Databases Programming Project: **Final Report**

이름 (Name): 이상준

학과 (Department): 정보컴퓨터공학부

학번 (Student ID): 201924530

---

1. 프로젝트 개요 (Project topic)

※ 프로젝트 주제, 목표와 본인 프로젝트의 차별성, 장점 등에 대해 상세히 언급해주세요.
※ 만약 Github 프로젝트로 관리가 되고 있고 README에 설명이 충분히 되어 있다면 Github Repository 링크를 적어도 좋습니다.

사용한 언어 및 라이브러리: Python, psycopg2, re
프로젝트 개요:
맛집의 실시간 대기 시스템.
실시간 대기 시스템은 가게의 사장이 등록한 가게에 대하여 이를 방문하고자 하는 고객이 원격으로 줄을 서는 시스템이다. 고객은 식당의 실시간 대기열에 들어가 가게에 방문하지 않고 줄서기에 참여할 수 있다.
위 시스템은 원격 줄서기 외에도 대기열 인원의 분포를 통한 실시간 대기 시간, 실제 식사를 마친 사용자를 통한 리뷰 기능 및 리뷰를 통한 가게의 평균 평점 공개 서비스 또한 부가 요소로 제공한다.
이 시스템을 통해 고객은 평점과 리뷰를 바탕으로 원하는 가게를 줄 설 필요 없이 방문할 수 있으며, 사장은 가게의 홍보와 테이블 회전율을 높일 수 있다.

2. 사용자 (역할) (Users / Roles)

맛집의 실시간 대기 시스템을 이용하는 사용자로는 고객, 사장, 관리자가 있다.
고객(user)은 현재 방문가능한 음식점의 리스트를 조회하여 각 가게의 실시간 대기 현황과 평균 평점 등의 정보를 바탕으로 원하는 음식점에 원격 줄서기 참여할 수 있으며 대기열 참여 중 실시간 자신이 속한 대기열의 상황을 확인할

수 있다. 또한, 실 방문 이후 리뷰쓰기가 가능하다.
사장(owner)은 자신의 가게를 등록하고 삭제, 조회할 수 있으며, 등록된 가게의 시작과 마감을 관리할 수 있다. 또한, 각 가게의 대기열에서 고객을 관리(입장)할 수 있다. 추가적으로 사장은 자신의 가게에 해당하는 리뷰 중 악의적인 내용이 있다면 관리자에게 신고할 수 있다.
관리자(Admin)는 사장으로부터 신고 받은 리뷰를 검토하며 악의적인 리뷰로 판단할 경우 리뷰를 삭제할 수 있다. 또한 고객, 사장을 임시 차단할 수 있다.

3. 기능 (Functions)

※ 각 기능이 어떤 사용자를 위한 것이며, 어떤 SQL feature를 사용하는지 명시해주세요.

* 모든 기능을 함수화, 각 함수별로 트랜잭션으로 묶어서 data integrity를 충족하고자 하였습니다.

공통 기능

1. 회원 가입 – 새로운 사용자는 회원 가입을 통해 자신의 계정, 암호, 역할을 등록할 수 있습니다. 사장과 관리자로서 회원 가입하려면 각 암호키(imowner, imadmin)가 필요하며 특히 입력한 암호는 정규표현식을 통해 기준에 맞는지 유효성 검사를 진행합니다. 검사에 통과될 경우 users 테이블에 insert됨과 동시에 데이터베이스 사용자가 create되며 가입시 선택한 역할에 따라 차등 되어 권한이 grant 됩니다.
2. 로그인 – 모든 사용자는 회원 가입을 통해 생성한 계정을 이용해 프로그램에 로그인할 수 있으며, 로그인을 통해 초기 connect를 자신의 connect로 변경하게 됨으로써 각자 역할에 맞는 권한이 주어집니다.

고객(user) 기능

1. 가게 조회 – 고객이 현재 대기열에 참여할 수 있는 가게의 목록을 보여줍니다. restaurants에서 where open_status = true를 활용해 현재 오픈중인 가게만 selection하여 restaurant_id를 기준으로 정렬해 사용자에게 출력해줍니다.
2. 가게 상세 조회 – 가게 조회를 통해 고객이 선택한 가게의 상세 정보(가게id, 이름, 주소, 평점, 대기열, 예상대기시간)을 보여줍니다. resturants, waitings 테이블을 각각 select하여 필요한 정보를 가져옵니다.

