc;
    }

    td {
      /* Behave like a "row" */
      border: none;
      border-bottom: 1px solid #eee;
      position: relative;
      padding-left: 50%;
    }

    td:before {
      /* Now like a table header */
      position: absolute;
      /* Top/left values mimic padding */
      top: 0;
      left: 6px;
      width: 45%;
      padding-right: 10px;
      white-space: nowrap;
    }

    /*
    Label the data
    You could also use a data-* attribute and content for this. That way "bloats" the
  
```


HTML, this way means you need to keep HTML and CSS in sync. Lea Verou has a clever way to handle with text-shadow.

```

    */
    td:nth-of-type(1):before { content: "상품정보"; }
    td:nth-of-type(2):before { content: "금액"; }
    td:nth-of-type(3):before { content: "수량"; }
    td:nth-of-type(4):before { content: "판매자"; }
    td:nth-of-type(5):before { content: "선택"; }
  }
</style>

```

```

<Category.vue>
<template>
  <div class="list-group">
    <div v-for="product in Products">
      <a class="list-group-item" list-grou+55p-item-action flex-column
align-items-start">
        <div class="d-flex w-100 justify-content-between row">
          <div class="col-md-3">
            
          </div>
          <div class="col-md-4">
            <h5><router-link :to=""/DetailProduct/' + product.productcode"
class="nav-link">{{product.productname}}</router-link></h5>
            <!--<h6>{{product.description}}</h6>-->
            <p></p>
            <small class="text-muted col-md-1">{{product.category}}</small>
          </div>
          <div class="col-md-3">
            <h5><strong class="text-black">{{(product.price).toLocaleString()}}원
</strong></h5>
          </div>
          <div class="col-md-2">
            </div>
          <!--
          <div class="col-md-3">
            </div>
          </div>
          <div>
            
          </div>
          <div class="row">
            <div class="col-md-9">
              <h5><router-link :to=""/DetailProduct/' + product.productcode"
class="nav-link">{{product.name}}</router-link></h5>
            </div>
            <div class="col-md-3">
              <h5><strong class="text-black">{{product.price}}원
</strong></h5>
            </div>
            <div class="col-md-1"></div>
            <div class="col-md-8">
              <p>{{product.description}}</p>
            </div>
          </div>
        </div>
      </a>
    </div>
  </div>
</template>

```

```

        </div>
        <div class="col-md-3">
            <small class="text-muted">{{product.category}}</small>
        </div>
    </div>
    -->
</div>
</a>
<br>
</div>
</div>
</template>
<script>
export default {
  name: "Category",
  props: {
    choice : ""
  },
  data () {
    return {
      Products : [],
      //lnk : "http://localhost:3000/img/"
      lnk : "/img/"
    }
  },
  methods : {
    imglnk : function () {
      for (var i=0; i<(this.Products.length); i++) {
        this.Products[i].thumbnail = this.lnk+this.Products[i].thumbnail;
      }
    }
  },
  created() {
    this.choice = 'all';
  },
  computed : {
    /*
    imglnk2 : function () {
      for (var i=0; i<(this.Products.length); i++) {
        return this.lnk+this.Products[i].thumbnail;
      }
    }*/
  },
  watch : {
    choice : function (category) {
      if ( category == 'all') {
        this.$store.dispatch('GetProduct')
          .then( response => {
            //alert('카테고리 결과 2 : '+JSON.stringify(response));
            this.Products = response.data.Product;
            this.imglnk();
            console.log('카테고리 성공 1 : '+JSON.stringify(this.Products));
            console.log('카테고리 성공 1');
          })
          .catch( err => {
            console.log('카테고리 실패 1 : ' + err);
            //alert(err);
          })
      } else {
        let selectCategory = {

```

```

        category : category
    };
    this.$store.dispatch('FindCategory', selectCategory)
    .then( response => {
        //alert('카테고리 결과 3 : '+JSON.stringify(response));
        this.Products = response.data.Product;
        console.log('카테고리 성공 3 : '+JSON.stringify(this.Products));
    })
    .catch( err => {
        console.log('카테고리 실패 3 : ' + err);
        //alert(err);
    })
    }
    }
}
}
</script>

<style scoped>
    .widthSet {
        max-width: 150px;
    }
    .heightSet {
        max-height: 150px;
    }
    /*
    img.border-shadow{
        border:0px solid #888888;
        box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    }*/
    .imageBox{
        max-width: 100px;
        max-height: 100px;
        border:0px solid #888888;
        box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    }
</style>

```

```

<Cert.vue>
<template>
    <div class="row" v-if="user.id">
        <div class="col-md-1"> </div>
        <div class="col-md-9">
            <h2>인증서 발급</h2><br>
            <input type="password" v-model="p" class="form-control" placeholder="비밀번호">
            <p></p>
            <input type="password" v-model="c" class="form-control" placeholder="비밀번호 확인">
            <p></p>
            <h5 v-if="m.co == true" style="color:green;" class="float-left">{{m.me}}</h5>
            <h5 v-if="m.co == false" style="color:red;" class="float-left">{{m.me}}</h5>
            <br><br>
            <button @click="newCertSubmit" type="button" class="btn btn-primary">
인증서 발급</button>
        </div>
    <div class="col-md-2"> </div>

```

```

    </div>
</template>

<script>
export default {
  name: "Cert",
  data(){
    return {
      p : "",
      c : "",
      m : {
        me : "",
        co : Boolean
      },
      user : {}
    }
  },
  created() {
    this.$store.dispatch('GetProfile')
    .then( response => {
      //console.log('토큰검증 성공'+JSON.stringify(response.data.user));
      console.log('토큰검증 성공');
      this.user = response.data.user;
    }, err => {
      console.log('검증 실패' + err);
      this.$store.dispatch('LOGOUT');
      this.$router.replace({path : '/Login'});
    })
    .catch( err => {
      console.log('검증 에러' + err);
      this.$store.dispatch('LOGOUT');
      this.$router.replace({path : '/Login'});
    });
  },
  methods : {
    newCertSubmit : function () {
      if ( (this.p == this.c ) && (this.m.co == true) ) {
        let certR = {
          pa : this.p,
          user : this.user
        };
        this.link = './'+this.user.id+'pem.txt';
        this.$store.dispatch('certRequest', certR)
        .then( response => {
          if(response.data.success == true) {
            console.log('Cert Request 1 : '+JSON.stringify(response));
            var re = {
              response : response.data,
              id : this.user.id
            };
            this.$store.dispatch('storeMCert', re);
            alert('Cert Request Success');
            console.log('Cert Request Success');
          } else {
            this.$store.dispatch('deletePem', certR);
            console.log('Cert Request Failure');
            alert('Cert Request Failure');
          }
        })
        .catch( err => {
          console.log('Cert Request Err : '+ err);
        });
      }
    }
  }
}

```

```

        }
        else {
            alert('인증서 비밀번호를 확인해주세요');
        }
    }
},
watch : {
    p : function (pa) {
        if ( this.c == " " ) {

        }
        else if ( pa == this.c ) {
            this.m.co = true;
            this.m.me = '일치합니다';
        } else {
            this.m.co = false;
            this.m.me = '일치하지 않습니다.';
        }
    },
    c : function (confirm) {
        if ( confirm == this.p ) {
            this.m.co = true;
            this.m.me = '일치합니다';
        } else {
            this.m.co = false;
            this.m.me = '일치하지 않습니다.';
        }
    }
}
}
}
</script>
<style scoped>
</style>

```

```

<ChangePassword.vue>
<template>
  <from>
    <div class="row" v-if="choice == 'change'">

      <label for="password" class="col-sm-2 col-form-label">비밀번호</label>
      <div class="col-sm-10">
        <input type="password" v-model="Password.password1"
class="form-control" placeholder="Password">
      </div>
      <label for="password" class="col-sm-2 col-form-label">비밀번호 확인</label>
      <div class="col-sm-10">
        <input type="password" v-model="Password.password2"
class="form-control" placeholder="Password">
      </div>

      <button @click="onchangeSubmit" type="button" class="btn btn-primary">변경
하기</button>
    </div>

  </from>

```

```

</template>

<script>
  export default {
    name: "ChangePassword",
    data(){
      return {
        user : {},
        Password : {
          password1: "",
          password2: ""
        }
      }
    },
    props: {
      choice : ""
    },
    created() {
      this.$store.dispatch('GetProfile')
        .then( response => {
          console.log('토큰검증 성공');
          this.user = response.data.user;})
    },
    methods : {
      onChangeSubmit() {
        const Password = this.Password;
        console.log('user:' + JSON.stringify(this.user));
        if (Password.password1 !== Password.password2) {
          console.log('빠꾸꺼다름:' + Password.password1, + Password.password2);
          return alert('비밀번호가 같지 않다');
        } else {
          console.log('빠꾸꺼같음:' + Password.password2);
          let ChangePassword = {
            Password: this.Password.password2,
            number: this.user.number
          };

          this.$store.dispatch('ChangePass', ChangePassword)
            .then(res => {
              console.log('변경 성공 : ' + JSON.stringify(res));
              alert('변경 성공');
            })
            .catch(err => {
              console.log("Login Error! : ", err);
              return alert('변경 실패' + err);
            });
        }
      }
    },
    watch : {
      choice : function (category) {
        if ( category === 'change') {
          console.log('왔다 : '+category);
        }
        else {
          console.log('안왔다: ');
        }
      }
    }
  }

```

```

    }
  }

```

```

</script>

```

```

<style scoped>

```

```

</style>

```

```

<ChoiceMemberType.vue>
<template>
  <div>
    <div> <h2> <strong>회원가입</strong> </h2> <br> </div>
    <div class="row">
      <div class="col-md-6">
        <div class="card mb-3" style="max-width: 30rem;">
          <div class="card-header text-white bg-danger"> <strong>개인 회원 가
          입</strong> </div>
          <div class="list-group">
            <a class="list-group-item list-group-item-action flex-column
            align-items-start">
              <div class="float-left">
                <h5 class="card-text">개인 구매회원</h5>
                <h6 class="card-subtitle text-muted">만 14세 이상</h6>
              </div>
              <router-link to="/RegisterIndivisual" class="nav-link">
                <button type="submit" class="btn btn-danger
                float-right">회원가입</button>
              </router-link>
            </a>
          </div>
        </div>
      </div>
      <div class="col-md-6">
        <div class="card mb-3" style="max-width: 40rem;">
          <div class="card-header text-white bg-info"> <strong>사업자 회원 가입
          </strong> </div>
          <div class="list-group">
            <a class="list-group-item list-group-item-action flex-column
            align-items-start">
              <div class="float-left">
                <h5 class="card-text">사업자 구매회원</h5>
              </div>
              <router-link to="/RegisterEnterpriseBuyer" class="nav-link">
                <button type="submit" class="btn btn-info float-right">회
                원가입</button>
              </router-link>
            </a>
            <a class="list-group-item list-group-item-action flex-column
            align-items-start">
              <div class="float-left">
                <h5 class="card-text">사업자 판매회원</h5>
              </div>
              <router-link to="/RegisterEnterpriseSeller" class="nav-link">
                <button type="submit" class="btn btn-info float-right">회
                원가입</button>
              </router-link>
            </a>
          </div>
        </div>
      </div>
    </div>
  </div>

```

```

        </div>
      </div>
    </div>
  </template>

  <script>
    export default {
      name: "ChoiceMemberType"
    }
  </script>

  <style scoped>

</style>

```

```

<ChoicePayType.vue>
<template>
  <div class="row" v-if="allow">
    <div class="col-md-2"></div>
    <div class="col-md-8">
      <h2>결제</h2>
      <hr noshade/>
      <table class="table">
        <thead>
          <tr class="table-active">
            <th scope="col">상품 정보</th>
            <th scope="col">금액</th>
            <th scope="col">수량</th>
            <th scope="col">판매자</th>
          </tr>
        </thead>
        <tbody v-for="Pro in Products">
          <tr>
            <td>
              <div class="media">
                
                <div class="media-body">
                  <h5 class="mt-0">{{Pro.productname}}</h5>
                  <h6 class="text-muted">
                    {{Pro.description}}<br>
                    {{Pro.category}}
                  </h6>
                </div>
              </div>
            </td>
            <td><h5>{{(Pro.oquantity*Pro.price).toLocaleString()}}원</h5></td>
            <td><h5>{{Pro.oquantity}}개</h5></td>
            <td><h5>{{Pro.seller}}</h5></td>
          </tr>
        </tbody>
      </table>
      <hr noshade/>
      <div class="card">
        <div class="card-body">
          <div class="row">
            <h6 class="col-md-3"></h6>
            <h6 class="col-md-3">총 주문 상품수</h6>
            <h6 class="col-md-3">{{kind[0]}}종 {{kind[1]}}개</h6>
          </div>
        </div>
      </div>
    </div>
  </template>

```



```

        <h6 class="col-md-3"> </h6>
      </div>
      <hr class="my-4">
      <div class="row">
        <h6 class="col-md-3"> </h6>
        <h6 class="col-md-3"> 총 결제 예상 금액 </h6>
        <h5
          style="color: crimson"
class="col-md-4"> {{allprice.toLocaleString()}}원 </h5>
        <h6 class="col-md-2"> </h6>
      </div>
    </div>
  </div>
  <br>
  <div class="row">
    <div class="col-md-6">
      <button @click="nomalChoice" type="button" class="btn btn-lg
btn-primary col-md-12"> 바로 결제 </button>
      <div v-if="choiceType==true">
        <hr noshade/>
        <div class="alert alert-warning" role="alert">
          <h3 class="page-header"> 인증서 비밀번호 입력 </h3>
          <div class="form-group">
            <label> Password </label>
            <input
              type="password"
              class="form-control"
              v-model="Cpass" name="password">
            </div>
            <button @click="Trade" type="button" class="btn btn-primary
align-self-center"> 결제 </button>
          </div>
        </div>
      </div>
      <br>
      <div class="col-md-6">
        <button @click="qrChoice" type="button" class="btn btn-lg
btn-success col-md-12"> QR코드 결제 </button>
        <div v-if="choiceType==false">
          <hr noshade/>
          <div class="alert alert-warning" role="alert">
            <qrcode-vue
              :value="value"
              :size="size"
              level="H"> </qrcode-vue>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
import QrcodeVue from 'qrcode.vue';
export default {
  name: "ChoicePayType",
  data () {
    return {
      value : "",
      time : null,
      Cpass : "",
      size : 300,
      kind : [ 0 ],
      allprice : 0,
      choiceType : null,

```

```

        allow : false,
        user : {},
        order : {},
        pcode : "",
        pquan : [],
        seller : "",
        pnum : {
            orderno : this.$route.params.order
        },
        Products : [],
        //lnk : 'http://localhost:3000/img/',
        lnk : "/img/"
    }
},
methods : {
    Trade : function () {
        let certR = {
            pa : this.Cpass,
            id : this.user.id,
            unum : this.user.number,
            order : this.order,
            order_no : this.$route.params.order
        };
        this.$store.dispatch('TradeRequest', certR)
        .then( (response) => {
            console.log('결제 : '+JSON.stringify(response));
            alert('결제 완료');
            console.log('TradeRequest Success : '+JSON.stringify(response));
            var p = response.data.order;
            var p1 = p.split('/');
            this.$router.replace({ path : '/PurchaseSuccess/'+p1[0] });
        }).catch( err => {
            console.log('TradeRequest Err : '+ err);
        });
    },
    quantityAppend : function(Prod) {
        for (let i=0; i<Prod.length; i++) {
            for(let j=0; j<Prod.length; j++) {
                console.log('prod '+i+' : '+Prod[i].productcode);
                console.log('pquan '+j+' : '+ this.pquan[j][0]);
                if(Prod[i].productcode == this.pquan[j][0]) {
                    Prod[i].oquantity = this.pquan[j][1];
                    console.log('일치');
                }
            }
        }
        this.Products = Prod;
    },
    imglnk : function () {
        for (var i=0; i<(this.Products.length); i++) {
            this.Products[i].thumbnail = this.lnk+this.Products[i].thumbnail;
        }
    },
    nomalChoice : function () {
        this.choiceType = true;
    },
    qrChoice : function () {
        /*
        var v = {
            no : this.order.order_no,
            t : new Date().getTime()
        }
        */
    }
}

```

```

    };
    */
    //this.value = JSON.stringify(v);
    //console.log('value : '+JSON.stringify(v));
    this.time = this.order.order_no+'/'+new Date().getTime();
    console.log(this.time);
    this.value = this.time;
    this.choiceType = false;
  }
},
created() {
  this.$store.dispatch('GetProfile')
    .then( response => {
      console.log('토큰검증 성공');
      this.user = response.data.user;
      return this.$store.dispatch('GetOrder', this.pnum);
    })
    .then( response => {
      console.log('get order : '+JSON.stringify(response.data));
      this.order = response.data.order[0];
      var p = response.data.order[0].product;
      var p1 = p.split(',');
      var p2 = new Array;
      var p3 = new Array;
      for (var i=0; i<p1.length; i++) {
        p2.push(p1[i].split('/'));
        p3.push(p2[i][1]);
      }
      console.log('p2 : '+JSON.stringify(p2));
      console.log('p3 : '+JSON.stringify(p3));
      this.pquan = p2;
      var pcode2 = {
        productcode : p
      };
      console.log("pcode2 : "+JSON.stringify(pcode2));
      return this.$store.dispatch('GetProductDetail2', pcode2);
    })
    .then( response => {
      console.log('product detail : '+JSON.stringify(response.data));
      let pp = response.data.result;
      for (let i=0; i<pp.length; i++) {
        for(let j=0; j<pp.length; j++) {
          this.pquan[i][j] *= 1;
          console.log('pp '+i+' : '+pp[i].productcode);
          console.log('pquan '+j+' : '+ this.pquan[j][0]);
          if(pp[i].productcode == this.pquan[j][0]) {
            pp[i].oquantity = this.pquan[j][1];
            console.log('일치');
          } else {
            console.log('불일치');
          }
        }
      }
      this.Products = pp;
      this.seller = response.data.result[0].seller;
      this.imglnk();
      //this.quantityAppend(response.data.result);
      console.log('product detail2 : '+JSON.stringify(this.Products));
      if (this.user.number == this.order.orderer) {
        this.allow = true;
      } else {

```

```

        alert('잘못된 요청입니다.');
```

```

        this.$router.replace({ path : '/' });
    }
})
    .finally( () => {
        this.kind[0] = this.Products.length;
        this.kind[1] = 0;
        for (let i=0; i<this.Products.length; i++){
            this.allprice = this.allprice + this.Products[i].oquantity *
this.Products[i].price;
            this.Products[i].oquantity *= 1; //스트링을 정수로 형변환
            this.kind[1] = this.kind[1] + this.Products[i].oquantity;
        }
    })
    .catch( err => {
        alert('잘못된 요청입니다.');
```

```

        this.$router.replace({ path : '/' });
    });
},
computed : {
},
watch : {
},
components: {
    QrcodeVue
}
}
</script>

<style scoped>
    .widthSet {
        max-width: 80px;
    }
    .heightSet {
        max-height: 80px;
    }
/*
    Max width before this PARTICULAR table gets nasty. This query will take effect for
    any screen smaller than 760px and also iPads specifically.
    */
    @media
    only screen
    and (max-width: 768px), (min-device-width: 768px)
    and (max-device-width: 1024px) {
        /* Force table to not be like tables anymore */
        table, thead, tbody, th, td, tr {
            display: block;
        }
        /* Hide table headers (but not display: none;, for accessibility) */
        thead tr {
            position: absolute;
            top: -9999px;
            left: -9999px;
        }
        tr {
            margin: 0 0 1rem 0;
        }
        tr:nth-child(odd) {
            background: #ccc;
        }
        td {

```

```

/* Behave like a "row" */
border: none;
border-bottom: 1px solid #eee;
position: relative;
padding-left: 50%;
}
td:before {
/* Now like a table header */
position: absolute;
/* Top/left values mimic padding */
top: 0;
left: 6px;
width: 45%;
padding-right: 10px;
white-space: nowrap;
}
/*

```

Label the data

You could also use a data-* attribute and content for this. That way "bloats" the HTML, this way means you need to keep HTML and CSS in sync. Lea Verou has a clever way to handle with text-shadow.

