텍스트이(가) 표시된 사진

자동 생성된 설명

ICE4016 데이터베이스설계

CAR DEALER 프로젝트

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2022년 12월 17일

학부 정보통신공학

학년 3

성명 박경대

학번 12191759

1. 개요

본 프로젝트에서는 자동차 판매원과 고객을 서비스할 수 있는 데이터베이스 및 홈페이지를 설계하고 구상하였습니다. 차량의 종류와 차량의 실제 매물을 분류하여 구성하였습니다. 로그인/로그아웃 기능을 구현해 고객/관리자(판매원)가 자신의 역할(Role)에 맞는 기능을 사용할 수 있도록 쿠키를 사용해 구성하였습니다. 사용자(고객) 페이지에서는, 고객의 정보와 현재 예약 가능한 차량 목록, 그리고 예약한 목록들을 볼 수 있습니다. 관리자(판매원) 페이지에서는, 해당 판매원에게 할당된 차량들의 목록을 볼 수 있고, 그 차량들 중 예약되어 있는 차량의 정보를 확인할 수 있습니다. 또한, 예약 정보의 고객 번호나 예약 날짜를 수정할 수 있게 하였습니다. 차량의 정보를 등록하고 수정할 수 있게 하였고, 차량의 실제 매물 역시 등록 및 수정이 가능하도록 구성하였습니다. 예약된 차량의 판매여부 역시 수정할 수 있게 하였습니다. 고객이 예약할 때는 날짜만 설정하고, 시간은 영업 시간 내 언제든지 방문한다고 가정해 시간은 따로 정해주지 않습니다.

회원들은 이미 만들어져 있다고 가정해 관리자페이지에서 회원 정보를 만들거나 수정하는 기능은 구현하지 않았습니다. 모든 차량들은 각각의 차대번호가 있고, 모든 차량매물들은 모든 판매원들에게 1:1로 할당된다고 가정하였습니다. 고객의 입장에서 차량 매물의 예약과 구매를 구분하였고, 예약을 한 고객은 실제 매장에 오프라인으로 방문해 차량 구매를 결정하고 과정을 거치고 나면 관리자가 직접 관리자페이지에서 구매 여부를 수정해 판매완료 처리하는 것으로 가정하였습니다.

관리자 페이지에서는 차량 정보를 입력/수정/삭제하고, 차량 구매 예약 조회/수정/완료가 가능하도록 만들었습니다. 또한, 구매 예약 후 차량 판매가 될 때 가지 다른 사용자에게 그 차량이 검색되지 않도록 하였습니다.

사용자 페이지에서는 차량 구매 예약, 예약 조회, 예약 취소 기능을 구현하였습니다.

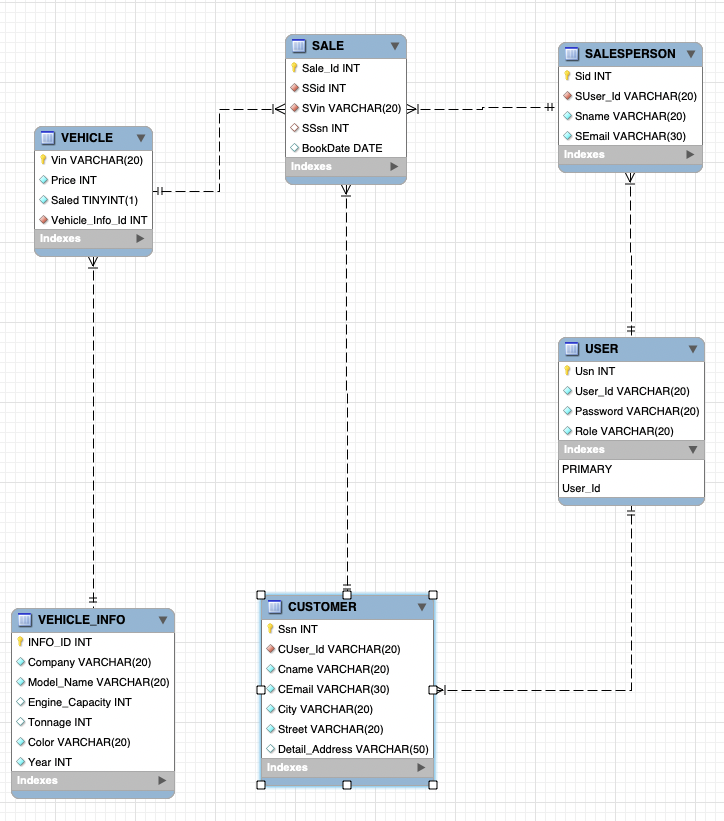
차량들의 실제 차대번호는 국제기구 및 기업에서 정한 규칙에 맞게 구성되어 있습니다. 하지만 기업마다 차대번호 규칙이 모두 달라 이번 프로젝트에 직접 반영하기에는 어려움이 있다고 판단해 유효성 검사 제약조건을 따로 걸어주지는 않았습니다.

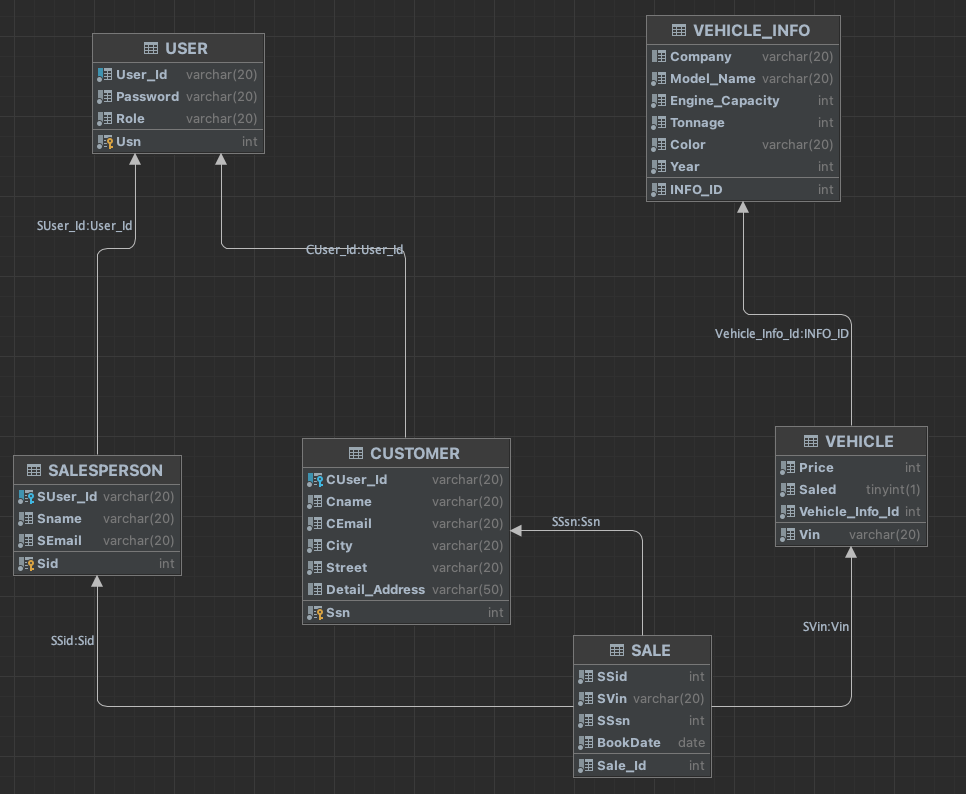
상세한 제약 조건 및 트랜잭션에 대해서는 이후 설명하겠습니다.

1. 상세 설계 내용

먼저 테이블 구조에 대해 설명드리겠습니다. 최종 ERD는 다음과 같습니다.

[[ERD 및 제약조건]]





[USER테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

User테이블은 데이터베이스에 등록되는 모든 회원 (Customer, Salesperson)들의 로그인 정보가 담겨있습니다. User\_Id는 로그인할 때 사용하는 Id 정보이고, Password는 비밀번호, Role은 역할(구매자/판매자)입니다. 이 데이터베이스에서는 Role을 “CUSTOMER”와 “ADMIN”으로 구분하여 두 종류만 저장합니다. 두 종류 외에는 삽입을 금지하기 위해 CHECK 제약조건을 설정하였습니다. 제약조건을 어기고 삽입하면 다음과 같은 오류가 발생하며 삽입이 불가합니다.



Usn은 가입한 회원들의 PK이고, AUTO\_INCREMENT를 통해 새로운 값이 들어올때마다 1씩 증가합니다. 회원의 아이디 User\_Id는 중복되어서는 안되므로 UNIQUE로 지정해 주었습니다. 또한, USER테이블의 모든 정보들은 꼭 있어야 하므로 NOT NULL 제약조건을 모든 애트리뷰트에 걸어주었습니다.

[SALESPERSON 테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

Salesperson테이블은 판매원들의 정보를 담고 있는 테이블입니다. Sid는 PK이고 판매원 구분 번호입니다. SUser\_Id는 User테이블의 User\_Id 애트리뷰트를 참조하며 다음과 같은 구문으로 정의되어 있습니다.



User테이블의 User\_Id를 fk로 참조하며, 만약 User테이블의 회원 튜플이 삭제되면 같이 삭제될 수 있도록 ON DELETE CASCADE 제약조건을 사용하였습니다. USER 테이블의 User\_Id와 1:1 관계이므로 UNIQUE 제약조건을 걸어주었습니다.

Sname은 판매원의 이름입니다.

SEmail은 판매원의 이메일 주소입니다. 이메일 형식임을 확인하기 위해 정규식을 활용해 CHECK 제약조건을 설정하였습니다.



이메일 형식에 맞지 않은 값이 삽입되는 것을 막습니다.

SALESPERSON테이블의 모든 정보들은 꼭 있어야 하므로 NOT NULL 제약조건을 모든 애트리뷰트에 걸어주었습니다.

[CUSTOMER 테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

Customer 테이블은 고객들의 정보를 담고 있는 테이블입니다. Salesperson 테이블과 유사합니다. Ssn은 PK이고 고객 구분 번호입니다. CUser\_Id는 User테이블의 User\_Id 애트리뷰트를 참조하며 다음과 같은 구문으로 정의되어 있습니다.



User테이블의 User\_Id를 fk로 참조하며, 만약 User테이블의 회원 튜플이 삭제되면 같이 삭제될 수 있도록 ON DELETE CASCADE 제약조건을 사용하였습니다. USER 테이블의 User\_Id와 1:1 관계이므로 UNIQUE 제약조건을 걸어주었습니다.

Cname은 판매원의 이름입니다.

CEmail은 판매원의 이메일 주소입니다. 이메일 형식임을 확인하기 위해 정규식을 활용해 CHECK 제약조건을 설정하였습니다.



이메일 형식에 맞지 않은 값이 삽입되는 것을 막습니다.

Detail\_Address애트리뷰트는 상세 주소입니다. 상세 주소가 필요 없는 경우 (개인 주택 거주자)를 고려해 NULL을 허용하였습니다. 그 외에는 모두 NOT NULL 조건을 걸어주었습니다.

[SALE 테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

Sale테이블은 판매하는 차량, 고객, 판매원간을 연결해주는 관계 테이블입니다. 관계되는 모든 정보들은 Sale\_Id 관리번호를 가집니다. 이는 AUTO\_INCREMENT로 새로운 값이 삽입될 때 마다 1씩 증가하고, 이 테이블의 PK입니다.

SSid는 판매원의 번호입니다. SVin은 판매할 자동차의 차대번호입니다. 이 둘은 반드시 존재해야 판매 등록을 할 수 있고, 고객이 예약할 수 있습니다. 따라서 NOT NULL 제약조건을 걸어주었습니다. 한 차량을 여러 판매원이 팔 수는 없으므로 UNIQUE 제약조건을 걸어주었습니다.

SSsn은 구매자의 번호, BookDate는 예약날짜 입니다. 이들의 초기상태는 NULL값인데, 구매자가 예약을 한다면 여기에 값이 들어가며 예약 상태가 됩니다.

예약날짜에 대해서 데이터베이스에는 특별한 조건을 걸어주지 않았습니다. Html에서 오늘 날짜로부터 N일 이후까지만 정할 수 있도록 제약을 걸어 고객이 너무 먼 날짜를 선택하는것을 방지할 수 있고, 이렇게 하면 관리자가 임의로 과거 혹은 먼 미래의 날짜를 삽입할 수 있게 하여 관리자의 자유도를 높였습니다.

판매원 및 구매자의 번호는 각각 SALESPERSON, CUSTOMER 테이블의 PK를 참조합니다. 만약 참조하는 테이블의 PK가 변경되면 SALE테이블에서의 판매원 및 구매자의 번호도 변동될 수 있도록 ON UPDATE CASCADE 제약조건을 걸어주었습니다. 또한, 차량의 차대번호는 VEHICLE테이블의 Vin 애트리뷰트를 참조합니다. 만약 참조하는 테이블의 Vin 값이 변동되면 SALE테이블에서의 Vin도 변동될 수 있도록 ON UPDATE CASCADE 제약조건을, 참조 테이블에서 Vin이 삭제된다면 SALE테이블에서도 삭제될 수 있도록 ON DELETE CASCADE 조건을 걸어주었습니다.

[VEHICLE\_INFO 테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

Vehicle\_info테이블은 차량 정보들을 담고 있는 테이블입니다.  
INFO\_ID는 차량 정보 번호입니다. PK이며 AUTO\_INCREMENT로 새로운 값이 삽입될 때 마다 1씩 증가합니다.  
Company는 제조 회사, Model\_Name은 차량 모델 이름, Engine\_Capacity는 엔진 용량 (단위:cc), Tonnage는 차량 적재 톤수, Color는 색상, Year는 제작년도 입니다. 차량 용량, 차량 톤수는 차량 종류에 따라 NULL이 될수 있다고 생각해 NOT NULL 제약조건을 걸지 않았고, 그 외의 애트리뷰트들은 차량이라면 모두 다 가지고 있는 정보라고 판단해 NOT NULL 제약조건을 걸었습니다.

VEHICLE\_INFO 테이블은 모든 애트리뷰트들에 대해 UNIQUE 제약조건을 가지고 있습니다.



특징이 정해져 있는 차량 정보가 두 개 이상 존재할 수 없기 때문에 모든 애트리뷰트에 대해 UNIQUE 제약조건을 걸었습니다. 다만, Tonnage와 Engine\_Capacity가 NULL일 경우 이에 대한 정보가 정해지지 않았기에 알 수 없다고 고려해 UNIQUE 조건을 고려하지 않아도 상관없다고 생각했습니다.

[VEHICLE 테이블]

텍스트이(가) 표시된 사진

자동 생성된 설명

Vehicle테이블은 차량 매물들을 담고 있는 테이블입니다.  
Vin은 차대번호이고 PK입니다. Price는 가격, Saled는 판매 여부, Vehicle\_Info\_Id는 차량 정보 번호입니다. 차량 정보 번호는 Vehicle\_Info 테이블을 참조하고 있으며 참조하는 테이블의 값이 삭제되면 해당 튜플도 같이 삭제될 수 있도록 ON DELETE CASCADE 제약조건을 활용하였습니다. 하나의 차량 정보는 여러 대의 차량 매물을 가르킬 수 있으므로 Vehicle\_Info 테이블의 Info\_Id와는 1:1 관계가 아니기 때문에 UNIQUE 제약조건을 걸지 않았습니다. 모든 값들은 차량 정보에 필수이므로 NOT NULL 제약조건을 걸어주었습니다.

* CUSTOMER : USER = 1 : 1, SALESPERSON : USER = 1 : 1  
  고객 및 판매원은 모두 하나의 User\_Id만 가질 수 있고, 모든 회원들은 반드시 고객 혹은 판매원 중 하나의 아이디를 가집니다.
* VEHICLE\_INFO : VEHICLE = 1 : N  
  모든 차량 매물(VEHICLE)들은 하나의 차량 정보(VEHICLE\_INFO)만 가질 수 있지만, 모든 차량 정보(VEHICLE\_INFO)들은 여러 대의 차량 매물(VEHICLE)들에 사용될 수 있습니다.
* VEHICLE : SALESPERSON = N : 1, VEHICLE : CUSTOMER = N : 1, CUSTOMER : SALESPERSON = N : M  
  모든 고객 및 판매원은 여러 대의 자동차를 예약하거나 할당받을 수 있습니다.  
  하지만 모든 차량 매물들은 한 명의 고객 혹은 한 명의 판매원에게 예약되거나 할당받을 수 있습니다.  
  한 명의 고객은 여러 명의 판매원들과 거래할 수 있고, 한 명의 판매원은 여러 명의 고객과 거래할 수 있습니다.

[[정규화]]

다음은 전체 테이블에 대한 스키마입니다.

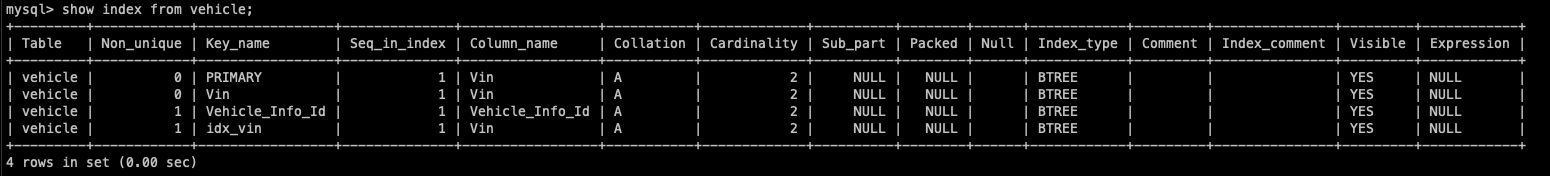
테이블이(가) 표시된 사진

자동 생성된 설명

다치 애트리뷰트를 가지는 테이블이 없고, 2개 이상의 PK를 가지는 테이블이 없습니다. 또한, PK이외에 다른 애트리뷰트를 결정할 수 있는 애트리뷰트가 없습니다. 따라서 부분 함수 및 이행 함수 종속이 존재하지 않아 최소 3NF를 만족한다고 할 수 있습니다.

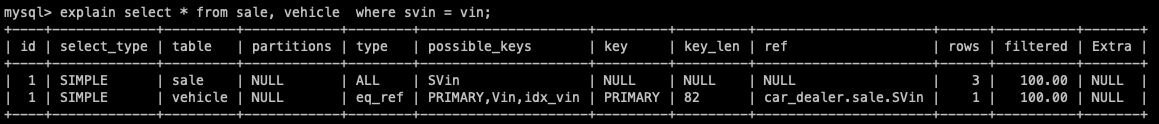
[[인덱스 생성]]

인덱스를 사용하면 데이터베이스의 성능 향상을 기대할 수 있습니다. WHERE, GROUP BY 절에 많이 사용되고 정적인 테이블에 거는것이 유리합니다. 이 데이터베이스에서 가장 카디널리티가 높고 수정이나 삭제의 빈도가 가장 적은 Vin 애트리뷰트가 인덱스를 걸기 적당하다고 생각했습니다. 또, 차대 번호는 모두 다 다르기 때문에 카디널리티가 가장 높다고 할 수 있습니다.



텍스트이(가) 표시된 사진

자동 생성된 설명



VEHICLE테이블의 Vin애트리뷰트 외에 다른 애트리뷰트들은 인덱스를 걸지 않아도 괜찮다고 판단해 걸지 않았습니다.

인덱스를 너무 많이 걸면 데이터베이스의 성능에 좋지 않습니다. 오히려 삽입/삭제/수정을 자주 할 수록 성능이 저하되고 인덱스를 관리하기 위한 추가 작업 및 추가 저장 공간이 필요하기에 위에서 제시한 조건에 부합하는 애트리뷰트에 인덱스를 사용하는게 제일 적합합니다.

[[View생성]]

VIEW를 사용하면 사용자에게 허용된 자료만을 제한적으로 보여줄 수 있고, 복잡한 쿼리문을 사전에 구현하여 개발 시 편리하게 사용할 수 있습니다. 저는 이 프로젝트에서 사용자에게 제한적으로 보여주어야 하는 정보는 쿼리문이 아닌 hbs 파일에서 제어하였기 때문에 편의 측면에서만 사용하겠습니다. 이 프로젝트에서 길이가 긴 쿼리문 중 WHERE절을 사용하지 않아 추가 변수를 필요로 하지 않는 쿼리문을 view로 제작하겠습니다.

view로 제작할 쿼리문은 다음과 같습니다.



이 쿼리문의 결과를 hbs에서 출력하면 다음과 같이 나옵니다.



엔진 용량의 “cc”, 제작 년도의 “년”, 가격의 “만원”은 쿼리문에서 추출한것이 아닌 hbs파일에서 조작해준 것입니다. 이들도 함께 같이 나올 수 있도록 view를 다음과 같이 제작하겠습니다. 또한, “차량 톤수”나 “엔진 용량”등 null이 허용되는 값들에 대해 null인 경우 “-“로 표현하여 가독성을 높이겠습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 view를 제작하였습니다. 이 view의 실행 결과는 다음과 같습니다.

텍스트, 전자기기, 점수판, 스크린샷이(가) 표시된 사진

자동 생성된 설명

concat(str1, str2) 구문은 문자열 str1과 str2를 합치는 명령어입니다. IFNULL(attr1, str1)은 attr1의 애트리뷰트의 값이 NULL이라면 str1 문자열로 대치해서 보여주라는 의미입니다.

[[웹 페이지 생성]]

먼저 각 URI의 접두어를 Router JS로 설정해주는 index.js입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<<로그인 기능>>

먼저 브라우저에 쿠키에 회원 정보가 담겨있는지 확인합니다. User\_Ssn, User\_Role이 없으면 로그인되지 않았다고 판단해 /login으로 Redirect하여 GET요청을 보냅니다.

[login.js]

텍스트이(가) 표시된 사진

자동 생성된 설명

로그인 정보와 관련된 쿠키가 없다면 48번째 line의else문에서 login.hbs를 응답합니다.

[login.hbs]

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

로그인 화면을 구성하는 hbs파일입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명  
▲login.hbs의 로그인 화면

위와 같이 아이디와 비밀번호를 입력하는 공간이 있고, 로그인 버튼을 누르면 “/”로 POST요청을 보냅니다. 이 POST요청은 login.js에서 처리합니다.

[login.js]

텍스트이(가) 표시된 사진

자동 생성된 설명

62번째 줄에서 login.hbs로부터 받은 로그인 form의 id와 password를 가지고 있는 회원이 user에 있는지 확인합니다. 만약 있다면, user테이블의 해당 회원의 **Usn**과 Role을 가져옵니다. 하지만 Usn을 쿠키에 그대로 담아 사용하기에는 다른 테이블에서 고객 및 판매원의 회원 정보가 Usn이 아닌 Sid 및 Ssn 즉 해당 테이블들의 PK로 구성되어 있어 이를 변환하는 작업이 필요합니다. 이를 74번째 줄에서 getssn()을 이용해 추출해내고 있습니다. getssn은 sql.js의 selectSql 객체에 정의되어 있습니다.

[sql.js]

텍스트, 스크린샷, 화면, 닫기이(가) 표시된 사진

자동 생성된 설명

입력받은 Role이 판매원이라면 SALESPERSON 테이블의 Sid를, 고객이라면 CUSTOMER 테이블의 Ssn을 추출합니다. 다시 login.js의 “/” POST 요청 설명으로 돌아가서, 이렇게 받은 회원 ID를 사용해 82번째줄부터 User\_Ssn과 User\_Role이라는 이름으로 쿠키를 만들었습니다. 이렇게 만든 쿠키를 이용해 다시 인증할 수 있도록 90번째 줄에서 로그인 페이지로 Redirect 즉 GET요청을 보내고 있습니다.

이제 login.js의 24번째 줄 혹은 36번째 줄에서 ADMIN인지, CUSTOMER인지 판단해 각각 saleman\_main.hbs, customer\_main.hbs를 적절한 객체들과 함께 응답하고 있습니다.  
만약 이러한 인증절차에 문제가 생겼다면 43번째 줄에서 모든 쿠키를 지우고 다시 login하도록 구현하였습니다.