3. 대기열 등록 – 상세 조회를 마친 고객이 원할 경우 대기열에 등록됩니다. – waitings을 select하여 현재 로그인 된 고객이 다른 대기열에 참여중 인지 확인한 후 참여중인 대기열이 없다면 waitings에 insert하여 대기열에 추가됩니다.
4. 대기열 조회 – 고객은 현재 자신이 대기중인 대기열의 상황을 조회함으로써 남은 예상 대기시간을 확인할 수 있습니다. waitings와 restaurants 테이블을 join하여 select하여 사용자가 대기중인 restaurant에 참여중인 waiting의 개수를 세어 구현하였습니다.
5. 리뷰 작성 – 고객은 가게에 실 방문이후 리뷰를 작성할 수 있습니다. 실 방문시에만 리뷰를 작성할 수 있도록 제약사항을 두기 위해서, 사장이 가게의 대기열에서 해당 고객을 pop하여 waitings 테이블에서 delete하는 순간에만 reviews에 새로운 리뷰를 insert(평점과 내용의 defauilt: null)합니다. 고객은 이후 새로운 리뷰를 한번만 수정함(평점과 내용이 null인 review만 수정 가능함)으로써 리뷰를 작성할 수 있도록 구현하였습니다. 또한 리뷰 작성과 동시에 해당 가게의 평균 평점을 갱신하여 이후 가게 상세 조회 등에 활용할 수 있도록 하였습니다.

사장(owner) 기능

1. 내 가게 조회 – 사장은 자신 소유의 가게를 여러 개 가질 수 있으며, 이를 구현하기 위해 users와 restaurants 테이블을 따로 작성하였습니다. 사장이 자신 소유의 가게를 조회할 수 있도록 restaurants 테이블에서 자신의 id와 동일한 가게만 selection함으로써 자신의 가게만 조회할 수 있도록 구현하였습니다.
2. 내 가게 상태 변경(오픈 및 마감) – 사장은 자신 소유의 가게의 상태를 각각 수정(오픈 혹은 마감)할 수 있습니다. 각 가게는 사용자가 조회 시 오픈일 경우에만 조회되며 마감인 경우에는 사용자에게 보이지 않습니다. 이를 구현하기 위해서 자신의 가게가 맞는지 유효성 검사 이후 update를 통해 restaurants의 open_status attribute를 바꾸도록 구현하였습니다.
3. 대기열 관리 (고객 입장) – 사장은 자신 소유의 가게를 선택하여 해당 가게의 대기열에서 우선순위가 가장 높은(1) 고객 한 명을 입장시킬 수 있습니다. 고객이 입장한다면 자동으로 남은 다른 고객들은 우선순위가 1씩 감소하게 됩니다. 이를 구현하기 위해 먼저 자신의 가게가 맞는지

유효성 검사를 진행한 후, 고객을 입장시킬 때마다 reviews에 입장한 고객을 추가, waitings에서는 입장한 고객을 삭제, waitings중 해당 가게에 해당하는 waiting들의 priority를 모두 1씩 감소하여 구현하였습니다.

4. 리뷰 조회 (악성 리뷰 신고) – 사장은 자신 소유의 가게를 선택하여 해당 가게에 달린 리뷰를 조회할 수 있으며, 이 중 악의적이다고 판단한 리뷰를 관리자에게 신고하여 검토를 요청할 수 있습니다. 이를 구현하기 위해 자신 소유의 가게가 맞는지 유효성 검사를 진행한 후, reviews에서 해당 가게 id에 해당하는 row들을 select하여 조회합니다. 이후 리뷰를 신고하게 된다면 해당 리뷰가 존재하는지, 이미 신고된 리뷰는 아닌지, 자신의 가게에 해당하는 리뷰가 맞는지 유효성 검사를 진행하며 통과할 경우 reports에 insert함으로써 리뷰를 신고할 수 있도록 구현하였습니다.
5. 가게 등록 및 삭제 – 사장은 자기 소유의 가게를 등록하거나 삭제할 수 있습니다. 시스템은 사장으로부터 가게의 이름, 주소를 입력 받아 restaurants 테이블에 등록함으로써 가게 등록을 구현합니다. 가게 삭제 기능은 먼저 자신의 가게의 목록을 보여준 뒤 삭제할 가게의 ID를 입력 받아 자신의 가게가 맞는지 유효성 검사를 거친 뒤 restaurants 테이블에서 delete 함으로써 구현합니다.