```

/*
td:nth-of-type(1):before { content: "상품정보"; }
td:nth-of-type(2):before { content: "금액"; }
td:nth-of-type(3):before { content: "수량"; }
td:nth-of-type(4):before { content: "판매자"; }
}
</style>

```

```

<CreateOrder.vue>
<template>
  <div>
    <div class="row">
      <div class="col-md-3"> </div>
      <div class="col-md-6">
        <h2>{{user.name}}님의 주문서</h2>
        <hr noshade/>
        <br>
        <h5 class="float-left">1. 주문상품 확인</h5>
        <table class="table">
          <thead>
            <tr class="table-active">
              <th scope="col">상품정보</th>
              <th scope="col">금액</th>
              <th scope="col">수량</th>
              <th scope="col">판매자</th>
            </tr>
          </thead>
          <tbody v-for="Pro in Product">
            <tr>
              <td>{{Pro.productname}}</td>
              <td>{{(Pro.quantity*Pro.price).toLocaleString()}}원</td>
              <td>{{Pro.quantity}}개</td>
              <td>{{Pro.seller}}</td>
            </tr>
          </tbody>
        </table>
        <br>
        <h5 class="float-left">2. 배송지 정보 입력</h5>
        <table class="table">

```

```

        <thead>
        <tr>
            <th class="table-active" scope="row"> <h6>고객입력 상세주소
</h6> </th>
            <th scope="col">
                <input type="text" v-model="addrDetail" id="addrDetail"
class="form-control" name="addrDetail" placeholder="상세 주소">
            </th>
        </tr>
        <tr>
            <th class="table-active" scope="row"> <h6>우편번호</h6> </th>
            <th scope="col">
                <input type="text" class="form-control" v-model="zipNo"
id="zipNo" name="zipNo" placeholder="우편번호">
            </th>
        </tr>
        <tr>
            <th class="table-active" scope="row"> <h6>연락처</h6> </th>
            <th scope="col">
                <input type="text" class="form-control" v-model="orderTel"
placeholder="01012341234">
            </th>
        </tr>
        </thead>
    </table>
    <br> <br>
    <button @click="newOrderSubmit" type="button" class="btn btn-primary
btn-lg col-md-12">주문하기</button>
</div>
<div class="col-md-3"> </div>
</div>
</div>
<!--
<from>
    <div v-for="(Pro, i) in Product">
        <li>{{Pro.seller}}</li>
        <li>{{Pro.productcode}}</li>
        <li>{{Pro.name}}</li>
        <li>{{Pro.price}}</li>
        <li>{{Pro.category}}</li>
        <li>{{Products.product[i].quantity}}</li>
        <br>
    </div>
    <div>
        <li>{{Product}}</li>
    </div>
</from>
-->
</template>

<script>
export default {
    name: "CreateOrder",
    data () {
        return {
            Product : {},
            Products : {},
            quantitys : {},
            aArray : [],
            bArray : [],
            roadFullAddr : "",

```

```

        roadAddrPart1 : "",
        roadAddrPart2 : "",
        addrDetail : "",
        zipNo : "",
        unum : Number,
        orderTel : "",
        number : 1,
        user : {},
        numbers : {},
    }
},
created() {
    this.$store.dispatch('GetProfile')
    .then( response => {
        console.log('토큰검증 성공'+JSON.stringify(response.data.user));
        this.unum = response.data.user.number;
        this.user = response.data.user;
        var a = this.$route.params.product;
        console.log("이건뭐지"+JSON.stringify(this.$route.params.product));
        var number = this.$route.params.number;
        console.log("number : "+number);
        if ( number == '1') {//바로구매
            this.aArray = a.split('&');
            this.Products = {
                productcode : this.aArray[0],
                quantity : this.aArray[1]
            };
            this.quantities = this.aArray[1];
            return this.$store.dispatch('GetProductDetail', this.Products);
        } else if ( number == '0') {//장바구니
            this.bArray = a.split(',');
            console.log("안쪽에1"+JSON.stringify(this.bArray));
            this.aArray = this.bArray.toString().split('&');
            console.log("안쪽에2"+JSON.stringify(this.aArray));
            this.bArray = this.aArray.toString().split(',');
            console.log("안쪽에3"+JSON.stringify(this.bArray));
            var cArray = [];
            for (var i=0; i<(this.bArray.length); i++) {
                if ( ((i+2) % 2) == 0) {
                    cArray.push(
                        {
                            productcode : this.bArray[i],
                            quantity : this.bArray[i+1]
                        }
                    );
                }
            }
            this.Products = {
                product : cArray,
                id : this.user.id
            };
            console.log('product : '+JSON.stringify(this.Products));
        }
        return this.$store.dispatch('GetProductOrder', this.Products);
        this.numbers = this.$route.params.number;
    }, err => {
        console.log('검증 실패' + err);
        this.$store.dispatch('LOGOUT');
        this.$router.replace({path : '/Login'});
    })
    .then( res => {

```

```

        console.log('성공');
        console.log('가지고 온 상품들 : '+JSON.stringify(res));
        var a = res.data.result;
        console.log('number'+ this.$route.params.number);
        if ( this.$route.params.number == '1') {
            a[0].quantity = this.quantities
        }else{
            console.log('data : ' +JSON.stringify(JSON.stringify(res.data)));
        }
        this.Product = a;
        console.log('data : ' +JSON.stringify(JSON.stringify(res.data)));
        console.log('result : ' +JSON.stringify(JSON.stringify(res.data.result)));
        console.log('Product : ' +JSON.stringify(JSON.stringify(this.Product)));
    })
    .catch( err => {
        console.log('검증 실패' + err);
        this.$store.dispatch('LOGOUT');
        this.$router.replace({path : '/Login'});
    });
},
methods : {
    newOrderSubmit : function () {
        if ( this.addrDetail == " || this.orderTel == " ) {
            alert('주소와 전화번호를 입력하세요');
            console.log('validate fail');
        } else {
            var allproduct = [];
            for (var i=0; i<(this.Product.length); i++) {
                allproduct.push(
                    {
                        product:      this.Product[i].productcode      +      '/'      +
this.Product[i].quantity,
                        price: this.Product[i].price * this.Product[i].quantity,
                    }
                );
            }
            console.log(JSON.stringify(" number :"+this.$route.params.number));
            console.log(JSON.stringify(" length :"+this.Product.length));
            console.log(JSON.stringify(" code :"+this.Product[0].productcode));
            console.log(JSON.stringify(" price :"+this.Product[0].price));
            console.log(JSON.stringify(" quantity :"+this.Product[0].quantity));
            console.log(JSON.stringify(" allproduct :"+allproduct[0].product));
            console.log(JSON.stringify(" quantity :"+this.quantities));
            let newOrder = {
                product : allproduct,
                orderer: this.unum,
                delivery_address: this.addrDetail,
                delivery_tel: this.orderTel
            };
            console.log("newOrder : "+JSON.stringify(newOrder));
            this.$store.dispatch('newOrder', newOrder)
                .then( response => {
                    if(response.data.success == true) {
                        //alert('Order Success');
                        console.log('Order Success : '+ JSON.stringify(response));
                        this.$router.push({
                            path
                        } :
                        '/ChoicePayType/'+response.data.order.insertId);
                    } else {
                        alert('Order Failure');
                        console.log('Order Failure : '+ JSON.stringify(response))
                    }
                })
        }
    }
}

```



```

    }).catch( err => {
      console.log('Order Err : '+ err);
    });
  }
}
}
}
}
</script>

<style scoped>
</style>

```

```

<CreateStore.vue>
<template>
  <div v-if="choice == 'Create'">
    <h2>상품 등록</h2>
    <br>
    <div class="form-group row">
      <label for="name" class="col-md-2 col-form-label">상품명</label>
      <div class="col-md-10">
        <input type="text" id="name" v-model="newStore.name"
class="form-control" >
      </div>
      <br><br><br>
      <label for="price" class="col-md-2 col-form-label">가격</label>
      <div class="col-md-10">
        <input type="text" id="price" v-model="newStore.price"
class="form-control" >
      </div>
      <br><br><br>
      <label for="quantity" class="col-md-2 col-form-label">수량</label>
      <div class="col-md-10">
        <input type="text" id="quantity" v-model="newStore.quantity"
class="form-control">
      </div>
      <br><br><br>
      <label for="exampleSelect" class="col-md-2 col-form-label">카테고리
</label>
      <div class="col-md-10">
        <select class="form-control" id="exampleSelect"
v-model="newStore.category">
          <option value="의류">의류</option>
          <option value="식품">식품</option>
          <option value="생활용품">생활용품</option>
          <option value="가전디지털">가전디지털</option>
          <option value="스포츠/레저">스포츠/레저</option>
          <option value="자동차용품">자동차용품</option>
          <option value="도서/음반/DVD">도서/음반/DVD</option>
          <option value="완구/취미">완구/취미</option>
          <option value="문구/오피스">문구/오피스</option>
          <option value="반려동물용품">반려동물용품</option>
          <option value="뷰티">뷰티</option>
          <option value="출산/유아동">출산/유아동</option>
          <option value="주방용품">주방용품</option>
        </select>
      </div>
      <br><br><br>
      <label for="description" class="col-md-2 col-form-label">설명</label>
      <div class="col-md-10">

```

```

        <input type="text" id="description" v-model="newStore.description"
class="form-control">
      </div>
      <br><br><br>
      <label for="description" class="col-md-2 col-form-label">이미지 </label>
      <div class="col-md-10">
        <div id="app">
          <file-pond
            name="bin"
            ref="pond"
            label-idle="마우스로 이미지 파일을 끌어오거나
<strong><span class='filepond--label-action'>직접 첨부</span></strong>"
            allow-multiple="false"
            v-bind:instantUpload="instanceFlag"
            max-files="1"
            v-bind:server="server"
            accepted-file-types="image/jpeg, image/png"
            v-on:init="handleFilePondInit"
            v-on:processfile="onload"
          />
        </div>
        <br>
        <button @click="newStoreSubmit" type="button" class="btn
btn-primary btn-lg col-md-12">상품등록</button>
      </div>
    </div>
  </template>

  <script>
    import vueFilePond from 'vue-filepond'
    // Import FilePond styles
    import 'filepond/dist/filepond.min.css'
    // Import FilePond plugins
    // Please note that you need to install these plugins separately
    // Import image preview plugin styles
    import 'filepond-plugin-image-preview/dist/filepond-plugin-image-preview.min.css'
    // Import image preview and file type validation plugins
    import FilePondPluginFileValidateType from 'filepond-plugin-file-validate-type'
    import FilePondPluginImagePreview from 'filepond-plugin-image-preview'
    // Create component
    const FilePond = vueFilePond(
      FilePondPluginFileValidateType,
      FilePondPluginImagePreview
    );
    export default {
      name: "CreateStore",
      data(){
        return {
          server: {
            //url: "http://localhost:3000/users/imgupload",
            url: `/users/imgupload`,
            process: {
              headers: {
                Authorization : this.$store.state.pToken
              }
            }
          },
          newStore : {
            name: "",

```

```

        price: "",
        quantity: "",
        category: "",
        description: "",
        seller: "",
        number: "",
        image: ""
    },
    instanceFlag: false
},
props: {
    choice: ""
},
watch: {
    choice: function (category) {
        if (category === 'Create') {
            console.log('왔다: ' + category);
            this.$store.dispatch('GetProfile')
                .then(response => {
                    //alert('토큰검증 성공: ' + JSON.stringify(response.data.user));
                    console.log('토큰검증 성공');
                    //console.log('response: ' + JSON.stringify(response));
                    if (response.data.user.indi === 0) {
                        let UserNumber = {
                            number: response.data.user.number
                        };
                        return this.$store.dispatch('FoundEnt', UserNumber);
                    } else {
                        console.log('기업 검증 실패');
                        alert('기업 검증 실패');
                        this.$store.dispatch('LOGOUT');
                        this.$router.replace({path: '/Login'});
                    }
                })
                .then(res => {
                    if (res.data.store.seller === 1) {
                        this.server.process.headers = this.$store.state.pToken;
                        this.newStore.seller = res.data.store.company;
                        this.newStore.number = res.data.store.number;
                        console.log('판매자 검증 성공');
                    } else {
                        console.log('판매자 검증 실패');
                        alert('판매자 검증 실패');
                        this.$store.dispatch('LOGOUT');
                        this.$router.replace({path: '/Login'});
                    }
                })
                .catch(err => {
                    console.log('검증 실패' + err);
                    //alert(err);
                    this.$store.dispatch('LOGOUT');
                    this.$router.replace({path: '/Login'});
                })
        } else {
            console.log('안왔다: ');
        }
    }
},

```

```

methods : {
  newStoreSubmit : function () {
    this.$store.dispatch('NewProduct', this.newStore)
      .then( response => {
        if(response.data.success == true) {
          alert('상품 등록 성공');
          console.log('상품 등록 성공');
          this.$router.replace({ path : '/AllProduct' });
        } else {
          alert('상품 등록 실패 1');
          console.log('상품 등록 실패 1');
        }
      }). catch((err) => {
        alert('상품 등록 실패 2');
        console.log('상품 등록 실패 2');
      });
  },
  handleFilePondInit: function() {
    console.log('FilePond has initialized'+this.$refs.pond);
    // FilePond instance methods are available on `this.$refs.pond`
  },
  onload : function (e, r) {
    if (e) {
      console.log(e);
    } else {
      console.log(r);
      console.log(r.serverId.filename);
      console.log(r.fileExtension);
      console.log(r.serverId);
      var link = r.serverId;
      this.newStore.image = link;
      console.log("img : " + this.newStore.image);
    }
  },
  created() {
    this.$store.dispatch('GetProfile')
      .then( response => {
        //alert('토큰검증 성공 : '+JSON.stringify(response.data.user));
        console.log('토큰검증 성공');
        //console.log('response : '+JSON.stringify(response));
      });
  }
}
</script>
<style scoped>
</style>

```

```

<DetailOrder.vue>
<template>
  <div class="row">
    <div class="col-md-2"> </div>
    <div class="col-md-8">
      <h2>주문번호 : {{ordernumber.orderno}}</h2>
      <hr noshade/>
      <table class="table">
        <thead>

```

```

<tr class="table-active">
  <th scope="col">상품정보 </th>
  <th scope="col">금액 </th>
  <th scope="col">수량 </th>
  <th scope="col">판매자 </th>
  <th scope="col">영수증 </th>
</tr>
</thead>
<tbody v-for="Pro in Products">
  <tr>
    <td>
      <div class="media">
        
        <div class="media-body">
          <h5 class="mt-0">{{Pro.productname}}</h5>
          <h6 class="text-muted">
            {{Pro.description}}<br>
            {{Pro.category}}
          </h6>
        </div>
      </div>
    <td> <h5>{{(Pro.oquantity*Pro.price).toLocaleString()}}원 </h5> </td>
    <td> <h5>{{Pro.oquantity}}개 </h5> </td>
    <td> <h5>{{Pro.seller}} </h5> </td>
    <td>
      <h5 v-if="receipt == false">미발급 </h5>
      <h5 v-if="receipt == true">발급 완료 </h5>
    </td>
  </tr>
</tbody>
</table>
<hr noshade/>
<div class="card">
  <div class="card-body">
    <div class="row">
      <h6 class="col-md-1"> </h6>
      <h6 class="col-md-2"> <strong>주문자 </strong> </h6>
      <h6 class="col-md-3">{{order.orderer}} </h6>
      <h6 class="col-md-1"> </h6>
      <h6 class="col-md-2"> <strong>구매자 </strong> </h6>
      <h6 class="col-md-3">{{order.buyer}} </h6>
    </div>
    <hr class="my-4">
    <div class="row">
      <h6 class="col-md-1"> </h6>
      <h6 class="col-md-2"> <strong>배송 주소 </strong> </h6>
      <h6 class="col-md-3">{{order.delivery_address}} </h6>
      <h6 class="col-md-1"> </h6>
      <h6 class="col-md-2"> <strong>전화번호 </strong> </h6>
      <h6 class="col-md-3">{{order.delivery_tel}} </h6>
    </div>
    <hr class="my-4">
    <div class="row">
      <h6 class="col-md-1"> </h6>
      <h6 class="col-md-2"> <strong>결제 시간 </strong> </h6>
      <h6 class="col-md-3"> </h6>
      <h6 class="col-md-1"> </h6>
    </div>
  </div>
</div>
<div class="col-md-4">{{date.getFullYear()}}-{{("0" + (date.getMonth() + 1)).slice(-2)}}-{{("0" + (date.getDate() + 1)).slice(-2)}}

```

