

**CS2102**

**Project Report**

**Topic D – Car Pooling**

**Group 8**

1. Ang Wei Ming, A0168721B
2. Benjamin Chin Choon Kiat, A0168698B
3. Lee Yu Choy, A0177151H
4. Yeo Cheng Hong, A0168369L

Contents

[1](#_Toc529202609)

[**General Architecture** 3](#_Toc529202610)

[**ER Diagram** 3](#_Toc529202611)

[**SQL DDL** 4](#_Toc529202612)

[Triggers and Functions 6](#_Toc529202613)

[Sample SQL 7](#_Toc529202614)

[Search Ride 7](#_Toc529202615)

[**Rejection of ride bid** 8](#_Toc529202616)

[**Trigger behind assert** 8](#_Toc529202617)

[Simple Insert Query 9](#_Toc529202618)

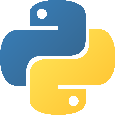
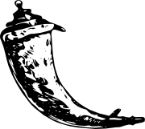
[Ride History 9](#_Toc529202619)

[Trigger 9](#_Toc529202620)

[Simple Update Query 10](#_Toc529202621)

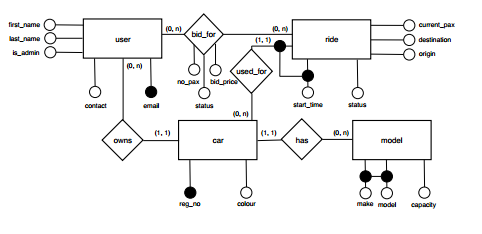
[Screen shots 10](#_Toc529202622)

**General Architecture**



* Server Language – Python 3.7.0
* Web Server - Python Flask 1.0.2
* Database – PostgreSQL 11.0

**ER Diagram**



**SQL DDL**

**user Schema**

create table if not exists "user" -- `"` used because PostgreSQL use 'user' as a keyword  
(  
email varchar(256) not null  
constraint user\_pkey  
primary key,  
contact numeric(8),  
first\_name varchar(50) not null,  
last\_name varchar(50) not null,  
is\_admin boolean default false not null,  
password varchar(512) not null  
)  
;  
​

**car model Schema**

create table if not exists model  
(  
model varchar(256) not null,  
make varchar(256) not null,  
capacity integer not null  
constraint capacity\_min  
check (capacity > 0),  
constraint model\_pk  
primary key (model, make)  
)  
;  
​  
​**car Schema**

create table if not exists car  
(  
reg\_no varchar(8) not null  
constraint car\_pkey  
primary key,  
colour varchar(50),  
email varchar(256) not null  
constraint car\_email\_fkey  
references "user",  
make varchar(50) not null,  
model varchar(50) not null,  
constraint car\_make\_fkey  
foreign key (make, model) references model (make, model)  
)  
;  
​  
​

**ride Schema**

create table if not exists ride  
(  
start\_time timestamp not null,  
status varchar(11) not null  
constraint ride\_status\_type  
check (((status)::text = 'in progress'::text) OR ((status)::text = 'completed'::text)),  
current\_pax integer not null,  
destination varchar(256) not null,  
origin varchar(256) not null,  
reg\_no varchar(8) not null  
constraint ride\_reg\_no\_fkey  
references car,  
constraint ride\_pkey  
primary key (start\_time, reg\_no)  
)  
;  
​  
​**ride\_bid Schema**

create table if not exists ride\_bid  
(  
email varchar(256) not null  
constraint ride\_bid\_email\_fkey  
references "user",  
start\_time timestamp not null,  
reg\_no varchar(8) not null,  
no\_pax integer not null  
constraint min\_pax  
check (no\_pax > 0),  
bid\_price double precision,  
status varchar(13) default 'pending'::character varying not null  
constraint bid\_status\_type  
check (((status)::text = 'pending'::text) OR ((status)::text = 'successful'::text) OR ((status)::text = 'unsuccessful'::text)),  
constraint ride\_bid\_pkey  
primary key (email, start\_time, reg\_no),  
constraint ride\_bid\_start\_time\_fkey  
foreign key (start\_time, reg\_no) references ride  
)  
;  
​