관리자(admin) 기능

1. 리뷰 검토 – 관리자는 사장이 신고한 리뷰들을 검토하여 삭제하거나 반려할 수 있습니다. 이를 구현하기 위해 관리자는 reports 테이블에서 approved가 아직 null인 report들을 조회하여 리뷰를 검토할 수 있습니다. 한 review에 대해 여러 번 신고를 막기 위해 reports에 approved attribute를 추가하여 신고를 승인할 경우 reviews에서 해당 리뷰를 delete한 후 report에서 approved값을 true로, 반려할 경우 리뷰를 삭제하지 않고 report에서 approved값을 false로 바꿉니다.
2. 사용자 관리 – 관리자는 고객 또는 사장을 임시 차단하거나 해제할 수 있습니다. 이를 구현하기 위해서 users테이블에 blocked라는 attribute를 두어 blocked가 true인 경우 해당 사용자가 로그인하지 못하도록 구현하였습니다. 관리자가 임의의 사용자를 차단하고자 하는 경우 users 테이블에서 역할이 user 또는 owner이며, blocked가 false인 리스트를 조회하여 해당 사용자의 blocked를 true로 변경할 수 있으며, 임의의 사용자

를 차단 해제하고자 하는 경우 users 테이블에서 역할이 user 또는 owner이며, blocked가 true인 리스트를 조회하여 해당 사용자의 blocked를 false로 변경할 수 있습니다.

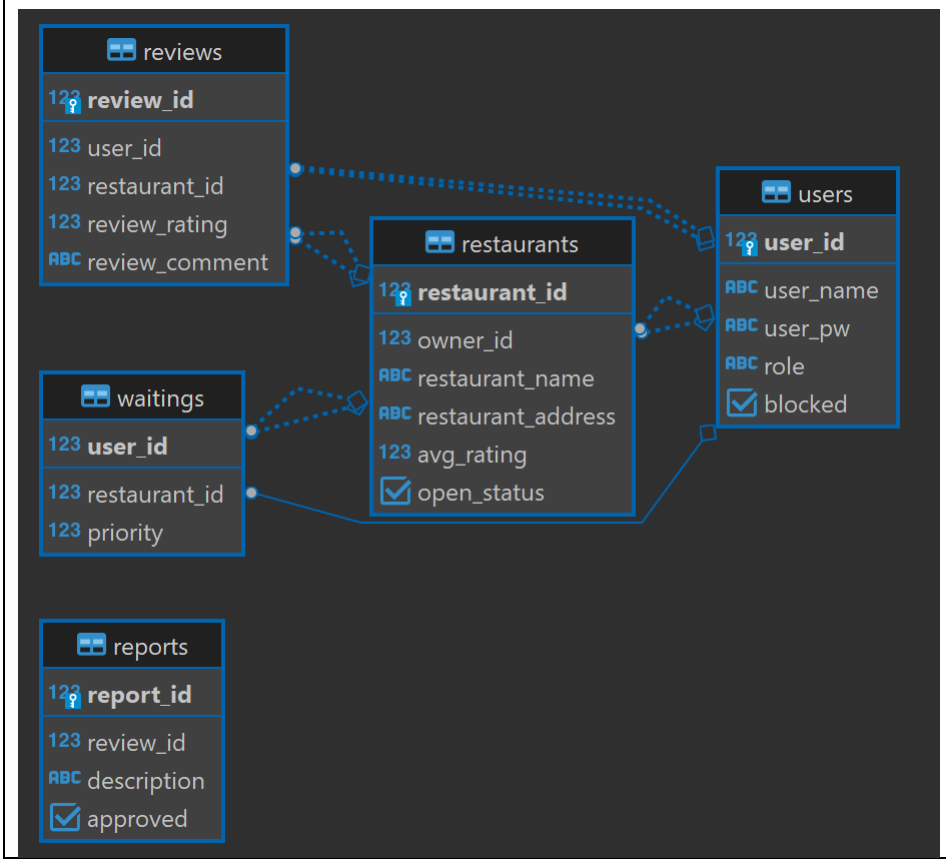
4. 데이터베이스 스키마 및 다이어그램 (Database schema / Schema diagram)

```
CREATE TABLE users (
    user_id serial4 NOT NULL,
    user_name varchar(255) NULL,
    user_pw varchar(255) NULL,
    "role" varchar(50) NULL,
    "blocked" bool NULL DEFAULT false,
    CONSTRAINT users_pkey PRIMARY KEY (user_id),
    CONSTRAINT users_role_check CHECK (((role)::text = ANY ((ARRAY['admin'::character varying, 'owner'::character varying, 'user'::character varying])::text[]))),
    CONSTRAINT users_user_name_key UNIQUE (user_name),
    CONSTRAINT users_user_pw_check CHECK (((length((user_pw)::text) >= 8) AND (POSITION((' '::text) IN (user_pw)) = 0)))
);
CREATE TABLE restaurants (
    restaurant_id int4 NOT NULL DEFAULT nextval('restaurant_restaurant_id_seq'::regclass),
    owner_id int4 NULL,
    restaurant_name varchar(255) NULL,
    restaurant_address varchar(255) NULL,
    avg_rating numeric NULL,
    open_status bool NULL,
    CONSTRAINT restaurant_pkey PRIMARY KEY (restaurant_id),
    CONSTRAINT restaurant_restaurant_name_key UNIQUE (restaurant_name),
```

```
    CONSTRAINT fk_restaurant_owner FOREIGN KEY (owner_id) REFERENCES public.users(user_id),
    CONSTRAINT restaurant_owner_id_fkey FOREIGN KEY (owner_id) REFERENCES public.users(user_id) ON DELETE SET NULL
);

CREATE TABLE waitings (
    restaurant_id int4 NULL,
    user_id int4 NULL,
    priority int4 NULL,
    CONSTRAINT waiting_priority_check CHECK ((priority >= 0)),
    CONSTRAINT waiting_user_id_key UNIQUE (user_id),
    CONSTRAINT fk_waiting_restaurant FOREIGN KEY (restaurant_id) REFERENCES restaurants(restaurant_id),
    CONSTRAINT waiting_restaurant_id_fkey FOREIGN KEY (restaurant_id) REFERENCES public.restaurants(restaurant_id),
    CONSTRAINT waiting_user_id_fkey FOREIGN KEY (user_id) REFERENCES public.users(user_id)
);
CREATE TABLE reports (
    report_id int4 NOT NULL DEFAULT nextval('report_report_id_seq'::regclass),
    review_id int4 NULL,
    description varchar(255) NULL,
    approved bool NULL,
    CONSTRAINT report_description_check CHECK ((length((description)::text) >= 15)),
    CONSTRAINT report_pkey PRIMARY KEY (report_id)
);
CREATE TABLE reviews (
    review_id serial4 NOT NULL,
    user_id int4 NULL,
    restaurant_id int4 NULL,
```

```
    review_rating numeric NULL,
    review_comment varchar(255) NULL,
    CONSTRAINT reviews_pkey PRIMARY KEY (review_id),
    CONSTRAINT reviews_review_comment_check CHECK (((review_comment IS NULL) OR (length((review_comment)::text) >= 15))),
    CONSTRAINT reviews_review_rating_check CHECK (((review_rating IS NULL) OR ((review_rating >= (0)::numeric) AND (review_rating <= (5)::numeric)))),
    CONSTRAINT fk_review_restaurant FOREIGN KEY (restaurant_id) REFERENCES public.restaurants(restaurant_id),
    CONSTRAINT fk_review_user FOREIGN KEY (user_id) REFERENCES public.users(user_id),
    CONSTRAINT reviews_restaurant_id_fkey FOREIGN KEY (restaurant_id) REFERENCES public.restaurants(restaurant_id),
    CONSTRAINT reviews_user_id_fkey FOREIGN KEY (user_id) REFERENCES public.users(user_id)
);
```



5. (팀 프로젝트인 경우만 해당) 팀원의 역할 배분

※ 각 팀원이 수행한 업무를 기재해주세요.