```

e()+1)).slice(-2)}}
{{{("0" + (date.getHours() + 1)).slice(-2)}}:{{{("0" + (date.getMinutes() + 1)).slice(-2)}}:{{{("0" + (date.getSeconds() + 1)).slice(-2)}}}</h6>
    <h6 class="col-md-1"></h6>
  </div>
</div>
</div>
<hr noshade/>
<div class="card">
  <div class="card-body">
    <div class="row">
      <h6 class="col-md-3"></h6>
      <h6 class="col-md-3">총 주문 상품수</h6>
      <h6 class="col-md-3">{{{kind[0]}}}<div class="col-md-3">{{{kind[1]}}}개</h6>
    </div>
    <hr class="my-4">
    <div class="row">
      <h6 class="col-md-3"></h6>
      <h6 class="col-md-3">총 결제금액</h6>
      <h5 class="col-md-4">{{{allprice.toLocaleString()}}}</h5>
      <h6 class="col-md-2"></h6>
    </div>
  </div>
</div>
<br>
<div class="row">
  <div class="col-md-12">
    <div v-if="receipt == true">
      <button @click="Receiptcheck" type="button" class="btn btn-lg btn-primary col-md-12">영수증 확인</button>
      <div v-if="receiptchecking == true">
        <hr noshade/>
        <br>
        <h3 class="align-center">영수증</h3>
        <br>
        <h5>주 소 : {{{user.addr}}}</h5>
        <h5>대표자 : {{{user.name}}}</h5>
        <h5>전화번호 : {{{user.tel}}}</h5>
        <h5>매출일 : {{{date.getFullYear()}}}-{{{("0" + (date.getMonth() + 1)).slice(-2)}}}-{{{("0" + (date.getDate() + 1)).slice(-2)}}} / {{{("0" + (date.getHours() + 1)).slice(-2)}}:{{{("0" + (date.getMinutes() + 1)).slice(-2)}}:{{{("0" + (date.getSeconds() + 1)).slice(-2)}}}</h5>
        <h5>번 호 : {{{order.order_no}}}</h5>
        <br>
        <table class="table">
          <thead>
            <tr class="table-active">
              <th scope="col">상품명</th>
              <th scope="col">단가</th>
              <th scope="col">수량</th>
              <th scope="col">금액</th>
            </tr>
          </thead>
          <tbody v-for="Pro in Products">
            <tr>
              <td>
                <div class="media">
                  <div class="media-body">
                    <
                    h

```

```

class="mt-0">{{Pro.productname}}</h5>
</div>
</div>
</td>
<td><h5>{{(Pro.price).toLocaleString()}}</h5></td>
<td><h5>{{Pro.oquantity}}</h5></td>
</tr>
</tbody>
</table>
<div class="card" v-for="reSign in receiptSign">
  <div class="card-body">
    <div class="row">
      <h6 class="col-md-3"><strong>영수증 서명값
</strong></h6>
      <h6 class="col-md-6">{{reSign}}</h6>
      <h6 class="col-md-3"><button
@click="receiptcheckSubmit" type="button" class="btn btn-primary col-md-12">영수증 서명
확인</button></h6>
    </div>
  </div>
</div>
</div>
</div>
<br><br><br>
</div>
</div>
</div>
</template>
<script>
export default {
  name: "DetailOrder",
  data () {
    return {
      user : {},
      temp : [],
      Products : [],
      size : 300,
      kind : [ 0 ],
      allprice : 0,
      choiceType : null,
      allow : false,
      order : {},
      pcode : "",
      pquan : [],
      seller : "",
      date : "",
      receiptPro : [],
      receiptSign : [],
      receipt : false,
      receipttmp : null,
      receiptissuing : false,
      receiptchecking : false,
      ordernumber : {
        ordeno : this.$route.params.order
      },
      //lnk : 'http://localhost:3000/img/',
      lnk : "/img/"
    }
  }
}

```

```

    }
  },
  created() {
    this.$store.dispatch('GetProfile')
      .then(response => {
        console.log('토큰검증 성공');
        this.user = response.data.user;
        return this.$store.dispatch('GetOrder', this.ordernumber);
      })
      .then(response => {
        console.log('get order : ' + JSON.stringify(response.data));
        this.order = response.data.order[0];
        var p = response.data.order[0].product;
        var p1 = p.split(',');
        var p2 = new Array;
        var p3 = new Array;
        for (var i = 0; i < p1.length; i++) {
          p2.push(p1[i].split('/'));
          p3.push(p2[i][1]);
        }
        console.log('p2 : ' + JSON.stringify(p2));
        console.log('p3 : ' + JSON.stringify(p3));
        this.pquan = p2;
        var pcode2 = {
          productcode: p
        };
        console.log("pcode2 : " + JSON.stringify(pcode2));
        return this.$store.dispatch('GetProductDetail2', pcode2);
      })
      .then(response => {
        console.log('product detail : ' + JSON.stringify(response.data));
        let pp = response.data.result;
        for (let i = 0; i < pp.length; i++) {
          for (let j = 0; j < pp[i].length; j++) {
            this.pquan[i][j] *= 1;
            console.log('pp ' + i + ' : ' + pp[i].productcode);
            console.log('pquan ' + j + ' : ' + this.pquan[j][0]);
            if (pp[i].productcode == this.pquan[j][0]) {
              pp[i].oquantity = this.pquan[j][1];
              console.log('일치');
            } else {
              console.log('불일치');
            }
          }
        }
        this.Products = pp;
        this.seller = response.data.result[0].seller;
        this.imglnk();
        //this.quantityAppend(response.data.result);
        console.log('product detail2 : ' + JSON.stringify(this.Products));
        if (this.user.number == this.order.orderer) {
          this.allow = true;
        } else {
          alert('잘못된 요청입니다.');
```



```

let char01 = '';
receipt00 = this.order.receipt.split(',');
for (var q=0; q<receipt00.length; q++){
    receipt01.push(receipt00[q].split('+'));
}
for (var l = 0; l < receipt01.length; l++) {
    receipt02.push(receipt01[l][0].split('-'));
    for (var m = 0; m < receipt02[l].length; m++) {
        receipt03.push(receipt02[l][m].split('/'));
        //receipt04.push(receipt03[j][0]);
    }
}
let receiptCount = 0;
for (var n=0; n<receipt03.length; n++){
    if (receiptCount > 0) {
        this.receipt = true;
        console.log('영수증 있음 3');
        break;
    } else {
        for (var o=0; o<this.Products.length; o++){
            if (receipt03[n][0] == this.Products[o].productcode) {
                receiptCount += 1;
                console.log('영수증 있음 1');
                char01 = receipt03[n][0] + '/';
                var isExist;
                for (var u = 0; u < receipt01.length; u++){
                    isExist = (receipt01[u][0].indexOf(char01)!=
-1);

                    console.log('isExist : '+isExist);
                    console.log('u : '+u);
                    if (isExist == true) {
                        console.log('receipt01['+u+'][1]
'+receipt01[u][1]);

                        this.receiptSign.push(receipt01[u][1]);
                        break;
                    }
                }
                break;
            } else {
                console.log('영수증 없음 1');
            }
        }
    }
}
if (n== (receipt03.length - 1)) {
    if (receiptCount == 0) {
        this.receipt = false;
        console.log('영수증 없음 2');
    } else {
        this.receipt = true;
        console.log('영수증 있음 2');
        break;
    }
}
}
} else {
    this.receipt = false;
    console.log('영수증 없음 4');
}
})
.finally(() => {
    this.kind[0] = this.Products.length;

```

```

        this.kind[1] = 0;
        for (let i = 0; i < this.Products.length; i++) {
            this.allprice = this.allprice + this.Products[i].oquantity *
this.Products[i].price;
            this.Products[i].oquantity *= 1; //스트링을 정수로 형변환
            this.kind[1] = this.kind[1] + this.Products[i].oquantity;
        }
        this.date = new Date((this.order.trade_time *= 1));
    })
    .catch(err => {
        alert('잘못된 요청입니다. : '+err);
        this.$router.replace({path: '/'});
    });
},
methods : {
    Receiptcheck : function() {
        if (this.receiptchecking == true) {
            this.receiptchecking = false;
        } else {
            this.receiptchecking = true;
        }
    },
    receiptcheckSubmit : function() {
        let orderReceiptnull = this.order;
        orderReceiptnull.receipt = null;
        let receiptR = {
            user : this.user,
            order : orderReceiptnull,
            product : "",
            pprice : 0,
            signature : this.receiptSign
        };
        for (var i = 0; i < this.Products.length; i++) { //product 문장 만들기
            if (i == 0) {
                receiptR.product = this.Products[i].productcode + '/' +
this.Products[i].quantity;
            } else {
                receiptR.product = receiptR.product + '-' +
this.Products[i].productcode + '/' + this.Products[i].quantity;
            }
            receiptR.pprice += (this.Products[i].quantity * this.Products[i].price);
        }
        console.log("receiptR : " + JSON.stringify(receiptR));
        this.$store.dispatch('ReceiptValidateRequest', receiptR)
            .then( response => {
                console.log('영수증 검증 : '+JSON.stringify(response));
                alert('검증 완료 완료');
                console.log('ReceiptValidate Request Success :
'+JSON.stringify(response));
            }).catch( err => {
                console.log('ReceiptValidate Request Err : '+ err);
            });
    },
    quantityAppend : function(Prod) {
        for (let i=0; i<Prod.length; i++) {
            for(let j=0; j<Prod.length; j++) {
                console.log('prod '+i+' : '+Prod[i].productcode);
                console.log('pquan '+j+' : '+ this.pquan[j][0]);
                if(Prod[i].productcode == this.pquan[j][0]) {
                    Prod[i].oquantity = this.pquan[j][1];
                    console.log('일치');
                }
            }
        }
    }
}

```

```

    }
  }
  this.Products = Prod;
},
imgInk : function () {
  for (var i=0; i<(this.Products.length); i++) {
    this.Products[i].thumbnail = this.Ink+this.Products[i].thumbnail;
  }
},
watch : {
}
}
</script>

<style scoped>
  .widthSet {
    max-width: 80px;
  }
  .heightSet {
    max-height: 80px;
  }
  .align-center {
    text-align: center;
  }
</style>

```

```

<DetailOrderSell.vue>

<template>
  <div class="row">
    <div class="col-md-2"></div>
    <div class="col-md-8">
      <h2>주문번호 : {{ordernumber.orderno}}</h2>
      <hr noshade/>
      <table class="table">
        <thead>
          <tr class="table-active">
            <th scope="col">판매 상품정보</th>
            <th scope="col">금액</th>
            <th scope="col">수량</th>
            <th scope="col">판매자</th>
            <th scope="col">영수증</th>
          </tr>
        </thead>
        <tbody v-for="Pro in Products">
          <tr v-if="Pro.seller == user.id">
            <td>
              <div class="media">
                
                <div class="media-body">
                  <h5 class="mt-0">{{Pro.productname}}</h5>
                  <h6 class="text-muted">
                    {{Pro.description}}<br>
                    {{Pro.category}}
                  </h6>

```

```

        </div>
    </div>
</td>
<td><h5>{{(Pro.quantity*Pro.price).toLocaleString()}}원</h5></td>
<td><h5>{{Pro.quantity}}개</h5></td>
<td><h5>{{Pro.seller}}</h5></td>
<td>
    <h5 v-if="receipt == false">미발급</h5>
    <h5 v-if="receipt == true" >발급 완료</h5>
</td>
</tr>
</tbody>
</table>
<hr noshade/>
<div class="card">
    <div class="card-body">
        <div class="row">
            <h6 class="col-md-1"></h6>
            <h6 class="col-md-2"><strong>주문자</strong></h6>
            <h6 class="col-md-3">{{order.orderer}}</h6>
            <h6 class="col-md-1"></h6>
            <h6 class="col-md-2"><strong>구매자</strong></h6>
            <h6 class="col-md-3">{{order.buyer}}</h6>
        </div>
        <hr class="my-4">
        <div class="row">
            <h6 class="col-md-1"></h6>
            <h6 class="col-md-2"><strong>배송 주소</strong></h6>
            <h6 class="col-md-3">{{order.delivery_address}}</h6>
            <h6 class="col-md-1"></h6>
            <h6 class="col-md-2"><strong>전화번호</strong></h6>
            <h6 class="col-md-3">{{order.delivery_tel}}</h6>
        </div>
        <hr class="my-4">
        <div class="row">
            <h6 class="col-md-1"></h6>
            <h6 class="col-md-2"><strong>결제 시간</strong></h6>
            <h6 class="col-md-3"></h6>
            <h6 class="col-md-1"></h6>
            <div class="col-md-4">{{date.getFullYear()}}-{{("0" + (date.getMonth() + 1)).slice(-2)}}-{{("0" + (date.getDate() + 1)).slice(-2)}}
            /
            {{("0" + (date.getHours() + 1)).slice(-2)}}:{{("0" + (date.getMinutes() + 1)).slice(-2)}}:{{("0" + (date.getSeconds() + 1)).slice(-2)}}</div>
            <h6 class="col-md-1"></h6>
        </div>
        </div>
    </div>
<br>
<div class="card">
    <div class="card-body">
        <div class="row">
            <h6 class="col-md-3"></h6>
            <h6 class="col-md-3"><strong>주문 상품수</strong></h6>
            <h6 class="col-md-3">{{kind[0]}}종 {{kind[1]}}개</h6>
            <h6 class="col-md-3"></h6>
        </div>
        <hr class="my-4">
        <div class="row">
            <h6 class="col-md-3"></h6>
            <h6 class="col-md-3"><strong>결제 금액</strong></h6>

```

```

        <h5 style="color: crimson"
class="col-md-4">{{allprice.toLocaleString()}}원</h5>
        <h6 class="col-md-2"></h6>
    </div>
</div>
<br>
<div class="row">
    <div class="col-md-12">
        <div v-if="receipt == false">
            <button @click="ReceiptissueButton" type="button" class="btn
btn-lg btn-primary col-md-12">영수증 발급</button>
            <div v-if="receiptissuing == true">
                <br>
                <div class="alert alert-warning" role="alert">
                    <h3 class="page-header">인증서 비밀번호 입력</h3>
                    <div class="form-group">
                        <label>Password</label>
                        <input type="password" class="form-control"
v-model="Cpass" name="password">
                    </div>
                    <button @click="Receiptissue" type="button" class="btn
btn-primary align-self-center">영수증 발급</button>
                </div>
            </div>
            <div v-if="receipt == true">
                <button @click="Receiptcheck" type="button" class="btn btn-lg
btn-primary col-md-12">영수증 확인</button>
                <div v-if="receiptchecking == true">
                    <hr noshade/>
                    <br>
                    <h3 class="align-center">영수증</h3>
                    <br>
                    <h5>상 호 : {{Store.company}}</h5>
                    <h5>사업자 번호 : {{Store.crn}}</h5>
                    <h5>주 소 : {{user.addr}}</h5>
                    <h5>대표자 : {{user.name}}</h5>
                    <h5>전화번호 : {{user.tel}}</h5>
                    <h5>매출일 :
{{date.getFullYear()}}-{{("0"+(date.getMonth()+1)).slice(-2)}}-{{("0"+(date.getDate()+1)).slice(-2)}} /
{{("0"+(date.getHours()+1)).slice(-2)}}:{{("0"+(date.getMinutes()+1)).slice(-2)}}:{{("0"+(date.getSeco
nds()+1)).slice(-2)}}</h5>
                    <h5>번 호 : {{order.order_no}}</h5>
                    <br>
                    <table class="table">
                        <thead>
                            <tr class="table-active">
                                <th scope="col">상품명</th>
                                <th scope="col">단가</th>
                                <th scope="col">수량</th>
                                <th scope="col">금액</th>
                            </tr>
                        </thead>
                        <tbody v-for="Pro in Products">
                            <tr>
                                <td>
                                    <div class="media">
                                        <div class="media-body">
                                            <
                                            h
class="mt-0">{{Pro.productname}}</h5>

```

```

        </div>
      </td>
      <td><h5>{{{(Pro.price).toLocaleString()}}}</h5></td>
      <td><h5>{{{Pro.quantity}}}</h5></td>
    </tr>
  </tbody>
</table>
<div class="card" v-for="reSign in receiptSign">
  <div class="card-body">
    <div class="row">
      <h6 class="col-md-3"><strong>영수증   서명값
    </strong></h6>
      <h6 class="col-md-6">{{{reSign}}}</h6>
      <h6 class="col-md-3"><button
@click="receiptcheckSubmit" type="button" class="btn btn-primary col-md-12">영수증 서명
확인</button></h6>
    </div>
  </div>
</div>
</div>
<br><br><br>
</div>
</div>
</div>
</template>
<script>
export default {
  name: "DetailOrderSell",
  data () {
    return {
      user : {},
      temp : [],
      Products : [],
      Store : {},
      size : 300,
      kind : [ 0 ],
      allprice : 0,
      choiceType : null,
      allow : false,
      order : {},
      pcode : "",
      pquan : [],
      seller : "",
      receiptPro : [],
      receiptSign : [],
      receipt : false,
      receipttmp : null,
      receiptissuing : false,
      receiptchecking : false,
      Cpass : "",
      ordernumber : {
        orderno : this.$route.params.order
      },
      date : "",
      //lnk : 'http://localhost:3000/img/',
    }
  }
}

```

```

        lnk : "/img/"
    },
    created() {
        this.$store.dispatch('GetProfile')
        .then(response => {
            console.log('토큰검증 성공');
            this.user = response.data.user;
            console.log('user : ' + JSON.stringify(this.user));
            let UserNumber1 = {
                number : this.user.number
            };
            return this.$store.dispatch('FoundEnt', UserNumber1);
        })
        .then(response => {
            this.Store = response.data.store;
            console.log('Store : ' + JSON.stringify(this.Store));
            return this.$store.dispatch('GetOrder', this.ordernumber);
        })
        .then(response => {
            console.log('get order : ' + JSON.stringify(response.data));
            this.order = response.data.order[0];
            var p = response.data.order[0].product;
            var p1 = p.split(';');
            var p2 = new Array;
            var p3 = new Array;
            for (var i = 0; i < p1.length; i++) {
                p2.push(p1[i].split('/'));
                p3.push(p2[i][1]);
            }
            console.log('p2 : ' + JSON.stringify(p2));
            console.log('p3 : ' + JSON.stringify(p3));
            this.pquan = p2;
            var re = response.data.order[0].receipt;
            if ( re != null ) {
                this.receipttmp = re;
            } else {
                this.receipt = null;
            }
            let UserNumber = {
                number : this.user.number
            };
            console.log("UserNumber : " + JSON.stringify(UserNumber));
            return this.$store.dispatch('MyGetProduct2', UserNumber);
        })
        .then(response => {
            console.log('My product : ' + JSON.stringify(response.data));
            let pp = response.data.Product;
            console.log('pp : ' + JSON.stringify(pp)); //판매자의 상품들
            console.log("pquan : " + JSON.stringify(this.pquan)); //주문정보의 상품코