**audit\_log Schema**

create table if not exists audit\_log  
(  
start\_time timestamp not null,  
end\_time timestamp not null,  
status varchar(11) not null  
constraint ride\_status\_type  
check ((status)::text = 'completed'::text),  
current\_pax integer not null,  
destination varchar(256) not null,  
origin varchar(256) not null,  
reg\_no varchar(8) not null  
constraint ride\_reg\_no\_fk  
references car,  
constraint ride\_pk  
primary key (start\_time, reg\_no)  
)  
;  
​  
​

### Triggers and Functions

​  
create or replace function on\_approval\_update\_pax() returns trigger  
language plpgsql  
as $$  
BEGIN  
IF NEW.status = 'successful' and OLD.STATUS <> 'successful'  
THEN  
UPDATE ride  
SET current\_pax = current\_pax + NEW.no\_pax  
WHERE reg\_no = NEW.reg\_no  
AND start\_time = NEW.start\_time;  
 UPDATE ride\_bid rb  
   SET status = 'unsuccessful'  
 FROM ride r, car c, model m  
 WHERE r.reg\_no = rb.reg\_no  
 AND r.start\_time = rb.start\_time  
 AND r.reg\_no = c.reg\_no  
 AND c.make = m.make  
 AND c.model = m.model  
 AND rb.reg\_no = NEW.reg\_no  
 AND rb.start\_time = NEW.start\_time  
 AND rb.status = 'pending'  
 AND rb.no\_pax > (m.capacity - r.current\_pax);  
END IF;  
​  
RETURN NULL;  
END  
$$  
;  
​  
​  
create trigger approval\_update  
after update  
on ride\_bid  
for each row  
execute procedure on\_approval\_update\_pax()  
;  
​  
create or replace function capacity\_checker() returns trigger  
language plpgsql  
as $$  
BEGIN  
  IF ( SELECT (r.current\_pax + NEW.no\_pax <= m.capacity)  
   FROM ride r  
  inner join car c on r.reg\_no = c.reg\_no  
  INNER JOIN model m on c.make = m.make and c.model =m.model  
   AND r.reg\_no = NEW.reg\_no  
   AND r.start\_time = NEW.start\_time)  
    THEN  
    RETURN NEW;  
ELSE  
RAISE EXCEPTION 'Exceeded maximum capacity, please reduce your number of passenger';  
    END IF;  
END  
$$  
;  
​  
​  
create trigger cap\_check  
before insert  
on ride\_bid  
for each row  
execute procedure capacity\_checker()  
;  
​  
create or replace function audit() returns trigger  
language plpgsql  
as $$  
BEGIN  
IF NEW.status = 'completed' THEN  
 INSERT INTO audit\_log(start\_time,end\_time,status,current\_pax,destination,origin,reg\_no)  
 VALUES (OLD.start\_time,now(),NEW.status,OLD.current\_pax,OLD.destination,OLD.origin,OLD.reg\_no);  
END IF;  
​  
RETURN NEW;  
END;  
$$  
;  
​  
create trigger to\_audit  
before update  
on ride  
for each row  
execute procedure audit()  
;

### Sample SQL

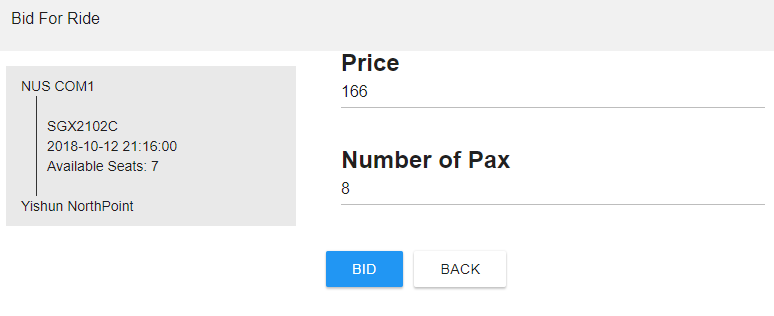
### Search Ride

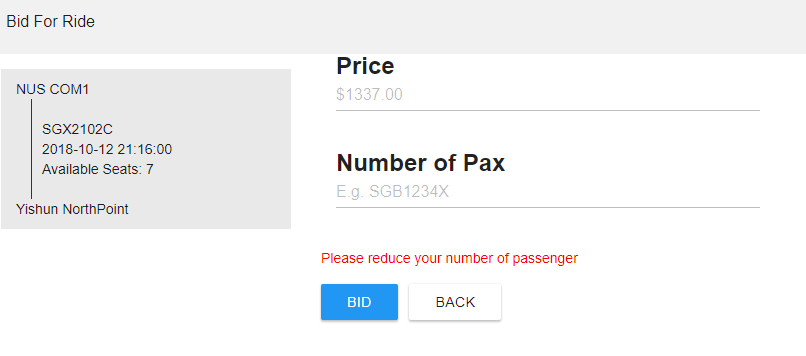
If origin and destination is NULL, return all rides

SELECT u.first\_name,u.email,r.origin,r.destination,r.status,r.reg\_no, r.start\_time, r.current\_pax, (m.capacity - r.current\_pax) as pax\_left,  
EXISTS (SELECT c1.email FROM car c1 WHERE c1.reg\_no = c.reg\_no AND c1.email = %s) as is\_driver,  
EXISTS (SELECT rb.email FROM ride\_bid rb WHERE rb.reg\_no = c.reg\_no AND rb.email = %s AND rb.status = 'successful' AND rb.start\_time = r.start\_time) as has\_success\_bid,  
EXISTS (SELECT rb.email FROM ride\_bid rb WHERE rb.reg\_no = c.reg\_no AND rb.email = %s AND rb.status = 'unsuccessful' AND rb.start\_time = r.start\_time) as has\_unsuccessful\_bid,  
EXISTS (SELECT rb.email FROM ride\_bid rb WHERE rb.reg\_no = c.reg\_no AND rb.email = %s AND rb.status = 'pending' AND rb.start\_time = r.start\_time) as has\_pending\_bid  
FROM ride r, "user" u, car c,model m  
WHERE r.reg\_no = c.reg\_no  
and c.email = u.email  
and LOWER(r.origin) LIKE LOWER(%s) and LOWER(r.destination) like LOWER(%s)  
and r.status = 'in progress'  
and c.make = m.make  
and c.model = m.model  
ORDER BY r.start\_time ASC

**Rejection of ride bid**

When the car ride available capacity is below what the user is bidding for, the system will reject the transaction via trigger





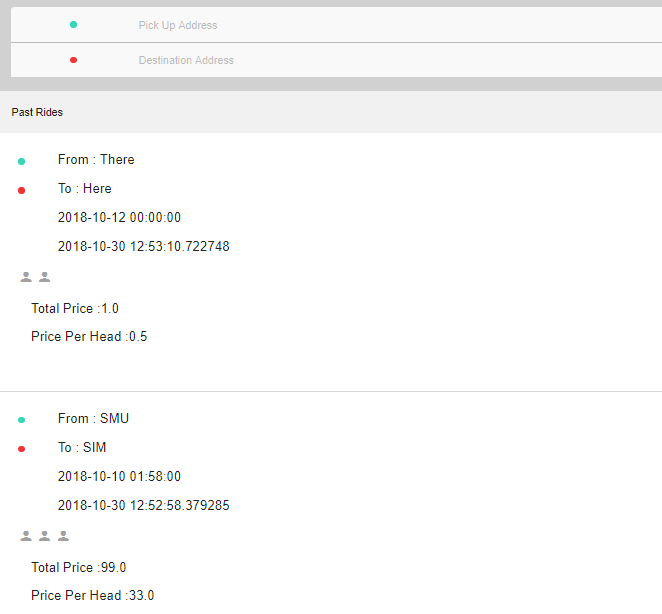
**Trigger behind assert**

create or replace function capacity\_checker() returns trigger  
language plpgsql  
as $$  
BEGIN  
  IF ( SELECT (r.current\_pax + NEW.no\_pax <= m.capacity)  
   FROM ride r  
  inner join car c on r.reg\_no = c.reg\_no  
  INNER JOIN model m on c.make = m.make and c.model =m.model  
   AND r.reg\_no = NEW.reg\_no  
   AND r.start\_time = NEW.start\_time)  
    THEN  
    RETURN NEW;  
ELSE  
RAISE EXCEPTION 'Exceeded maximum capacity, please reduce your number of passenger';  
    END IF;  
END  
$$  
;  
create trigger cap\_check  
before insert  
on ride\_bid  
for each row  
execute procedure capacity\_checker()  
;