            //console.log("pquan[0] : " + JSON.stringify(this.pquan[0]));
            //console.log("pquan[0][0] : " + JSON.stringify(this.pquan[0][0]));
            //console.log("pquan[0][1] : " + JSON.stringify(this.pquan[0][1]));
            if ( this.receipt != null ) {
                var r = this.receipttmp;
                var r1 = r.split('+');
                var r2 = new Array;
                var r3 = new Array;
                for (var i = 0; i < r1.length; i++) {
                    r2.push(r1[i].split('-'));
                }
            }
        })
    }
}

```

드+수량

```

        r3.push(r2[i][0]);
    }
    console.log('r2 : ' + JSON.stringify(r2));
    console.log('r3 : ' + JSON.stringify(r3));
}
let ppp = new Array;
for (var j=0; j<this.pquan.length; j++){
    for (var k=0; k<pp.length; k++) {
        if (this.pquan[j][0] == pp[k].productcode) {
            ppp.push({
                productcode : pp[k].productcode,
                productname : pp[k].productname,
                description : pp[k].description,
                category : pp[k].category,
                price : pp[k].price,
                seller : pp[k].seller,
                thumbnail : this.lnk + pp[k].thumbnail,
                quantity : this.pquan[j][1] * 1,
                receipt : null
            })
        }
    }
}
console.log('ppp : '+JSON.stringify(ppp));
this.Products = ppp;
var receipt00;
var receipt01 = [];
var receipt02 = [];
var receipt03 = [];
let char01 = '';
receipt00 = this.order.receipt.split(',');
for (var q=0; q<receipt00.length; q++){
    receipt01.push(receipt00[q].split('+'));
}
for (var l = 0; l < receipt01.length; l++) {
    receipt02.push(receipt01[l][0].split('-'));
    for (var m = 0; m < receipt02[l].length; m++) {
        receipt03.push(receipt02[l][m].split('/'));
        //receipt04.push(receipt03[j][0]);
    }
}
let receiptCount = 0;
for (var n=0; n<receipt03.length; n++){
    if (receiptCount > 0) {
        this.receipt = true;
        console.log('영수증 있음 3');
        break;
    } else {
        for (var o=0; o<this.Products.length; o++){
            if (receipt03[n][0] == this.Products[o].productcode) {
                receiptCount += 1;
                console.log('영수증 있음 1');
                char01 = receipt03[n][0] + '/';
                var isExist;
                for (var u = 0; u < receipt01.length; u++){
                    isExist = (receipt01[u][0].indexOf(char01)!= -1);
                    console.log('isExist : '+isExist);
                    console.log('u : '+u);
                    if (isExist == true) {
                        console.log('receipt01['+u+']'[1]
'+receipt01[u][1]);

```



```

                                this.receiptSign.push(receipt01[u][1]);
                                break;
                            }
                        }
                        break;
                    } else {
                        console.log('영수증 없음 1');
                    }
                }
            }
            if (n == (receipt03.length - 1)) {
                if (receiptCount == 0) {
                    this.receipt = false;
                    console.log('영수증 없음 2');
                } else {
                    this.receipt = true;
                    console.log('영수증 있음 2');
                    break;
                }
            }
        }
    })
    .finally(() => {
        this.kind[0] = this.Products.length;
        this.kind[1] = 0;
        for (let i = 0; i < this.Products.length; i++) {
            this.allprice = this.allprice + this.Products[i].quantity
this.Products[i].price;
            this.Products[i].quantity *= 1; //스트링을 정수로 형변환
            this.kind[1] = this.kind[1] + this.Products[i].quantity;
        }
        this.date = new Date((this.order.trade_time *= 1));
    })
    .catch(err => {
        alert('잘못된 요청입니다.');
```

//this.\$router.replace({path: '/'});

```

    });
},
methods : {
    ReceiptissueButton : function() {
        if (this.receiptissuing == true) {
            this.receiptissuing = false;
        } else {
            this.receiptissuing = true;
        }
        console.log("order : " + JSON.stringify(this.order));
    },
    Receiptissue : function() {
        let receiptR = {
            pa : this.Cpass,
            user : this.user,
            order : this.order,
            product : "",
            pprice : 0,
        };
        for (var i = 0; i < this.Products.length; i++) { //product 문장 만들기
            if (i == 0) {
                receiptR.product = this.Products[i].productcode + '/' +
this.Products[i].quantity;
            } else {
                receiptR.product = receiptR.product + '-' +

```

```

this.Products[i].productcode + '/' + this.Products[i].quantity;
    }
    receiptR.pprice += (this.Products[i].quantity * this.Products[i].price);
}
console.log("receiptR : " + JSON.stringify(receiptR));
this.$store.dispatch('ReceiptRequest', receiptR)
    .then( (response) => {
        console.log('영수증 : ' + JSON.stringify(response));
        alert('발급 완료');
        console.log('ReceiptRequest Success : ' + JSON.stringify(response));
        var p = response.data.order;
        //var p1 = p.split('/');
        //this.$router.push({ path : '/SalesHistory' });
    }).catch( err => {
        console.log('ReceiptRequest Err : ' + err);
    });
},
Receiptcheck : function() {
    if (this.receiptchecking == true) {
        this.receiptchecking = false;
    } else {
        this.receiptchecking = true;
    }
},
receiptcheckSubmit : function() {
    let orderReceiptnull = this.order;
    orderReceiptnull.receipt = null;
    let receiptR = {
        user : this.user,
        order : orderReceiptnull,
        product : "",
        pprice : 0,
        signature : this.receiptSign
    };
    for (var i = 0; i < this.Products.length; i++) { //product 문장 만들기
        if (i == 0) {
            receiptR.product = this.Products[i].productcode + '/' +
this.Products[i].quantity;
        } else {
            receiptR.product = receiptR.product + '-' +
this.Products[i].productcode + '/' + this.Products[i].quantity;
        }
        receiptR.pprice += (this.Products[i].quantity * this.Products[i].price);
    }
    console.log("receiptR : " + JSON.stringify(receiptR));
    this.$store.dispatch('ReceiptValidateRequest', receiptR)
        .then( response => {
            console.log('영수증 검증 : ' + JSON.stringify(response));
            alert('영수증 검증 성공');
            console.log('ReceiptValidate Request Success : ' + JSON.stringify(response));
        }).catch( err => {
            console.log('ReceiptValidate Request Err : ' + err);
        });
},
quantityAppend : function(Prod) {
    for (let i=0; i<Prod.length; i++) {
        for(let j=0; j<Prod.length; j++) {
            console.log('prod '+i+' : '+Prod[i].productcode);
            console.log('pquan '+j+' : '+ this.pquan[j][0]);
            if(Prod[i].productcode == this.pquan[j][0]) {

```

```

        Prod[i].oquantity = this.pquan[j][1];
        console.log("일치");
    }
    }
    }
    this.Products = Prod;
},
imglnk : function () {
    for (var i=0; i<(this.Products.length); i++) {
        this.Products[i].thumbnail = this.lnk+this.Products[i].thumbnail;
    }
},
watch : {
}
}
}
</script>

<style scoped>
    .widthSet {
        max-width: 80px;
    }
    .heightSet {
        max-height: 80px;
    }
    .align-center {
        text-align: center;
    }
</style>

```

```

<DetailProduct.vue>
<template>
    <div>
        <div class="row">
            <div class="col-md-6">
                <div class="imageBox">
                    
                </div>
            </div>
            <div class="col-md-1">
            </div>
            <div class="col-md-5">
                <br><br>
                <table border="0">
                    <tr>
                        <td>
                            <div class="float-left"><strong>{{Product.productname}}</strong></div>
                        </td>
                        <td>
                            <h3>
                                <div class="float-left"><b>남은 수량 :</b>
                                <div class="float-left"><b>{{Product.allquantity.toLocaleString()}}개</b>
                            </h3>
                        </td>
                    </tr>
                </table>
            </div>
        </div>
    </div>

```

```

        <td>
            <h3 class="float-left">{{Product.price.toLocaleString()}}원</h3>
        </td>
    </tr>
    <tr>
        <td>
            <br>
        </td>
    </tr>
    <tr>
        <td>
            <br>
        </td>
    </tr>
    <tr>
        <td>
            <br>
        </td>
    </tr>
    <tr>
        <td>
            <br>
        </td>
    </tr>
    <tr>
        <td>
            <button class="btn btn-sm btn-outline-dark float-left"
@click="decreaseQuantity" type="button">-</button>
            <input type="text" maxlength="2" title="수량" value="1"
v-model="quantity">
            <button class="btn btn-sm btn-outline-dark float-right"
@click="increaseQuantity" type="button">+</button>
        </td>
    </tr>
    <tr>
        <td></td>
        <td colspan="2"></td>
    </tr>
</table>
<h6>총 상품금액 {{(Product.price*quantity).toLocaleString()}}원</h6>
<table border="0">
    <tr>
        <td>
            <button @click="addBasketSubmit" type="button" class="btn
btn-block btn-lg btn-outline-success">장바구니</button>
        </td>
        <td>
            <button @click="CreateOrderSubmit" type="button" class="btn
btn-block btn-lg btn-primary">바로구매</button>
        </td>
    </tr>
</table>
</div>
</div>
</template>

<script>
export default {
  name: "DetailProduct",
  data () {
    return {
      Products : [],
      Product : {},
      imageThumbnail : "",
      quantity : 1,
    }
  },

```

```

created() {
  let product = {
    productcode : this.$route.params.product
  };
  this.$store.dispatch('GetProductDetail', product)
    .then( response => {
      //alert('성공 : '+JSON.stringify(response.data.user));
      console.log('성공');
      console.log('response : '+JSON.stringify(response));
      this.Products = response.data.result;
      this.Product = this.Products[0];

      //this.imageThumbnail
      "http://localhost:3000wwwimgwww"+this.Product.thumbnail;
      this.imageThumbnail = "/img/"+this.Product.thumbnail;
    });
},
methods : {
  decreaseQuantity : function() {
    if(this.quantity <= 1) {
      this.quantity = 1;
      alert('최소 수량은 1개입니다.');
```

=

```

        alert('add Basket Failure');
        console.log('add Basket Failure')
    }
    }).catch( err => {
        console.log('add Basket Err : '+ err);
    });
},
CreateOrderSubmit : function () {
    var arr = [];

    var a = String(this.Product.productcode);
    var b = String(this.quantity);

    console.log("number"+a);
    let ab = String(a+'&'+b);
    arr.push(ab);
    this.$router.push({ path : '/CreateOrder/'+1+'/'+arr });
    console.log("product"+JSON.stringify(this.Product));
    /*
    this.$router.push({
        path
    });
    */
}

}

}

</script>

<style scoped>
.tab {
    padding-left: 1.8em;
}
.productgroup {
    width: 1280px;
    position: absolute;
    top: 250px;
    left: 18%;
    //background: #fff;
    //transform: translate(-50%, -50%)
}
.widthSet {
    max-width: 498px;
}
.heightSet {
    max-height: 498px;
}
/*
img.border-shadow{
    border:0px solid #888888;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}*/
.imageBox{
    max-width: 500px;
    max-height: 500px;
    border:0px solid #888888;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}
</style>

```

```

<Home.vue>
<template>
  <div>
    <div class="jumbotron text-center">
      <h1>QR코드 간편결제 시스템</h1>
      <br>
      <div class="row">
        <div class="col-md-4">
          <qrcode-vue :value="value" :size="size" level="H"></qrcode-vue>
        </div>
        <div class="col-md-8">
          <h4>QR코드를 이용한 모바일 웹 애플리케이션</h4>
          <h4>인증서를 사용한 안전한 거래</h4>
          <h4>JSON Web Token기반의 무상태 서비스 제공</h4>
          <h4></h4>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <h3>Express Backend</h3>
        <p>Express는 웹 및 모바일 애플리케이션을 위한 일련의 강력한 기능을 제공하는 간
결하고 유연한 Node.js 웹 애플리케이션 프레임워크입니다.</p>
        <p>자유롭게 활용할 수 있는 수많은 HTTP 유틸리티 메소드 및 미들웨어를 통해 쉽
고 빠르게 강력한 API를 작성할 수 있습니다.</p>
      </div>
      <div class="col-md-4">
        <h3>Vue JS</h3>
        <p>Vue.js는 뷰(View)에 최적화된 프론트엔드 프레임워크입니다. 컨트롤러 대신 뷰 모
델을 가지는 MVVM(Model-View-ViewModel) 패턴을 기반으로 디자인되었으며, 컴포넌트를 사
용하여 재사용이 가능한 UI들을 묶고 뷰 레이어를 정리하는 것이 가장 강력한 기능입니
다.</p>
        <p></p>
      </div>
      <div class="col-md-4">
        <h3>JWT Tokens</h3>
        <p>JSON Web Token은 정보를 안전하게 전송하기 위해 정의된 공개된 표준(RFC
7519)입니다. JWT는 자체적으로 필요한 모든 정보를 포함합니다. 헤더 정보와, 실제 전달할 데
이터, 검증할 수 있는 서명 데이터를 모두 포함하고 있습니다.</p>
        <p>디지털 서명에 의해 검증할 수 있으며 신뢰할 수 있습니다. 비밀 값을 사용하는
HMAC 알고리즘이나 RDS or ECDSA와 같은 공개키, 개인키 쌍으로 서명될 수 있습니다.</p>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <h3>X.509 인증서</h3>
        <p>X.509는 PKI에서 사용하는 표준 인증서 형식이다. PKI에서 사용하는 공개키, 개인
키 같은 비대칭키를 X.509 인증서로 관리한다.</p>
        <p>X.509(V1)은 1993년 디렉터리 접근 제어를 위한 두 가지 내용을 추가하기 위해
X.509(V2) 형식으로 개정되었다. 그리고 e메일의 보안 요소 등 새로운 개념이 포함된
X.509(V3)가 1996년에 발표되었다.</p>
      </div>
      <div class="col-md-4">
        <h3>Vuex</h3>
        <p>Vuex는 Vue.js 애플리케이션에 대한 상태 관리 패턴 + 라이브러리 입니다. 애플
리케이션의 모든 컴포넌트에 대한 중앙 집중식 저장소 역할을 하며 예측 가능한 방식으로 상
태를 변경할 수 있습니다. 또한 Vue의 공식 확장 프로그램과 통합되어 설정 시간이 필요 없는
디버깅 및 상태 스냅 샷과 같은 고급 기능을 제공합니다.</p>
      </div>
      <div class="col-md-4">
        <h3></h3>

```

```

        <p></p>
    </div>
</div>
</div>

<!--
<div>
    <body>
    <div id="wrapper">-->
        <!--헤더시작
        <header>
            <iframe
                muted="muted"
                volume="0"
                width=100%
                height="480"

src="https://www.youtube.com/embed/4HG_CJzyX6A?rel=0;autoplay=1;mute=1;vq=hd720"
                frameborder="0"
                allow="accelerometer;      autoplay;      encrypted-media;      gyroscope;
picture-in-picture"
                allowfullscreen
            ></iframe>
        </header>
    -->
    <!--네비게이션-->
    <!--
    <nav>
        <p>nav</p>
    </nav>
    -->
    <!--콘텐츠부분-
    <div class="row">
        <div class="col-md-4">
            
        </div>
        <div class="col-md-4">
            
        </div>
        <div class="col-md-4">
            
        </div>
    </div>-->
    <!--
    <section>
        <p>section</p>
        <article>
            <p>
                What we do
            <br />STORY ABOUT US
            <br />Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
            eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
            quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
            </p>
        </article>
        <div class="row">
            <div class="col-md-4">
                
            </div>
            <div class="col-md-4">
                

```



```

    </div>
    <div class="col-md-4">
      
    </div>
  </div>
</section>
-->

<!--사이드바-->
<!--
<aside>
  <p>aside</p>
</aside>
-->
<!--풋터-->
<!--
<footer>
  <address>
    <br />코코넛페이 주소 : 서울특별시 서초구 강남대로
    <br />전화번호 : (02) 123-4567 사업자등록번호 123-4567-890 대표자 : 홍길동
    <br />copyright(c) 2019 coconut pay. all rights reserved
    <br />
    <div class="padding">
      * Not intended for use by people under 22 years old or those who have
      been previously diagnosed with atrial fibrillation.
      Apple Watch Series 3 has a water resistance rating of 50 meters under ISO
      standard 22810:2010. This means that it may be used for shallow-water activities like
      swimming in a pool or ocean. However, it should not be used for scuba diving,
      waterskiing, or other activities involving high-velocity water or submersion below shallow
      depth.
      Participating equipment manufacturers and gyms vary.
      Apple Music requires a subscription.
      Wireless service plan required for cellular service. Apple Watch and iPhone
      service provider must be the same. Not available with all service providers. Not all service
      providers support enterprise accounts or prepaid plans; check with your employer and
      service provider. Some legacy plans may not be compatible. Roaming is not available.
      Contact your service provider for more details. Check www.apple.com/watch/cellular for
      participating wireless carriers and eligibility.
      Features are subject to change. Some features, applications, and services may
      not be available in all regions or all languages. Click here to see complete list.
    </div>
  </address>
</footer>
-->
<!--
  <div class="padding">
    * Not intended for use by people under 22 years old or those who have been
    previously diagnosed with atrial fibrillation.
    Apple Watch Series 3 has a water resistance rating of 50 meters under ISO
    standard 22810:2010. This means that it may be used for shallow-water activities like
    swimming in a pool or ocean. However, it should not be used for scuba diving,
    waterskiing, or other activities involving high-velocity water or submersion below shallow
    depth.
    Participating equipment manufacturers and gyms vary.
    Apple Music requires a subscription.
    Wireless service plan required for cellular service. Apple Watch and iPhone service
    provider must be the same. Not available with all service providers. Not all service
    providers support enterprise accounts or prepaid plans; check with your employer and
    service provider. Some legacy plans may not be compatible. Roaming is not available.
    Contact your service provider for more details. Check www.apple.com/watch/cellular for
    participating wireless carriers and eligibility.
  </div>
-->

```

Features are subject to change. Some features, applications, and services may not be available in all regions or all languages. [Click here to see complete list.](#)

```
</div>
</div>
</body>
</div>-->
</template>

<script>
import QrcodeVue from 'qrcode.vue';
export default {
  name: "Home",
  data() {
    return {
      value : 'https://coconutpay.herokuapp.com/',
      size : 250
    }
  },
  components: {
    QrcodeVue
  }
};
</script>

<style scoped>
/*
.card-section {
  font-size: 20px;
}
body {
  text-align: center;
  color: #000;
  width: 100%;
}
div#wapper {
  width: 100%;
  text-align: center;
  margin: auto;
  height: -webkit-fill-available;
}
header,
footer,
nav,
aside,
section {
  /*border: 1px solid #999;
  padding: 10px;
}
header {
  height: 50%;
  /*background-color: #f8f8f8;
}
nav,
section,
aside {
  float: left;
  height: 800px;
}
nav {
  /*background-color: #f8f8f8;
  width: 10%;
```

```

}
section {
  /*background-color: #f8f8f8;
  width: 80%;
}
aside {
  /*background-color: #f8f8f8;
  width: 10%;
}
footer {
  font-size: 18px;
  text-align: center;
  color: black;
  height: auto;
  /*background-color: #f2f2f2;
  position: relative;
  clear: both;
}
article {
  color: #000;
  font-size: 15px;
  text-align: center;
  margin: 20px;
  /*background-color: #f8f8f8;
}
.padding {
  padding: 100px;
}
*/
</style>

```

```

<Login.vue>
<template>
  <div>
    <div class="row">
      <div class="col-md-3"> </div>
      <div class="col-md-6">
        <h2 class="page-header">로그인 </h2> <br>
        <div class="alert alert-warning" role="alert">
          <h3 class="page-header">ID/PASS 로그인 </h3>
          <div class="form-group">
            <label>ID (Username)</label>
            <input type="text" class="form-control" v-model="User.id"
name="username">
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" class="form-control"
v-model="User.password" name="password">
          </div>
          <button @click="onLoginSubmit" type="button" class="btn
btn-primary">ID/PASS 로그인 </button>
        </div>
      </div>
    </div>
    <!--
    <div class="col-md-6">
      <div class="alert alert-info" role="alert">
        <h3 class="page-header">인증서 간편 로그인 </h3>
        <p>
          인증서 로그인은 서버에서 인증서를 발급받은 경우에만 사용 가능함

```

니다.

```

        </p>
        <br>
        <div class="form-group">
            <label>ID (Username)</label>
            <input type="text" class="form-control" name="username1">
        </div>
        <input type="submit" class="btn btn-primary" value="전자서명 간
편 로그인">
    </div>
</div>
-->
<div class="col-md-3"> </div>
</div>
</template>
<script>
    export default {
        name: 'Login',
        data(){
            return {
                User : {
                    id: "",
                    password: ""
                }
            },
            methods:{
                onLoginSubmit() {
                    const User = this.User;
                    if ( !User.id || !User.password ) {
                        return false;
                    }
                    this.$store.dispatch('LOGIN', User)
                    .then( response => {
                        if(response.data.success === true) {
                            this.$store.commit('LOGIN', response);
                            console.log('로그인 성공');
                            alert('Login Success');
                            this.$router.replace({ path : '/' });
                        } else {
                            console.log("Login Error! 1");
                            return alert('Login Error');
                        }
                    })
                    .catch( err => {
                        console.log("Login Error! 2");
                        return alert('Login Error');
                    });
                }
            }
        }
    }
</script>
<style scoped>
</style>
```

```
<MyCategory.vue>
<template>
    <div class="list-group" v-if="choice == 'all'">
```

```

<h2>등록 상품 관리</h2>
<br>
<div v-for="product in Products">
  <a class="list-group-item list-group-item-action flex-column align-items-start">
    <div class="d-flex w-100 justify-content-between">
      <h4 class="mb-1">{{product.productname}}</h4>
      <h6 class="mb-1">수량 : {{product.allquantity}}</h6>
      <small class="text-muted">{{product.category}}</small>
    </div>
    <p class="mb-1">{{product.description}}</p>
    <small class="mb-1">상품번호 : {{product.productcode}}<br>
      {{product.price}}원</small>
    </a>
  <br>
</div>
</div>
</template>

<script>
export default {
  name: "MyCategory",
  props: {
    choice : ""
  },
  data () {
    return {
      user : {},
      Products : []
    }
  },
  methods : {
  },
  watch : {
    choice : function (category) {
      if ( category == 'all') {
        this.$store.dispatch('GetProfile')
          .then( response => {
            console.log('토큰검증 성공');
            let UserNumber = {
              number : response.data.user.number
            };
            this.user = response.data.user;
            return this.$store.dispatch('MyGetProduct', UserNumber);
          })
          .then( res => {
            console.log('내가 올린 상품 성공 : '+JSON.stringify(res));
            this.Products = res.data.Product;
            console.log('내가 올린 상품 성공 : '+JSON.stringify(this.Products));
          });
      }
      else {
        console.log('내가 올린 상품 실패 : '+JSON.stringify(this.Products));
      }
    }
  },
}
</script>

<style scoped>

```

```
</style>
```

```
<MyPage.vue>
<template>
  <div v-if="user">
    <h1>{{user.name}}님의 마이 페이지</h1>
    <p>잔액 : {{user.money.toLocaleString()}}원</p>
    <hr noshade/>
    <div class="AllProduct">
      <div class="row">
        <div class="col-md-2">
          <table class="table table-hover">
            <tbody>
              <tr class="table-success" v-if="Store.seller == 1">
                <th scope="row" v-on:click="onCategory('all')">
                  상품 관리
                </th>
              </tr>
              <tr class="table-success" v-if="Store.seller == 1">
                <th scope="row" v-on:click="onCategory('Create')">
                  상품 등록
                </th>
              </tr>
              <!--
              <tr class="table-success">
                <th scope="row" v-on:click="onCategory('change')">
                  비밀번호 변경
                </th>
              </tr>
              -->
              <!--
              <tr class="table-success">
                <th scope="row" v-on:click="onCategory('receipt')">
                  영수증
                </th>
              </tr>
              -->
              <tr class="table-success" v-if="storeTF == false">
                <th scope="row" v-on:click="onCategory('cart')">
                  장바구니
                </th>
              </tr>
              <tr class="table-success" >
                <th scope="row" v-if="storeTF == false"
v-on:click="onCategory('order')">
                  주문/구매내역
                </th>
              </tr>
              <tr class="table-success" v-if="storeTF == true">
                <th scope="row" v-on:click="onCategory('sales')">
                  판매 내역
                </th>
              </tr>
            </tbody>
          </table>
        </div>
        <div class="col-md-1"></div>
        <div class="col-md-7">
          <MyCategory v-bind:choice="choiceCategory"></MyCategory>
        </div>
      </div>
    </div>
  </div>
</template>
```

```

< C h a n g e P a s s w o r d
v-bind:choice="choiceCategory"></ChangePassword>
  <CreateStore v-bind:choice="choiceCategory"></CreateStore>
  <Cart v-bind:choice="choiceCategory"></Cart>
  <OrderView v-bind:choice="choiceCategory"></OrderView>
  <SalesHistory v-bind:choice="choiceCategory"></SalesHistory>
</div>
</div>
</div>
</div>
</template>

<script>
import MyCategory from "./MyCategory";
import CreateStore from "./CreateStore";
import ChangePassword from "./ChangePassword";
import OrderView from "./OrderView";
import SalesHistory from "./SalesHistory";
import Cart from "./Cart";
export default {
  name: "MyPage",
  components : {
    SalesHistory,
    OrderView,
    MyCategory,
    CreateStore,
    ChangePassword,
    Cart
  },
  data () {
    return {
      user : {},
      storeTF : null,
      Store : {},
      choiceCategory : ""
    }
  },
  methods : {
    onCategory : function ( select ) {
      this.choiceCategory = select;
      console.log(this.choiceCategory);
    }
  },
  created() {
    this.$store.dispatch('GetProfile')
      .then( response => {
        console.log('토큰검증 성공');
        this.user = response.data.user;
        if( response.data.user.indi == 0 ) {
          let UserNumber = {
            number : response.data.user.number
          };
          return this.$store.dispatch('FoundEnt', UserNumber);
        } else {
          this.storeTF = false;
          console.log('기업 검증 실패');
          //alert('기업 검증 실패');
          //return '기업검증 실패';
        }
      })
      .then( res => {

```

```

        this.Store = res.data.store;
        if (res.data.store.seller == 1) {
            this.storeTF = true;
            this.newStore.seller = res.data.store.company;
            this.newStore.number = res.data.store.number;
            console.log('판매자 검증 성공');
        } else {
            this.storeTF = false;
            console.log('판매자 검증 실패');
            //alert('판매자 검증 실패');
        }
    })
    .catch( err => {
        if(err == '기업검증 실패') {
            this.storeTF = false;
            console.log('검증 실패 err 1');
        }
        else if(err.response.status === 401) {
            this.storeTF = false;
            console.log('검증 실패' + JSON.stringify(err));
            //this.$store.dispatch('LOGOUT');
            //this.$router.replace({path : '/Login'});
        } else {
            this.storeTF = false;
            console.log('검증 실패 err 2');
        }
    })
    }
}
</script>
<style scoped>
</style>

```

```

<Navbar.vue>
<template>
  <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">
    <!--홈 <이미지-->
    <div class="container">
      <router-link to="/" class="navbar-brand float-left">
        COCONUT
      <!--<div class="centered">COCONUT</div-->
    </router-link>

    <button
      class="navbar-toggler"
      type="button"
      data-toggle="collapse"
      data-target="#navbarsExampleDefault"
      aria-controls="navbarsExampleDefault"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarsExampleDefault">
      <ul class="navbar-nav ml-auto">

```



```

        <li class="nav-item" v-if="this.$store.state.pToken">
            <router-link to="/MyPage" class="nav-link jb-default-3">마이페이지
    </router-link>
        </li>
        <li class="nav-item" v-if="this.$store.state.pToken">
            <router-link to="/Cert" class="nav-link jb-default-3">인증서 발급
    </router-link>
        </li>
        <!--
        <li class="nav-item" v-if="this.$store.state.pToken">
            <router-link to="/CertTest" class="nav-link jb-default-3">Cert
Test</router-link>
        </li>
        -->
        <!--
        <li class="nav-item" v-if="this.$store.state.pToken">
            <router-link to="/AdditionalCert" class="nav-link">Additional
Cert</router-link>
        </li>
        -->
        <li class="nav-item">
            <router-link to="/AllProduct" class="nav-link jb-default-3">모든 상
품</router-link>
        </li>
        <!--
        <li class="nav-item">
            <router-link to="/AllStore" class="nav-link">Store</router-link>
        </li>
        -->
        <li class="nav-item">
            <router-link to="/Pay" class="nav-link jb-default-3">QR결제
    </router-link>
        </li>
        <li class="nav-item" v-if="!this.$store.state.pToken">
            <router-link to="/ChoiceMemberType" class="nav-link jb-default-3">회
원가입</router-link>
        </li>
        <li class="nav-item" v-if="!this.$store.state.pToken">
            <router-link to="/Login" class="nav-link jb-default-3">로그인
    </router-link>
        </li>
        <!--
        <li class="nav-item">
            <router-link to="/Test" class="nav-link">Test</router-link>
        </li>
        <li class="nav-item">
            <router-link to="/ImageUpload" class="nav-link">Image</router-link>
        </li>
        -->
        <li class="nav-item" v-if="this.$store.state.pToken">
            <a class="nav-link jb-default-3" href="#"
@click.prevent="onLogoutClick">로그아웃</a>
        </li>
    </ul>
</div>
</div>
</nav>
</template>

<script>
    export default {

```

```

        name: "Navbar",
        methods: {
            onLogoutClick: function() {
                this.$store.dispatch("LOGOUT");
                this.$router.replace({ path: "/Login" });
            }
        }
    };
</script>

<style scoped>
    .centered {position: absolute; left: 45%; bottom: 25%; }
    router-link{
        text-decoration: none;
        color: #333333;
    }
    .jb-default-1 { font-size: 16px; }
    .jb-default-2 { font-size: 22px; }
    .jb-default-3 { font-size: 25px; }
    .jb-smaller { font-size: smaller; }
    .jb-larger { font-size: 40px; }
</style>

```

```

<NotFound.vue>
<template>
    <h1>요청한 페이지는 존재하지 않습니다</h1>
</template>

<script>
    export default {
        name: "NotFound"
    }
</script>

<style scoped>
</style>

```

```

<OrderView.vue>
<template>
    <div class="list-group" v-if="choice == 'order'">
        <h2>주문/구매내역</h2>
        <br>
        <div v-for="order in Orders">
            <a class="list-group-item list-group-item-action flex-column align-items-start">
                <div class="media d-flex w-100 justify-content-between">
                    <router-link :to=""/DetailProduct/'+order.procode" style="color: black" >
                        
                    </router-link>
                    <div class="media-body">
                        <h5
                            class="mt-0"> <router-link
:to=""/DetailProduct/'+order.procode" style="color: black" >{{order.pro}}</router-link>
                            포함 {{order.kind}}종 {{order.qu}}개</h5>

```

```

        <!--
        <h6>주문자 : {{order.orderer}}</h6>
        <h6 v-if="order.paid == 1">결제자 : {{order.buyer}}</h6>
        -->
    </div>
    <strong class="text-black" style="color: #495057">총 결제금액 :
    {{order.price.toLocaleString()}}원
    <br>
    <router-link :to="/DetailOrder/"+order.order_no" style="color:
    black"><small class="float-right">주문상세보기</small></router-link></strong>
    </div>
</a>
<a class="list-group-item list-group-item-action flex-column align-items-start">
    <div class="d-flex w-100 justify-content-between">
        <h6 class="mb-1">
            주문번호 :
            <router-link :to="/DetailOrder/"+order.order_no" style="color:
            black">{{order.order_no}}</router-link>
        </h6>
        <strong style="color: crimson;" v-if="order.paid == 1">결제 완료
    </strong>
        <strong style="color: black;" v-if="order.paid == 0">결제 대기
    </strong>
    </div>
    </a>
    <br>
</div>
</div>
</template>

<script>
    export default {
        name: "OrderView",
        props: {
            choice : ""
        },
        data () {
            return {
                user : {},
                Orders : [],
                temp : [],
                Products : [],
                kind : [ 0 ],
                allprice : 0,
                choiceType : null,
                allow : false,
                pcode : "",
                pquan : [],
                seller : "",
                pnum : {
                    orderno : this.$route.params.order
                },
                //lnk : 'http://localhost:3000/img/',
                lnk : "/img/"
            }
        },
        methods : {
        },
        watch : {
            choice : function (category) {
                if ( category == 'order') {

```

```

this.$store.dispatch('GetProfile')
.then( response => {
  console.log('토큰검증 성공');
  let UserNumber = {
    number : response.data.user.number
  };
  this.user = response.data.user;
  return this.$store.dispatch('GetOrder_3', this.user);
})
.then( response => {
  console.log('get order : '+JSON.stringify(response.data));
  var orders = response.data.order;
  var sortingField = "order_no";
  /*
  orders.sort(function(a, b) { // 오름차순
    return a[sortingField] - b[sortingField];
  });
  */
  orders.sort(function(a, b) { // 내림차순
    return b[sortingField] - a[sortingField];
  });
  this.temp = orders;
  var p;
  var p1;
  var p2 = new Array;
  var p3 = new Array;
  for (var j=0; j<orders.length; j++){
    p = response.data.order[j].product;
    p1 = new Array;
    p1 = p.split(',');
    for (var i=0; i<p1.length; i++) {
      p2.push(p1[i].split('/'));
    }
    //console.log('p2 : '+JSON.stringify(p2));
  }
  for (var i=0; i<p2.length; i++){
    p3.push(p2[i][0])
  }
  //console.log('p3 : '+JSON.stringify(p3));
  var uniq = p3.reduce(function(a,b){
    if (a.indexOf(b) < 0 ) a.push(b);
    return a;
  },[]);
  //console.log(uniq, p3);
  let findproducts = {
    products : uniq
  };
  return this.$store.dispatch('GetProductDetail4', findproducts);
})
.then( response => {
  console.log('response 2 : '+JSON.stringify(response.data.result));
  var p;
  var p1;
  var p2;
  var p3;
  //console.log('temp length : '+this.temp.length);
  for (var i=0; i<this.temp.length; i++){
    p = this.temp[i].product;
    p1 = new Array;
    p2 = new Array;
    p1 = p.split(',');

```

```

        for (var j=0; j<p1.length; j++) {
            p2.push(p1[j].split('/'));
        }
        //console.log('1. p2 : '+JSON.stringify(p2));
        for (var l=0; l<response.data.result.length; l++){
            //console.log('2. p2[0][0] : '+p2[0][0]);
            // console.log ( ' 2 .
response.data.result['+l+'].productcode : '+response.data.result[l].productcode);
            if (p2[0][0] == response.data.result[l].productcode)
{
                var qu = 0;
                for (var m=0; m<p2.length; m++){
                    p2[m][1] *= 1;
                    qu += p2[m][1];
                    //console.log('qu : '+qu);
                }
                this.temp[i].kind = p1.length;
                this.temp[i].qu = qu;
                this.temp[i].procode =
response.data.result[l].productname;
                this.temp[i].procode =
response.data.result[l].productcode;
                this.temp[i].img =
this.lnk+response.data.result[l].thumbnail;
                //console.log('실행됨');
                break;
            }
        }
        //console.log('temp : '+JSON.stringify(this.temp));
        this.Orders = this.temp;
    })
    .catch( err => {
        //alert('잘못된 요청입니다. ');
        console.log('잘못된 요청입니다. : '+err);
        //this.$router.replace({ path : '/' });
    });
}
else {
    console.log('주문내역 실패 : '+JSON.stringify(this.Products));
}
}
}
}
}
</script>
<style scoped>
    .widthSet {
        max-width: 80px;
    }
    .heightSet {
        max-height: 80px;
    }
    .jb-default-1 { font-size: 16px; }
    .jb-default-2 { font-size: 22px; }
    .jb-default-3 { font-size: 25px; }
    .jb-smaller { font-size: smaller; }
    .jb-larger { font-size: 40px; }
</style>

```

```

<Pay.vue>
<template>
  <div>
    <form @submit="PaySubmit">
      <p class="error">{{ error }}</p>
      <!--<p class="decode-result">Last result: <b>{{ result }}</b></p>-->
      <qr-code-stream v-if="!allow == true" @decode="onDecode" @init="onInit" />
      <div class="row" v-if="allow == true">
        <div class="col-md-2"></div>
        <div class="col-md-8">
          <h2>주문 정보 확인</h2>
          <hr noshade/>
          <table class="table">
            <thead>
              <tr class="table-active">
                <th scope="col">상품정보</th>
                <th scope="col">금액</th>
                <th scope="col">수량</th>
                <th scope="col">판매자</th>
              </tr>
            </thead>
            <tbody v-for="Pro in Products">
              <tr>
                <td>
                  <div class="media">
                    
                    <div class="media-body">
                      <h5 class="mt-0">{{Pro.productname}}</h5>
                      <h6 class="text-muted">
                        {{Pro.description}}<br>
                        {{Pro.category}}
                      </h6>
                    </div>
                  </div>
                </td>
                <td><h5>{{(Pro.oquantity*Pro.price).toLocaleString()}}원
</h5></td>
                <td><h5>{{Pro.oquantity}}개</h5></td>
                <td><h5>{{Pro.seller}}</h5></td>
              </tr>
            </tbody>
          </table>
          <hr noshade/>
          <div class="card">
            <div class="card-body">
              <div class="row">
                <h6 class="col-md-3"></h6>
                <h6 class="col-md-3">총 주문 상품수</h6>
                <h6 class="col-md-3">{{kind[0]}}종 {{kind[1]}}개</h6>
                <h6 class="col-md-3"></h6>
              </div>
              <hr class="my-4">
              <div class="row">
                <h6 class="col-md-3"></h6>
                <h6 class="col-md-3">총 결제 예상 금액</h6>
                <h5 class="col-md-4">{{allprice.toLocaleString()}}원</h5>
                <h6 class="col-md-2"></h6>
              </div>
            </div>
          </div>
        </div>
      </form>
    </div>
  </template>
</div>

```

```

        </div>
        <br>
        <div class="row">
            <div class="col-md-6">
                <button @click="onResetSubmit" type="button" class="btn
btn-lg btn-success col-md-12">QR코드 재인식 </button>
            </div>
            <br>
            <div class="col-md-6">
                <button @click="nomalChoice" type="button" class="btn
btn-lg btn-primary col-md-12">결제 </button>
                <div v-if="choiceType==true">
                    <hr noshade/>
                    <div class="alert alert-warning" role="alert">
                        <h3 class="page-header">인증서 비밀번호 입력</h3>
                        <div class="form-group">
                            <label>Password</label>
                            <input type="password" class="form-control"
v-model="Cpass" name="password">
                        </div>
                        <button @click="Trade" type="button" class="btn
btn-primary align-self-center">결제 </button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</form>
</div>
</template>

<script>
export default {
  name: "Pay",
  data() {
    return {
      choiceType : null,
      allow : false,
      Products : [],
      kind : [ 0 ],
      allprice : 0,
      Cpass : "",
      user : {},
      order : {},
      pcode : "",
      pquan : [],
      seller : "",
      result : {},
      ordernumber : "",
      error : "",
      //lnk : 'http://localhost:3000/img/',
      lnk : "/img/"
    }
  },
  methods: {
    Trade : function () {
      var ornum = (this.ordernumber).split("/");
      let certR = {
        pa : this.Cpass,
        id : this.user.id,

```

```

        unum : this.user.number,
        order : this.order,
        order_no : ornum[0]
    };
    this.$store.dispatch('TradeRequest', certR)
    .then( (response) => {
        alert('결제 완료');
        console.log('TradeRequest Success : '+JSON.stringify(response));
        var p = response.data.order;
        var p1 = p.split('/');
        this.$router.replace({ path : '/PurchaseSuccess/'+p1[0] });
    }).catch( err => {
        console.log('TradeRequest Err : '+ err);
        alert('TradeRequest Success : '+err);
    });
},
nomalChoice : function () {
    this.choiceType = true;
},
imglnk : function () {
    for (var i=0; i<(this.Products.length); i++) {
        this.Products[i].thumbnail = this.lnk+this.Products[i].thumbnail;
    }
},
onResetSubmit : function() {
    this.allow = false;
    this.choiceType = null;
},
onDecode (ordernumber) {
    this.ordernumber = ordernumber;
},
async onInit (promise) {
    try {
        await promise
    } catch (error) {
        if (error.name === 'NotAllowedError') {
            this.error = "ERROR: 카메라 허용을 해주십시오. "
        } else if (error.name === 'NotFoundError') {
            this.error = "ERROR: 카메라가 없습니다. "
        } else if (error.name === 'NotSupportedError') {
            this.error = "ERROR: 보안 프로토콜 미사용 (HTTPS, localhost)"
        } else if (error.name === 'NotReadableError') {
            this.error = "ERROR: 카메라 이미 사용중"
        } else if (error.name === 'OverconstrainedError') {
            this.error = "ERROR: 유효하지 않은 카메라"
        } else if (error.name === 'StreamApiNotSupportedError') {
            this.error = "ERROR: 브라우저에서 지원하지 않는 스트림 API"
        }
    }
},
PaySubmit() {
    let number = "";
    let t = "";
    let exit = false;
    let i = 0;//문자열에 한글자씩 접근하는 인덱스
    let j = 0;//0이면 주문번호, 1이면 jwt
    while(exit) {
        if(this.ordernumber.charAt(i)) {
            if(j==0) {
                number = number + this.ordernumber.charAt(i);
            }
        }
    }
}

```



```

        else {
            t = t + this.ordernumber.charAt(i);
        }
    }
}
const Payinfo = {
    order_no : number,
    t : t
};
alert('payinfo : ' + JSON.stringify(Payinfo));
this.$store.dispatch('PAY', Payinfo)
    .then(response => {
        console.log(response);
    })
}
},
watch : {
    ordernumber : function (order) {
        let ordernum = {
            orderno : order
        };
        this.$store.dispatch('GetProfile')
            .then( response => {
                console.log('토큰검증 성공');
                this.user = response.data.user;
                return this.$store.dispatch('GetOrder_2', ordernum)
            })
            .then( response => {
                //alert('주문내역 성공 : ' + JSON.stringify(response.data.order[0]));
                this.order = response.data.order[0];
                //alert('주문내역 : ' + JSON.stringify(response.data.order[0]));
                var p = response.data.order[0].product;
                var p1 = p.split(',');
                var p2 = new Array;
                var p3 = new Array;
                for (var i=0; i<p1.length; i++) {
                    p2.push(p1[i].split('/'));
                    p3.push(p2[i][1]);
                }
                console.log('p2 : ' + JSON.stringify(p2));
                console.log('p3 : ' + JSON.stringify(p3));
                this.pquan = p2;
                let pcode = {
                    productcode : response.data.order[0].product
                };
                return this.$store.dispatch('GetProductDetail2', pcode);
            })
            .then( response => {
                console.log('product detail : ' + JSON.stringify(response.data));
                let pp = response.data.result;
                for (let i=0; i<pp.length; i++) {
                    for(let j=0; j<pp.length; j++) {
                        this.pquan[i][j] *= 1;
                        console.log('pp '+i+' : '+pp[i].productcode);
                        console.log('pquan '+j+' : '+ this.pquan[j][0]);
                        if(pp[i].productcode == this.pquan[j][0]) {
                            pp[i].oquantity = this.pquan[j][1];
                            console.log('일치');
                        } else {
                            console.log('불일치');
                        }
                    }
                }
            })
    }
}

```

```

        }
    }
    this.Products = pp;
    this.seller = response.data.result[0].seller;
    this.imgInk();
    //this.quantityAppend(response.data.result);
    this.allow = true;
    //alert('product detail : '+JSON.stringify(response.data));
    console.log('product detail2 : '+JSON.stringify(this.Products));
})
.finally( () => {
    this.kind[0] = this.Products.length;
    this.kind[1] = 0;
    for (let i=0; i<this.Products.length; i++){
        this.allprice = this.allprice + this.Products[i].oquantity *
this.Products[i].price;
        this.Products[i].oquantity *= 1; //스트링을 정수로 형변환
        this.kind[1] = this.kind[1] + this.Products[i].oquantity;
    }
})
.catch( err => {
    console.log('주문내역 실패 : ' + err);
    //alert(err);
})
    }
}
}
</script>
<style scoped>
    .widthSet {
        max-width: 80px;
    }
    .heightSet {
        max-height: 80px;
    }
    /*
        Max width before this PARTICULAR table gets nasty. This query will take effect for
        any screen smaller than 760px and also iPads specifically.
        */
    @media
    only screen
    and (max-width: 768px), (min-device-width: 768px)
    and (max-device-width: 1024px) {
        /* Force table to not be like tables anymore */
        table, thead, tbody, th, td, tr {
            display: block;
        }
        /* Hide table headers (but not display: none;, for accessibility) */
        thead tr {
            position: absolute;
            top: -9999px;
            left: -9999px;
        }
        tr {
            margin: 0 0 1rem 0;
        }
        tr:nth-child(odd) {
            background: #ccc;
        }
        td {
            /* Behave like a "row" */

```

```

border: none;
border-bottom: 1px solid #eee;
position: relative;
padding-left: 50%;
}
td:before {
  /* Now like a table header */
  position: absolute;
  /* Top/left values mimic padding */
  top: 0;
  left: 6px;
  width: 45%;
  padding-right: 10px;
  white-space: nowrap;
}
/*

```

Label the data

You could also use a data-* attribute and content for this. That way "bloats" the HTML, this way means you need to keep HTML and CSS in sync. Lea Verou has a clever way to handle with text-shadow.

```

*/
td:nth-of-type(1):before { content: "상품정보"; }
td:nth-of-type(2):before { content: "금액"; }
td:nth-of-type(3):before { content: "수량"; }
td:nth-of-type(4):before { content: "판매자"; }
}
</style>

```

```

<PurchaseSuccess.vue>
<template>
  <div class="row" v-if="(order.paid == 1) && (user.number == order.buyer)">
    <div class="col-md-2"></div>
    <div class="col-md-8">
      <h2>결제 완료</h2>
      <hr noshade/>
      <table class="table">
        <thead>
          <tr class="table-active">
            <th style="width: 350px;">결제 금액</th>
            <th>결제일</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td><h5>{{order.price.toLocaleString()}}원</h5></td>
            <td>
              <h5>{{date.getFullYear()}}-{{("0" + (date.getMonth() + 1)).slice(-2)}}-{{("0" + (date.getDate() + 1)).slice(-2)}}
              /
              {{("0" + (date.getHours() + 1)).slice(-2)}}:{{("0" + (date.getMinutes() + 1)).slice(-2)}}:{{("0" + (date.getSeconds() + 1)).slice(-2)}}</h5>
            </td>
          </tr>
        </tbody>
      </table>
      <hr noshade/>
      <table class="table">

```

```

<thead>
<tr class="table-active">
  <th>구매자</th>
  <th>주문자</th>
</tr>
</thead>
<tbody>
<tr>
  <td style="width: 350px;">{{user.name}}</td>
  <td>{{orderer}}</td>
</tr>
</tbody>
<thead>
<tr class="table-active">
  <th>배송 주소</th>
  <th>주문자 전화번호</th>
</tr>
</thead>
<tbody>
<tr>
  <td style="width: 350px;">{{order.delivery_address}}</td>
  <td>{{order.delivery_tel}}</td>
</tr>
</tbody>
<thead>
<tr class="table-active">
  <th>주문번호</th>
</tr>
</thead>
<tbody>
<tr>
  <td>{{order.order_no}}</td>
</tr>
</tbody>
</table>
<br>
</div>
</div>
</template>

<script>
export default {
  name: "PurchaseSuccess",
  data() {
    return {
      choiceType : null,
      allow : false,
      Products : [],
      kind : [ 0 ],
      allprice : 0,
      user : {},
      order : {},
      orderer : "",
      pcode : "",
      ordernumber : this.$route.params.order,
      error : "",
      date : ""
    }
  },
  created() {
    this.$store.dispatch('GetProfile')
  }
}

```

```

        .then( response => {
            console.log('토큰검증 성공');
            this.user = response.data.user;
            let ordernum = {
                orderno : this.ordernumber
            };
            return this.$store.dispatch('GetOrder', ordernum);
        })
        .then( response => {
            console.log('get order : ' + JSON.stringify(response.data));
            this.order = response.data.order[0];
            this.date = new Date((this.order.trade_time * 1));
            let usernum = {
                number : this.order.orderer
            };
            return this.$store.dispatch('FindUsername', usernum);
        })
        .then( response => {
            console.log('response : ' + JSON.stringify(response.data));
            this.orderer = response.data.result[0].name;
        })
        .catch( err => {
            alert('잘못된 요청입니다.');
```

//this.\$router.replace({ path : '/' });

```

        });
    }
}
</script>

<style scoped>
</style>

```

```

<RegisterEnterpriseBuyer.vue>
<template>
    <div class="row">
        <div class="col-md-3"></div>
        <div class="col-md-6">
            <h2>사업자 구매 회원 가입입니다</h2>
            <br><br>
            <input type="text" v-model="newUser.id" class="form-control"
placeholder="아이디">
            <small id="idHelp" class="form-text text-muted float-left">아이디는 6글자
이상이어야 합니다.</small>
            <br><br>
            <input type="password" v-model="newUser.password" class="form-control"
placeholder="비밀번호">
            <small id="passwordHelp" class="form-text text-muted float-left">비밀번호
는 8글자 이상, 영문 대/소문자, 숫자, 특수문자가 포함되어야 합니다.</small>
            <br><br>
            <input type="text" v-model="newUser.name" class="form-control"
placeholder="이름">
            <br>
            <input type="text" v-model="newUser.tel" class="form-control"
placeholder="전화번호">
            <br>
            <input type="text" v-model="newUser.addr" class="form-control"
placeholder="주소">
            <br>
        </div>
    </div>

```

```

        <input type="text" v-model="newUser.email" class="form-control"
placeholder="이메일 abc@example.com">
        <br>
        <input type="number" v-model="newUser.crn" class="form-control"
placeholder="사업자 등록번호">
        <small id="crnHelp" class="form-text text-muted float-left">사업자 번호 10
자리를 입력해주세요.</small>
        <br> <br>
        <input type="text" v-model="newUser.company" class="form-control"
placeholder="상호명">
        <br> <br>
        <button @click="newUserSubmit" type="button" class="btn btn-primary">
회원가입</button>
    </div>
    <div class="col-md-3"> </div>
</div>
</template>

<script>
var indi = 0;
var seller = 0;
export default {
  name: "RegisterEnterpriseBuyer",
  data(){
    return {
      newUser : {
        id: "",
        password: "",
        name: "",
        tel: "",
        addr: "",
        email: "",
        indi: indi,
        crn: "",
        company: "",
        seller: seller
      }
    }
  },
  methods : {
    newUserSubmit : function () {
      let pathuser = {
        path : '/users/registerEnt',
        user : this.newUser
      };
      this.$store.dispatch('REGISTER', pathuser)
        .then( response => {
          if(response.data.success == true) {
            alert('Enterprise user register Success');
            console.log('Enterprise user register Success');
            this.$router.replace({ path : '/Login' });
          } else {
            alert('Enterprise user register Failure');
            console.log('Enterprise user register Failure')
          }
        })
        .catch( err => {
          console.log('Enterprise user register Err : '+ err);
        });
    }
  }
}

```

```
</script>
```

```
<style scoped>  
</style>
```

```
<RegisterEnterpriseSeller.vue>  
<template>  
  <div class="row">  
    <div class="col-md-3"> </div>  
    <div class="col-md-6">  
      <h2>사업자 판매 회원가입입니다</h2>  
      <br><br>  
      <input type="text" v-model="newSeller.id" class="form-control"  
placeholder="아이디">  
      <small id="idHelp" class="form-text text-muted float-left">아이디는 6글자  
이상이어야 합니다.</small>  
      <br><br>  
      <input type="password" v-model="newSeller.password" class="form-control"  
placeholder="비밀번호">  
      <small id="passwordHelp" class="form-text text-muted float-left">비밀번호  
는 8글자 이상, 영문 대/소문자, 숫자, 특수문자가 포함되어야 합니다.</small>  
      <br><br>  
      <input type="text" v-model="newSeller.name" class="form-control"  
placeholder="이름">  
      <br>  
      <input type="text" v-model="newSeller.tel" class="form-control"  
placeholder="전화번호">  
      <br>  
      <input type="text" v-model="newSeller.addr" class="form-control"  
placeholder="주소">  
      <br>  
      <input type="text" v-model="newSeller.email" class="form-control"  
placeholder="이메일 abc@example.com">  
      <br>  
      <input type="text" v-model="newSeller.crn" class="form-control"  
placeholder="사업자 등록 번호">  
      <small id="crnHelp" class="form-text text-muted float-left">사업자 번호 10  
자리를 입력해주세요</small>  
      <br><br>  
      <input type="text" v-model="newSeller.company" class="form-control"  
placeholder="회사이름">  
      <br><br>  
      <button @click="newSellerSubmit" type="button" class="btn btn-primary">  
회원가입</button>  
    </div>  
    <div class="col-md-3"> </div>  
  </div>  
</template>  
  
<script>  
  var indi = 0;  
  var seller = 1;  
  export default {  
    name: "RegisterEnterpriseSeller",  
    data(){  
      return {  
        newSeller : {  
          id: "",
```

```

        password: "",
        name: "",
        tel: "",
        addr: "",
        email: "",
        crn: "",
        company: "",
        indi: indi,
        seller: seller
    }
},
methods : {
    newSellerSubmit : function () {
        let pathuser = {
            path : '/users/registerEnt',
            user : this.newSeller
        };
        this.$store.dispatch('REGISTER', pathuser)
        .then( response => {
            if(response.data.success == true) {
                alert('Enterprise Seller user register Success');
                console.log('Enterprise Seller user register Success');
                this.$router.replace({ path : '/Login' });
            } else {
                alert('Enterprise Seller user register Failure');
                console.log('Enterprise Seller user register Failure')
            }
        }).catch( err => {
            console.log('Enterprise Seller user register Err : '+ err);
        });
    }
}
}
</script>
<style scoped>
</style>

```

```

<RegisterIndividual.vue>
<template>
    <div class="row">
        <div class="col-md-3"></div>
        <div class="col-md-6">
            <h2>개인 구매 회원가입</h2>
            <br><br>
            <input type="text" v-model="newUser.id" class="form-control"
placeholder="아이디">
            <small id="idHelp" class="form-text text-muted float-left">아이디는 6글자
이상이어야 합니다.</small>
            <br><br>
            <input type="password" v-model="newUser.password" class="form-control"
placeholder="비밀번호">
            <small id="passwordHelp" class="form-text text-muted float-left">비밀번호
는 8글자 이상, 영문 대/소문자, 숫자, 특수문자가 포함되어야 합니다.</small>
            <br><br>
            <input type="text" v-model="newUser.name" class="form-control"

```



```

placeholder="이름">
    <br>
    <input type="text" v-model="newUser.tel" class="form-control"
placeholder="전화번호">
    <br>
    <input type="text" v-model="newUser.addr" class="form-control"
placeholder="주소">
    <br>
    <input type="text" v-model="newUser.email" class="form-control"
placeholder="이메일 abc@example.com">
    <br><br>
    <button @click="newUserSubmit" type="button" class="btn btn-primary">
회원가입</button>
</div>
<div class="col-md-3"></div>
</div>
</template>

<script>
var indi = 1;
export default {
  name: "RegisterIndividual",
  data(){
    return {
      newUser : {
        id: "",
        password: "",
        name: "",
        tel: "",
        addr: "",
        email: "",
        indi: indi
      }
    }
  },
  methods : {
    newUserSubmit : function () {
      let pathuser = {
        path : '/users/register',
        user : this.newUser
      };
      this.$store.dispatch('REGISTER', pathuser)
        .then( response => {
          if(response.data.success == true) {
            alert('Individual user register Success');
            console.log('Individual user register Success');
            this.$router.replace({ path : '/Login' });
          } else {
            alert('Individual user register Failure');
            console.log('Individual user register Failure')
          }
        })
        .catch( err => {
          console.log('Individual user register Err : '+ err);
        });
    }
  }
}
</script>

<style scoped>
</style>

```

```

<SalesHistory.vue>
<template>
  <div class="list-group" v-if="choice == 'sales'">
    <h2>판매내역</h2>
    <br>
    <div v-for="order in Orders">
      <a class="list-group-item list-group-item-action flex-column align-items-start"
v-for="ppp in order.pp">
        <div class="media d-flex w-100 justify-content-between">
          <router-link :to="/DetailProduct/"+order.procode" style="color: black" >
            
          </router-link>
          <div class="media-body">
            <h5 class="mt-0"><router-link :to="/DetailProduct/" + ppp.pcode"
style="color: black" >{{ppp.pname}}</router-link>
              {{ppp.pquantity}}개</h5>
            <h6 class="text-muted">{{ppp.pdescription}}</h6>
            <h6 class="text-muted">{{ppp.pcategory}}</h6>
            <!--
            <h6>주문자 : {{order.orderer}}</h6>
            <h6 v-if="order.paid == 1">결제자 : {{order.buyer}}</h6>
            -->
          </div>
          <strong class="text-black" style="color: #495057">상품 금액 :
{{{ppp.pquantity * ppp.pprice}.toLocaleString()}}원
            <br>
            <router-link :to="/DetailOrderSell/"+order.order_no" style="color:
black"><small class="float-right">주문상세보기</small> </router-link> </strong>
          </div>
        </a>
        <a class="list-group-item list-group-item-action flex-column align-items-start">
          <div class="d-flex w-100 justify-content-between">
            <h6 class="mb-1">
              주문번호 :
              <router-link :to="/DetailOrderSell/"+order.order_no" style="color:
black" >{{order.order_no}}</router-link>
            </h6>
            <strong style="color: crimson;" v-if="order.paid == 1">결제 완료
          </strong>
            <strong style="color: black;" v-if="order.paid == 0">결제 대기
          </strong>
          </div>
        </a>
      <br>
    </div>
  </div>
</template>

<script>
export default {
  name: "SalesHistory",
  props: {
    choice : ""
  },
  data () {
    return {
      user : {},
      Orders : [],
      temp : [],
      temp2 : [],
    }
  }
}

```

```

        Products : [],
        kind : [ 0 ],
        allprice : 0,
        choiceType : null,
        allow : false,
        pcode : "",
        pquan : [],
        seller : "",
        //lnk : 'http://localhost:3000/img/',
        lnk : "/img/"
    }
},
methods : {
},
watch : {
    choice : function (category) {
        if ( category == 'sales') {
            this.$store.dispatch('GetProfile')
                .then( response => {
                    console.log('토큰검증 성공');
                    let UserNumber = {
                        number : response.data.user.number
                    };
                    this.user = response.data.user;
                    return this.$store.dispatch('MyGetProduct2', UserNumber);
                })
                .then( response => {
                    //console.log('response 1 : '+JSON.stringify(response));
                    let aaa = {
                        products : []
                    };
                    for (let i = 0; i < response.data.Product.length; i++) {
                        aaa.products.push(response.data.Product[i].productcode);
                    }
                    //console.log('aaa.products : '+aaa.products);
                    this.temp2 = response.data.Product;
                    /*
                    var orders = response.data.order;
                    var sortingField = "order_no";
                    orders.sort(function(a, b) { // 내림차순
                        return b[sortingField] - a[sortingField];
                    });
                    this.temp = orders;
                    let seller_number = {
                        seller : this.user.id,
                        number : this.user.number
                    };
                    */
                    return this.$store.dispatch('GetOrder_4', aaa);
                })
                .then( response => {
                    //console.log('response 2 : '+JSON.stringify(response));
                    //let myProduct = response.data.Product;
                    //console.log("all : "+JSON.stringify(response.data.order));
                    //console.log("[0] : "+JSON.stringify(response.data.order[0]));
                    //console.log("[1] : "+JSON.stringify(response.data.order[1]));
                    //console.log("[0][0]
"+JSON.stringify(response.data.order[0][0]));
                    //console.log("[1][0]
"+JSON.stringify(response.data.order[1][0]));
                    let arr = new Array;

```

```

for (let j=0; j<response.data.order.length; j++){
  for (let k=0; k<response.data.order[j].length; k++){
    arr.push(response.data.order[j][k]);
  }
}
//console.log('arr : '+JSON.stringify(arr));
var sortingField = "order_no";
/*
orders.sort(function(a, b) { // 오름차순
  return a[sortingField] - b[sortingField];
});
*/
arr.sort(function(a, b) { // 내림차순
  return b[sortingField] - a[sortingField];
});
let uniq = arr.reduce(function(a, b){ //중복제거
  //console.log("a : "+JSON.stringify(a));
  //console.log("b : "+JSON.stringify(b));
  if (JSON.stringify(a).indexOf(JSON.stringify(b)) < 0 ) {
    //console.log("들어옴1");
    a.push(b);
  }
  return a;
},[]);
/*
let uniq = arr.slice() // 정렬하기 전에 복사본을 만든다.
.sort(function(a, b){
  return b[sortingField] - a[sortingField];
})
.reduce(function(a, b){
  if (a.slice(-1)[0] !== b) a.push(b); // slice(-1)[0] 을 통해
  return a;
},[]); //a가 시작될 때를 위한 비어있는 배열
*/
/*
for (var i=0; i<this.temp.length; i++){
  //여기서 temp의 주문정보들이랑 내상품들을 비교해서
  temp에 상품정보를 추가하고 보이도록 하기
}
*/
//console.log('uniq : '+JSON.stringify(uniq));
this.temp = uniq;
var p;
var p1;
var p2;
var p3;
//console.log('temp length : '+this.temp.length);
for (let z=0; z<this.temp.length; z++) {
  this.temp[z].pp = [];
  p = this.temp[z].product;
  p1 = new Array;
  p2 = new Array;
  p1 = p.split(',');
  //console.log('p : ' + p);
  //console.log('p1 : ' + p1.length);
  for (let x = 0; x < p1.length; x++) {
    p2.push(p1[x].split('/'));
  }
  //console.log('1. p2 : ' + JSON.stringify(p2));
  for (let f=0; f<p2.length; f++) {

```

```

//console.log("f 반복문 f : "+f);
for (let q=0; q<this.temp2.length; q++){
    //console.log('2. p2['+f+'][0] : ' + p2[f][0]);
    //console.log('2. this.temp2[' + q + '].productcode
: ' + this.temp2[q].productcode);

    if (p2[f][0] == this.temp2[q].productcode) {
        //console.log("들어옴2");
        let proArr = {
            pcode : this.temp2[q].productcode,
            pprice : this.temp2[q].price,
            pname : this.temp2[q].productname,
            pdescription : this.temp2[q].description,
            pcategory : this.temp2[q].category,
            pname : this.temp2[q].productname,
            pthumbnail : this.lnk +
this.temp2[q].thumbnail,
            pquantity : p2[f][1]
        };
        //console.log("proArr
        :

"+JSON.stringify(proArr));

        this.temp[z].pp.push(proArr);
        //console.log("this.temp
        :

"+JSON.stringify(this.temp));

        break;
    } else {
        //console.log("못들어옴2");
    }
}
}
}
/*
console.log('temp length : '+this.temp.length);
for (var z=0; z<this.temp.length; z++){
    p = this.temp[z].product;
    p1 = new Array;
    p2 = new Array;
    p1 = p.split(',');
    console.log('p : '+p);
    console.log('p1 : '+p1.length);
    for (var x=0; x<p1.length; x++) {
        p2.push(p1[x].split('/'));
    }
    console.log('1. p2 : '+JSON.stringify(p2));
    for (var f=0; f<this.temp2.length; f++){
        console.log('2. p2[0][0] : '+p2[0][0]);
        console.log('2. this.temp2['+f+'].productcode
        :

'+this.temp2[f].productcode);

        if (p2[0][0] == this.temp2[f].productcode) {
            var qu = 0;
            for (var m=0; m<p2.length; m++){
                p2[m][1] *= 1;
                qu += p2[m][1];
                //console.log('qu : '+qu);
            }
            this.temp[z].kind = p1.length;
            this.temp[z].qu = qu;
            this.temp[z].pro = this.temp2[f].productname;
            this.temp[z].procode
            =

this.temp2[f].productcode;

            this.temp[z].img
            =

this.lnk+this.temp2[f].thumbnail;

```

```

                                console.log('실행됨');
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
    /*
    //console.log('temp : '+JSON.stringify(this.temp));
    this.Orders = this.temp;
    })
    .catch( err => {
        //alert('잘못된 요청입니다. ');
        console.log('잘못된 요청입니다. : '+err);
        //this.$router.replace({ path : '/' });
    });
    }
    else {
        console.log('판매내역 실패 : '+JSON.stringify(this.Products));
    }
    }
}
}
}
</script>
<style scoped>
    .widthSet {
        max-width: 80px;
    }
    .heightSet {
        max-height: 80px;
    }
    .jb-default-1 { font-size: 16px; }
    .jb-default-2 { font-size: 22px; }
    .jb-default-3 { font-size: 25px; }
    .jb-smaller { font-size: smaller; }
    .jb-larger { font-size: 40px; }
</style>

```

5.3 발표 PPT 자료

QR코드를 이용한 간편 결제 시스템 (시스템명 : Coconut Pay)

2019. 10. 15



지도 교수 : 이병천 교수님

1조 박종훈
권희민
마민기
윤정민

1

목차



1. 조원 편성
2. 주제 선정
3. 구 상 도
4. 추진 경과
5. 개발 환경 및 개발내용
6. 개발 시스템 운영
7. 결론 및 기대효과

1

조원 편성



이름	역할
박종훈	데이터베이스 설계, 구축 및 관리, PPT 작성 [프로젝트 총괄]
권혁민	프론트엔드 설계 및 구현, 보고서 작성
마민기	백엔드 설계 및 구현, 보고서 작성
윤정민	데이터베이스 설계, 구축 및 관리, PPT 작성

1

주제 선정(1/2)



□ 전세계적으로 QR코드의 활용도가 높아지는 추세



□ 우리 정부도 정책적으로 QR코드 활성화



공인인증서와 액티브X를 이용한 결제의 불편함에 대한 불만 폭증

1

주제 선정(2/2)



□ 스마트폰 카메라로 QR코드 촬영 후
즉시 결제 가능

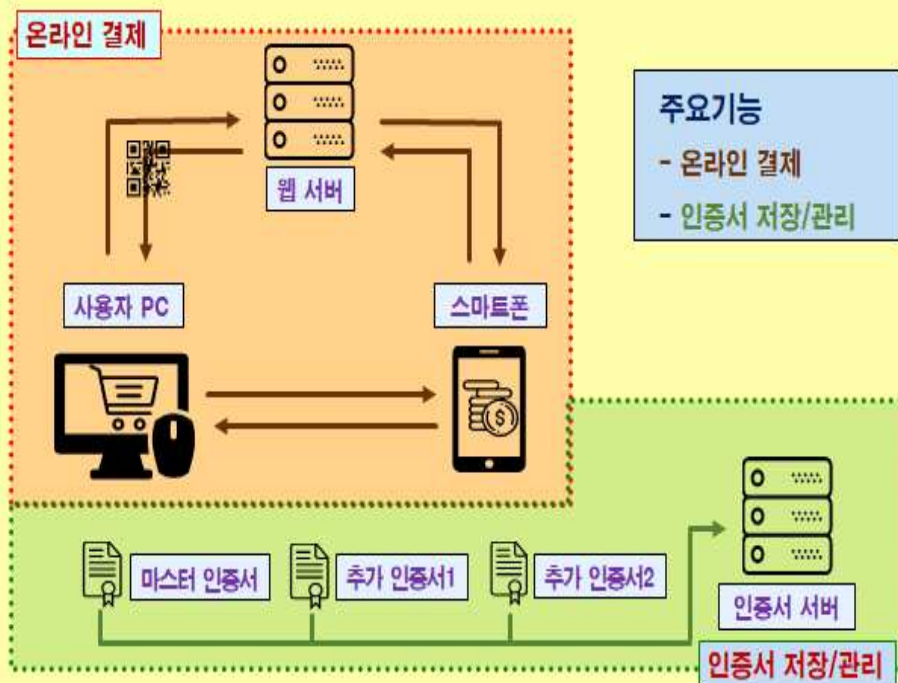
□ 보안/기기 관리를 위한 마스터인증서/추가인증서 관리



QR코드를 이용한 간편 결제시스템 구축

1

구상도(1/4)

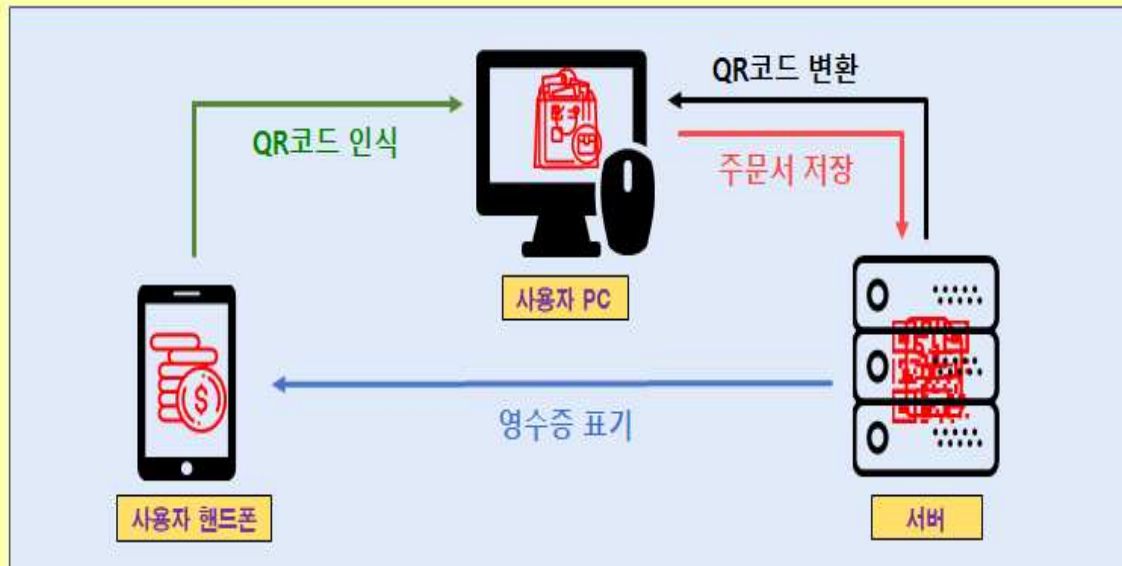


6

구상도(2/4)



온라인 구매

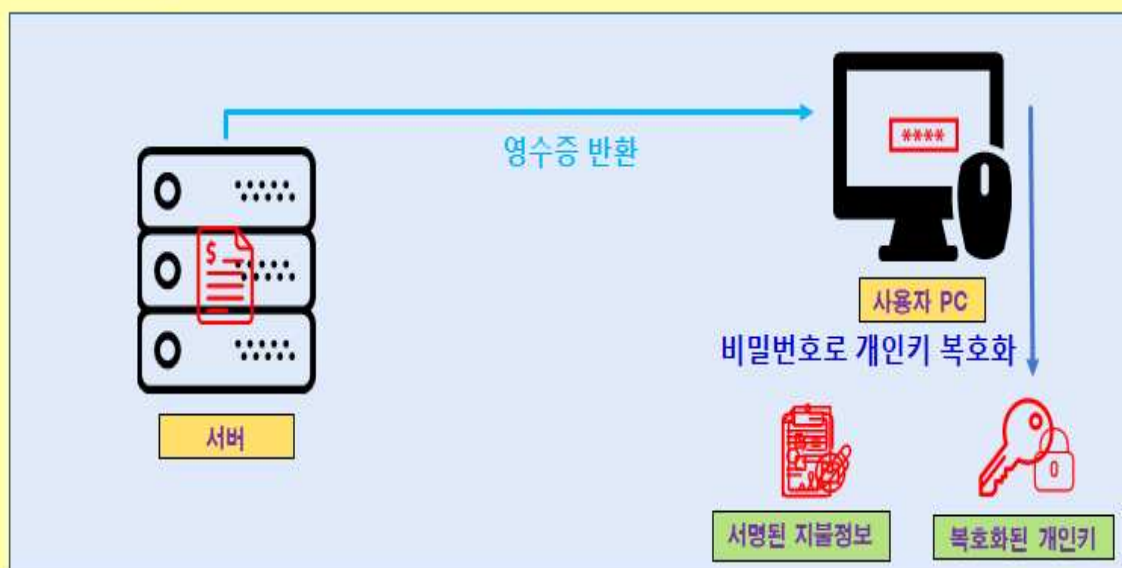


7

구상도(3/4)

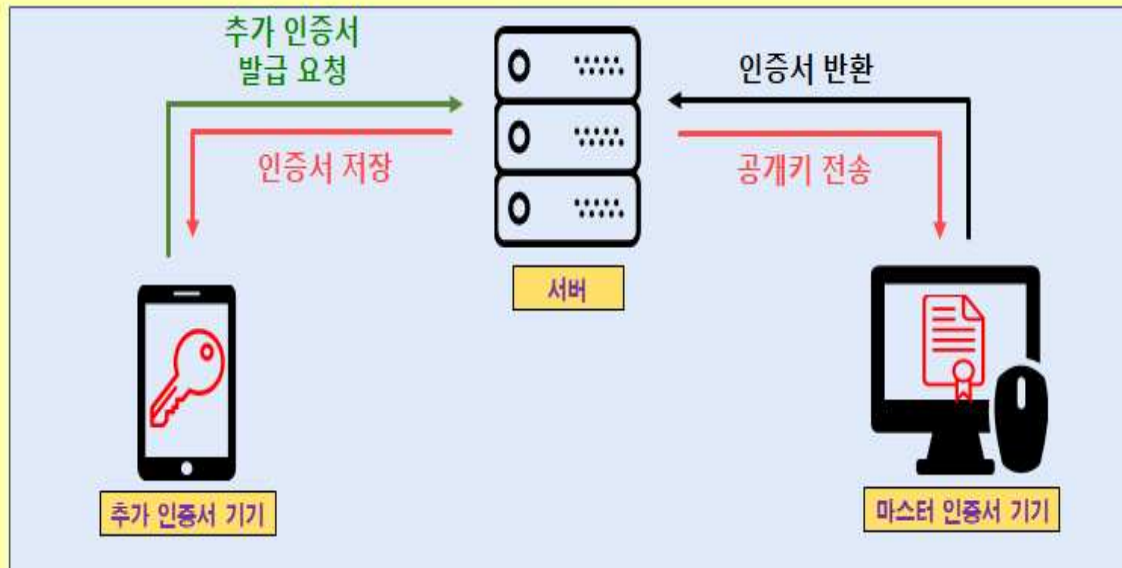


결제



8

추가 인증서 관리



9

추진경과

추진 기간 (2019년)		3월	4월	5월	6월	7월	8월	9월	10월
추진 내용									
자료조사 및 설계									
시스템 설계 및 구현	기본 페이지								
	결제 시스템								
	송금 시스템								
	인증서 발급/관리								
웹 업로드 및 테스트									

1

개발 환경 및 개발 내용(1/8)



개발 환경

- 프론트엔드 : Vue.js
- 백엔드 : Node.js/Express
- 데이터베이스 : MySQL
- 개발언어 : Javascript

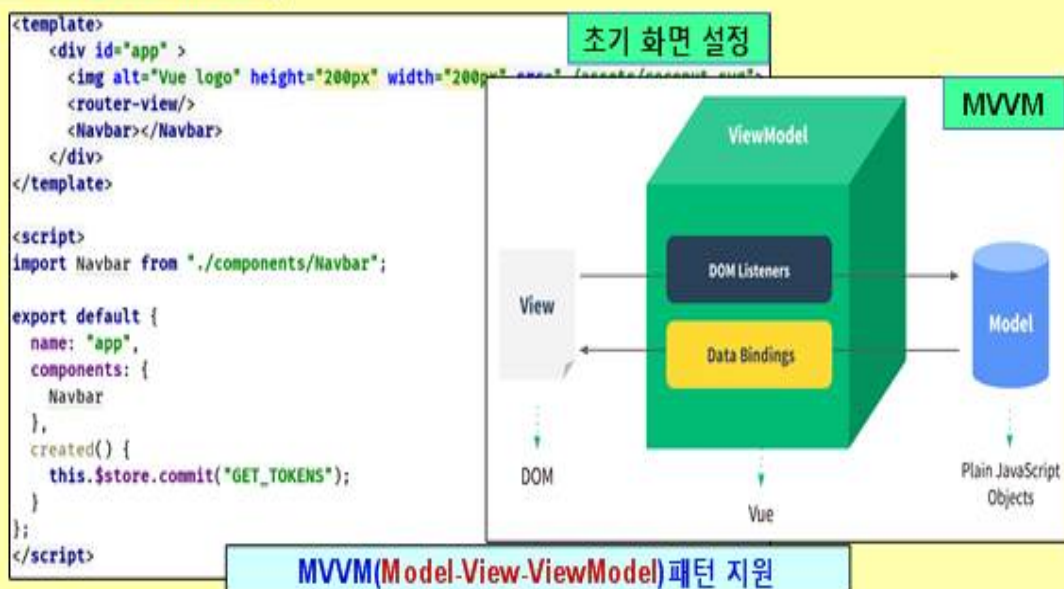


1

개발 환경 및 개발 내용(2/8)



초기 화면 구현



1

개발 환경 및 개발 내용(3/8)



회원 가입

Bcrypt : 비밀번호전용 해시함수, salt값을 이용하는 해시함수로서 입력값이 같아도 매번 다른 해시값을 반환

```
function CreateSalt(size) {
    return new Promise((executor, reject) => {
        bcrypt.genSalt(size, function (err, salt) {
            if (salt) {
                console.log("PasswordHash : " + salt);
                resolve(salt);
            } else {
                console.log("PasswordHash_err : " + err);
                reject(err);
            }
        });
    });
}

function PasswordHash(salt, pass) {
    return new Promise((executor, reject) => {
        bcrypt.hash(pass, salt, function (err, hash) {
            if (hash) {
                console.log("PasswordHash : " + hash);
                resolve(hash);
            } else {
                console.log("PasswordHash_err : " + err);
                reject(err);
            }
        });
    });
}
```

비밀번호 해쉬에 사용할 salt값을 생성

해시 알고리즘인 **bcrypt**를 이용하여 비밀번호를 해시화하여 저장

1

개발 환경 및 개발 내용(4/8)



로그인

로그인 정보 입력

```
<div class="col-md-6">
  <div class="form-group row">
    <label for="id" class="col-sm-2 col-form-label">아이디</label>
    <div class="col-sm-10">
      <input type="text" id="id" v-model="User.id" class="form-control" placeholder="아이디">
    </div>
    <br><br><br>
    <label for="password" class="col-sm-2 col-form-label">비밀번호</label>
    <div class="col-sm-10">
      <input type="password" id="password" v-model="User.password" class="form-control" placeholder="비밀번호">
    </div>
    <br>
    <button @click="onLoginSubmit" type="button" class="btn btn-primary">로그인</button>
  </div>
</div>
```

공개/비밀토큰 발급

```
function CreateAuthToken() {
  return new Promise((resolve, reject) => {
    const token = 'JWT ' + jwt.sign(
      { data: User },
      config.secret,
      { expiresIn: 86400 * 7 } //유효기간 7일
    );
    console.log("공개 토큰값 : ", token);

    const stoken = 'JWT ' + jwt.sign(
      { data: token },
      config.secret,
      { noTimestamp: true } //유효기간 무제한
    );
    console.log("비밀 토큰값 : ", stoken);
  });
}
```

Key	Value
loginWebApp	SILENT
pToken	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm91dG8uaW50IjpbImVudWkiXSwiaWF0IjoxNjUwMDAwMDAwfQ..
stoken	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkbGciOiJ0bm90bW91dG8uaW50IiwiaWF0IjoxNjUwMDAwMDAwfQ..

발급된 공개토큰/비밀토큰

로그인 상태를 유지하기 위해 공개/비밀토큰의 생성 활용

1

개발 환경 및 개발 내용(5/8)



인증서 발급

```
router.post('/newCert', (req, res, next) => {  
  console.log(JSON.stringify(req.body));  
  var cert = pki.createCertificate({  
    cert.publicKey = pki.getPublicKey(req.body.user.number);  
    cert.serialNumber = '1234567890';  
    cert.validity.notBefore = new Date(2020, 0, 1);  
    cert.validity.notAfter = new Date(2025, 0, 1);  
  });
```

유효기간 5년 설정

인증서 정보 입력

```
var userAttrs = [  
  {  
    name: 'commonName',  
    value: req.body.user.number  
  }, {  
    name: 'countryName',  
    value: 'kr'  
  }, {  
    name: 'organizationName',  
    value: 'Coconut'  
  }, {  
    shortName: 'OU',  
    value: req.body.deviceId  
  }  
];
```

서버키로 사용자 인증서 서명

```
cert.sign(caPrivateKey);
```

사용자가 입력한 정보로 인증서 생성 후 서버키로 서명

개발 환경 및 개발 내용(6/8)



서명 검증

```
for (var i=0; i<certPem.length; i++){  
  let cert = pki.certificateFromPem(certPem[i], cert);  
  console.log('cert : ' + JSON.stringify(cert));  
  var pss2 = forge.pss.create({  
    md: forge.md.sha1.create(),  
    mgf: forge.mgf.mgf1.create(forge.md.sha1.create()),  
    saltLength: 32  
  });  
  // Sign the message  
  var md2 = forge.md.sha1.create();  
  md2.update(sign, 'utf8');  
  var verifySignature = publicKey.verify(md2.digest().getBytes(), payload.signature[0], pss2);  
  console.log('Signature Verify : '+verifySignature);  
}
```

PEM 파일을 인증서로 변환

서명검증 객체 생성

사용자의 서명 검증

PEM 파일을 인증서로 변환 후 해당 인증서로 전자서명을 검증

개발 환경 및 개발 내용(7/8)



개인키 암호화

// 1. 공개키, 개인키 생성

사용자의 공개키쌍 생성

```
var keypair = pki.rsa.generateKeyPair( bits: 2048);
```

```
var publicKey = keypair.publicKey;
```

```
var privateKey = keypair.privateKey;
```

```
var encryptedPrivateKeyInfo = pki.encryptPrivateKeyInfo(
```

```
privateKeyInfo, payload.pa, options: {
```

```
algorithm: 'aes256', // 'aes128', 'aes192', 'aes256', '3des'
```

```
});
```

사용자가 입력한 비밀번호를 대칭키로 사용하여 대칭키 암호화

개인키는 사용자가 입력한 비밀번호로 암호화하여 안전하게 저장

1

개발 환경 및 개발 내용(8/8)



QR코드 인식

Terminal: Local (2) × +

NPM설치

```
C:\Users\gnsl4\WebstormProjects\Coconut>cd vue-coconut
```

```
C:\Users\gnsl4\WebstormProjects\Coconut\vue-coconut>npm install vue-qrcode-reader
```

NPM(Node Package Manger) : 자바스크립트 언어를 위한 패키지 관리자

```
import VueQrcodeReader from 'vue-qrcode-reader'
```

```
Vue.use(VueQrcodeReader);
```

QR코드 인식 패키지 적용

QR코드 인식 패키지를 사용하여 QR코드 인식 시스템 구축



1

개발 시스템 운영 (1/14)



초기화면



18

개발 시스템 운영 (2/14)



회원가입



19

개발 시스템 운영 (5/14)



추가 인증서 발급

추가 인증서 발급

추가 인증서는 발급 요청 후, 마스터 인증서 기기에
서 승인을 받은 뒤 사용이 가능합니다.
추가 인증서는 제한없이 소용이 가능합니다.

발급 승인 완료된 추가 인증서

▶ 기기명 : 한성 노트북12

기기 이름 입력

한성 노트북12

한정 사용되는 기기의 명칭이나 설명을 입력하세요. 다른 인증서와
구분해 지을 수 있습니다.

추가 인증서 비밀번호 입력

—

추가 인증서 비밀번호 확인

—

별도의 기기로 인증서 발급 신청

22

개발 시스템 운영 (6/14)



인증서 관리

인증서 관리			
기기 이름	발급 승인	상태	작업
한성 노트북01	<input type="button" value="발급 승인"/>	발급 승인 필요	
한성 노트북02	<input type="button" value="발급 승인"/>	발급 승인 필요	

마스터 인증서로 발급승인

인증서 비밀번호 입력

Password

23

개발 시스템 운영 (7/14)



마이페이지

seller3님의 마이 페이지

잔액 : 1,000,000원

상품 관리

상품 등록

판매 내역

기업판매 회원만 상품 등록 표시

24

개발 시스템 운영 (8/14)



상품 등록

상품 등록

상품명 액티브2 44mm 스테인리스

가격 400000

수량 300

카테고리 가전디지털

설명 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품)

이미지

데이터베이스에 저장

가전디지털 삼성전자 갤럭시 워치 액티브2 44mm 스테인리스 스틸 케이스 (정품) 액티브2 44mm 스테인리스 400000 10 284 JO 20 1571313645130.jpg

상품등록

25

개발 시스템 운영 (9/14)



상품 보기

카테고리별 정렬

상품 상세 보기

액티브2 44mm 스테인리스 400,000원

액티브2 44mm 스테인리스 400,000원

구매하기

26

개발 시스템 운영 (10/14)



결제

결제

바로 결제는 인증서로 결제

QR코드 결제는 QR코드로 결제

바로 결제

인증서 비밀번호 입력

Password

결제

액티브2 44mm 스테인리스 800,000 2 JOONGB

원 개

총 주문 상품수 1종 2개

총 결제 예상 금액 800,000원

바로 결제

QR코드 결제

QR코드 결제

27

개발 시스템 운영 (11/14)



QR코드 결제



PC의 QR코드를 모바일로 인식

주문 정보 확인

인증서 비밀번호 입력

Password

결제

주문금 상환유
1년 2개월

인증코드 확인하기

결제

모바일기기의 인증서로 결제

결제 완료

결제 금액	결제일
800,000원	2019-10-18 / 23:45:07

구매자	주문자
individual3	individual3

배송 주소	주문자 전화 번호
경기도 고양시 덕양구 원흥 1로 23	010-3333-3303

주문번호

78

28

개발 시스템 운영 (12/14)



장바구니

장바구니

상품정보	금액	수량	판매자	선택
 다동도 스퀴즈	7,000원	1개	JOONGBU3	<input checked="" type="checkbox"/>
 다우니 초고농축	9,000원	3개	JOONGBU3	<input checked="" type="checkbox"/>
 스타벅스 머그컵	10,000원	1개	JOONGBU3	<input checked="" type="checkbox"/>

바로구매

원하는 물품만 선택

individual3님의 주문서

1. 주문상품 확인

상품정보	금액	수량	판매자
다동도 스퀴즈	7,000원	1개	JOONGBU3
다우니 초고농축	9,000원	3개	JOONGBU3
스타벅스 머그컵	10,000원	1개	JOONGBU3

2. 배송지 정보 입력

고객입력 상세주소

경기도 고양시 덕양구 원흥 1로 23

우편번호

10502

연락처

010-3333-3003

주문하기

29

개발 시스템 운영 (13/14)



주문/구매내역

individual3님의 마이 페이지
잔액 : 5,426,000원

주문/구매내역

다용도 스위치 포함 3종 5개 총 결제금액 : 25,000원
주문번호: 79

결제 내역 확인

액티보2 44mm 스테인리스 포함 1종 2개 결제금액 : 800,000원
주문번호: 79

액티보2 44mm 스테인리스 포함 1종 2개 결제금액 : 800,000원
주문번호: 79

액티보2 44mm 스테인리스 포함 1종 2개 결제금액 : 800,000원
주문번호: 79

결제 대기 / 완료 확인

주문번호 : 79

상품명	금액	수량	판매자	비고
다용도 스위치 3M 스위치(리프트 다용도 스위치) (3개) 생물용품	7,000원	1개	JOONGBUS	미발
스타벅스 머그컵 스타벅스 블랙 데이지 컵 300ml 생물용품	10,000원	1개	JOONGBUS	미발
다우니 초고농축 F&G (다우니 초고농축 유제품) (3개) 생물용품	9,000원	3개	JOONGBUS	미발

주문자 : N 구매자 : N

배송 주소 : 경기도 고양시 덕양구 용문 1로 23 전화번호 : 010-3333-3003

결제 시간 : 2019-10-18 23:50:00

총 주문 상품수 : 3종 5개

총 결제금액 : 25,000원

30

개발 시스템 운영 (14/14)



영수증 발급

seller3님의 마이 페이지
잔액 : 7,439,000원

영수증은 판매자가 발급

판매내역

다용도 스위치 1개 상품 금액 : 7,000원
주문번호: 78

다우니 초고농축 3개 상품 금액 : 9,000원
주문번호: 78

스타벅스 머그컵 1개 상품 금액 : 10,000원
주문번호: 78

액티보2 44mm 스테인리스 2개 상품 금액 : 800,000원
주문번호: 78

영수증 발급

주문번호 : 78

영수증

영수증 번호 : 214783647

주소 : 경기도 고양시 덕양구 용문 1로 23

대표자 : seller3

전화번호 : 010-3333-3003

발행일 : 2019-10-18 23:45:07

번호 : 78

상품명	단가	수량	금액
액티보2 44mm 스테인리스	400,000	2	800,000

영수증 생성 : 2019-10-18 23:45:07

영수증 QR코드 : [QR Code]

영수증 비밀번호 입력

영수증 발급

31

결론 및 기대효과



○ 결 론

- QR코드를 이용한 간편 결제 시스템 구축에 필요한 시스템 설계, 회원가입 및 로그인, 인증서 발급 등의 프로그램 개발을 정상적으로 완료하고 시스템 구축에 성공
- 특히 QR코드 인식처리 및 온라인/오프라인 결제 등 난이도가 높은 부분의 코딩에 많은 어려움이 있었으나 조원 간 기술공유 및 협동으로 난관을 극복

○ 기대효과

- 시스템 개발과정에서 어려운 기술이론 부분의 코딩 등을 직접 체험함으로써 기술 역량을 배가하고 성과에 대한 성취감
- 간편 결제 시스템은 소상공인 점포 등에서 운영할 경우 편의성과 보안성 측면에서 유용할 것으로 기대

- 끝 -

1

Q&A

감사합니다



1