### Simple Insert Query

INSERT INTO ride\_bid (email,start\_time,reg\_no,no\_pax,bid\_price) VALUES (%s,%s,%s,%s,%s)

## Ride History



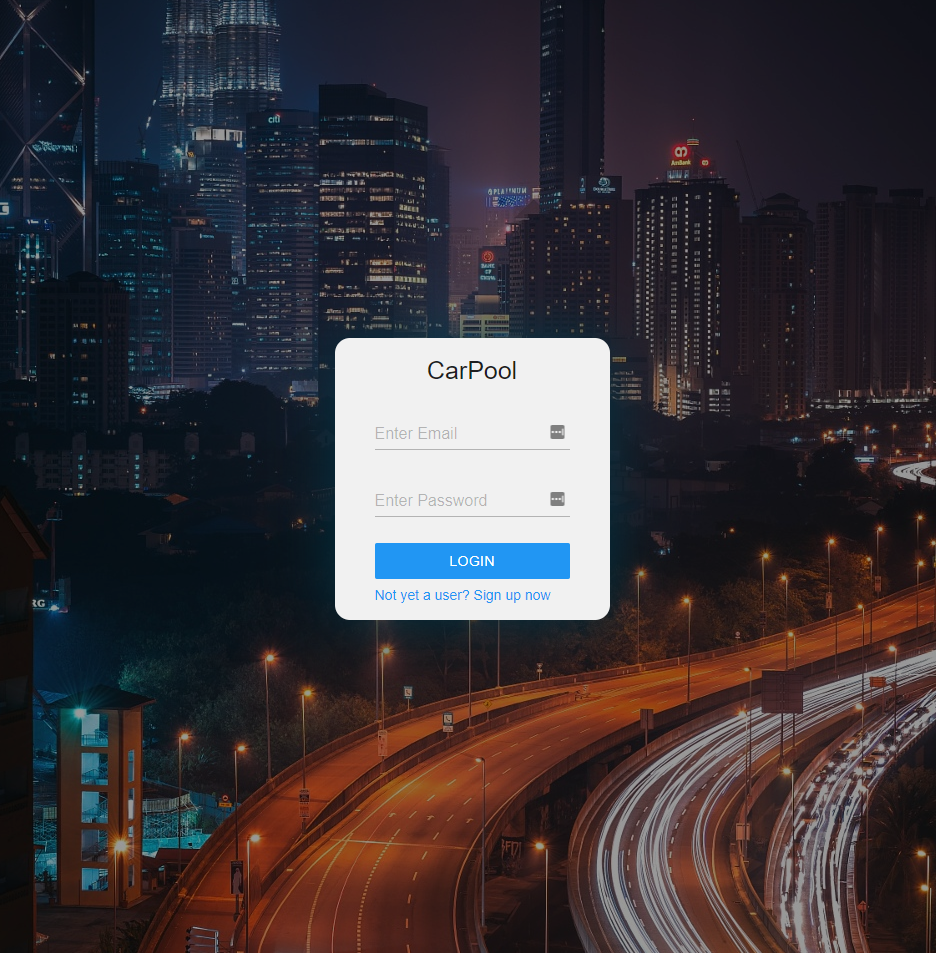
### Trigger

create or replace function audit() returns trigger  
language plpgsql  
as $$  
BEGIN  
IF NEW.status = 'completed' THEN  
 INSERT INTO audit\_log(start\_time,end\_time,status,current\_pax,destination,origin,reg\_no)  
 VALUES (OLD.start\_time,now(),NEW.status,OLD.current\_pax,OLD.destination,OLD.origin,OLD.reg\_no);  
END IF;  
​  
RETURN NEW;  
END;  
$$  
;  
​  
create trigger to\_audit  
before update  
on ride  
for each row  
execute procedure audit()  
;

### Simple Update Query

UPDATE ride  
   SET origin = %s, destination = %s, status = %s  
   WHERE reg\_no = %s AND start\_time = %s

## Screen shots

Login Page:

Index Page:

