CS2102

Project Report

Topic D – Car Pooling

Group 8

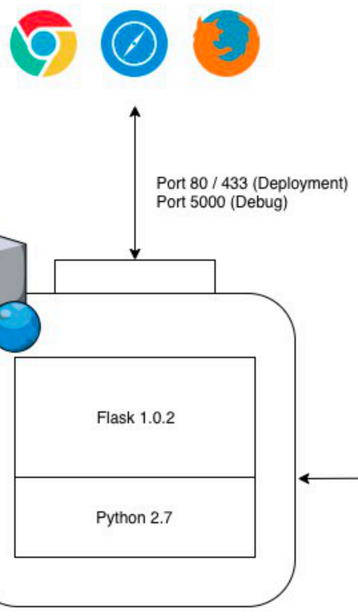
1. Ang Wei Ming, A0168721B
2. Benjamin Chin Choon Kiat, A0168698B
3. Lee Yu Choy, A0177151H
4. Yeo Cheng Hong, A0168369L

6 November 2018

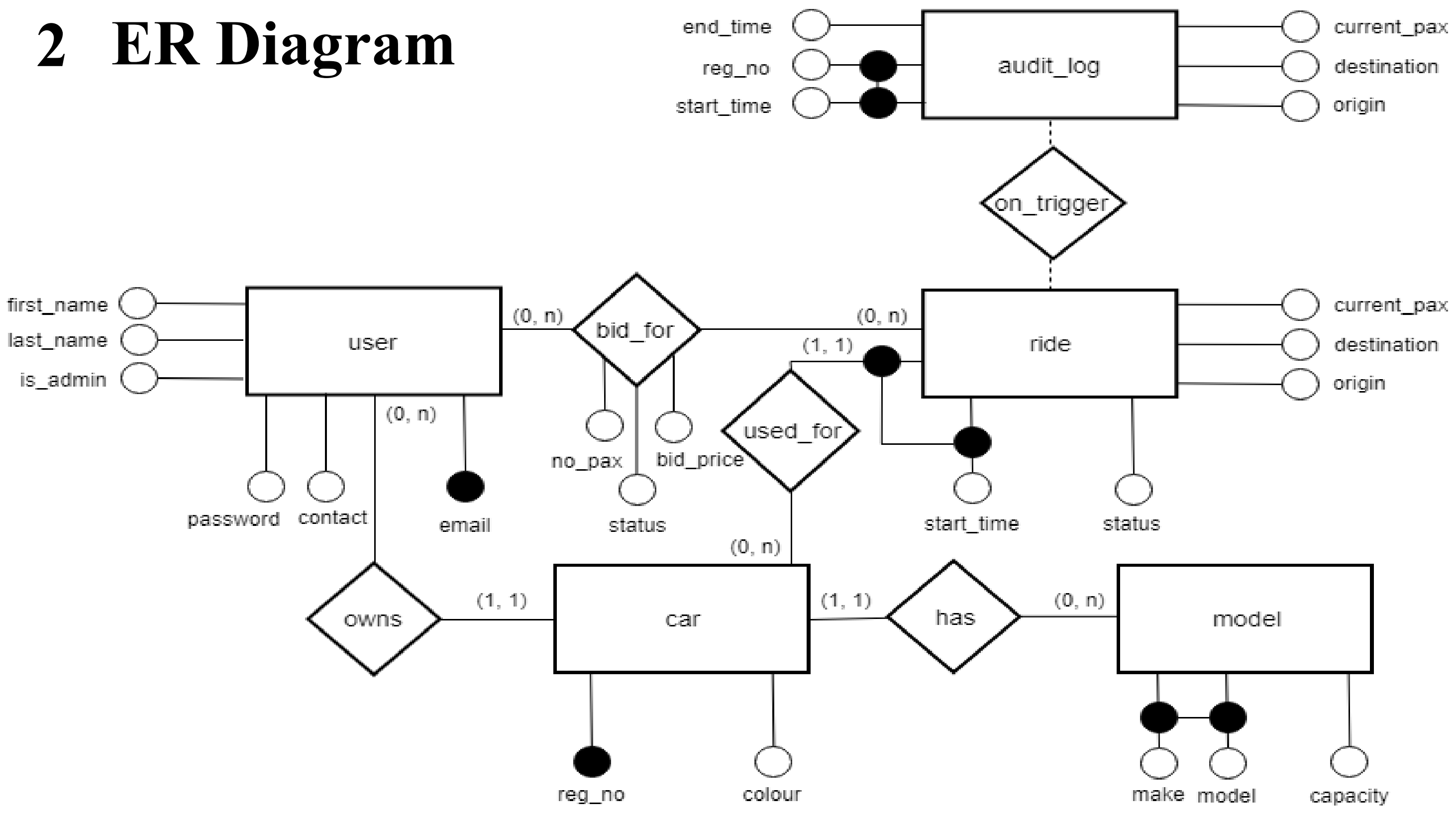
Table Of Content

1 General Architecture

- Server Language – Python 2.7
- Web Server - Python Flask 1.0.2
- Database – PostgreSQL 11.0



2 ER Diagram



The on trigger function for the audit_log table copies over the entry from ride table when an entry status changes from 'in progress' to 'completed'

3 DDL

3.1.1 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
)
;
```

3.1.2 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)
)
;
```

3.1.3 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)
)
;
```

3.1.4 Car 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)
)
;
```

3.1.5 Car 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 check (bid_price > 0),
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
(start_time, reg_no)
)
;
```

3.1.6 Transaction 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)
)
;
```

3.2 Triggers and Functions

3.2.1 After Approval Checks

This trigger will runs when a bid changes from pending/unsuccessful to successful.

When it runs, it will increase the current number of passengers in the ride with the number stated in the ride bid.

Following which, all ‘pending’ bids which have their number of passengers greater than the remaining seats will be rejected.

```
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()
;
```

3.2.2 Constraint checks before bidding.

This function will reject insertion/update if the bids have its numbers of passengers exceed the remaining seats.

It will also reject insertion if the bids is bid by the driver himself.

```
create or replace function capacity_checker()
returns trigger
language plpgsql
as $$
BEGIN
IF (SELECT (r.current_pax + NEW.no_pax <= m.capacity)
     FROM ride r
            inner join car c on r.reg_no = c.reg_no
            INNER JOIN model m on c.make = m.make and c.model = m.model
                                   AND r.reg_no = NEW.reg_no
                                   AND r.start_time = NEW.start_time)
THEN
 --
 -- Do nothing
ELSE
  RAISE EXCEPTION 'Exceeded maximum capacity, please reduce your number of passenger';
END IF;
IF (SELECT (1)
     FROM ride r
            INNER JOIN car c on r.reg_no = c.reg_no
     where r.reg_no = NEW.reg_no
       AND r.start_time = NEW.start_time
       AND c.email = NEW.email)
THEN RAISE EXCEPTION 'Cannot Bid for Own Ride';
ELSE
  RETURN NEW;
END IF;
END
$$;

create trigger cap_check
before insert OR update
on ride_bid
for each row
```

```
execute procedure capacity_checker()
;
```

3.2.3 Insert into audit table.

This trigger is used to insert into a similar table which help in querying transaction history. The end time of the ride will also be updated upon completion.

This extra table although is redundant, but it can help to maintain integrity of the logs, as there will not be any updates in this table. (Only Insertion)

```
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()
;
```

3.2.4 Login / Register Stored Procedure

This function will be called when the user register or login into our web application.

NOTE : Need to run `CREATE EXTENSION` as postgres to install the `pgcrypto` extension.

```
CREATE EXTENSION pgcrypto; -- Have to run as postgres / super user. Run only once.

create or replace function register(email character varying, contact numeric,
first_name character varying, last_name character varying, password character varying)
returns boolean
language plpgsql
as $$
DECLARE
success BOOLEAN;
hashPassword varchar(512);
BEGIN
SELECT encode(digest($5, 'sha256'), 'hex') INTO hashPassword;
INSERT INTO "user" VALUES ($1,$2,$3,$4,false ,hashPassword) Returning 1 into success;
RETURN success;
END;
$$
;

create or replace function login(email character varying, password character varying)
returns SETOF "user"
language plpgsql
as $$
```

```
DECLARE
hashPassword varchar(512);
BEGIN
SELECT encode(digest($2, 'sha256'), 'hex') INTO hashPassword;
RETURN QUERY SELECT * FROM "user" where "user".email = $1 and "user".password =
hashPassword;
END;
$$
;
```

3.3 Sample Function and SQL

3.3.1 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
```

3.3.2 Login / Register with Stored Procedure

The following is the SQL for register and login respectively

```
SELECT REGISTER(%s,%s,%s,%s,%s)
SELECT * from login(%s,%s)
```

3.3.2.1 Stored Procedure for login and register

The stored procedure is used so that the password is hash on the database side before inserting into the table, which improve the security factor.

```
CREATE OR REPLACE FUNCTION register(email varchar(256),contact numeric(8),first_name
varchar(50), last_name varchar(50),password varchar(256))
RETURNS BOOLEAN AS $$
DECLARE
success BOOLEAN;
hashPassword varchar(512);
BEGIN
SELECT encode(digest($5, 'sha256'), 'hex') INTO hashPassword;
INSERT INTO "user" VALUES ($1,$2,$3,$4,false ,hashPassword) Returning 1 into success;
RETURN success;
```

```
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION login(email varchar(256),password varchar(256))
RETURNS setof "user" AS $$
DECLARE
hashPassword varchar(512);
BEGIN
SELECT encode(digest($2, 'sha256'), 'hex') INTO hashPassword;
RETURN QUERY SELECT * FROM "user" where "user".email = $1 and "user".password =
hashPassword;
END;
$$ LANGUAGE plpgsql;
```

3.3.3 Ride History

This allow us to keep track of when is a ride completed. With another table in place, it can help to prevent accidental update. As there are no other query except this trigger that is link to the table.

3.3.3.1 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()
;
```

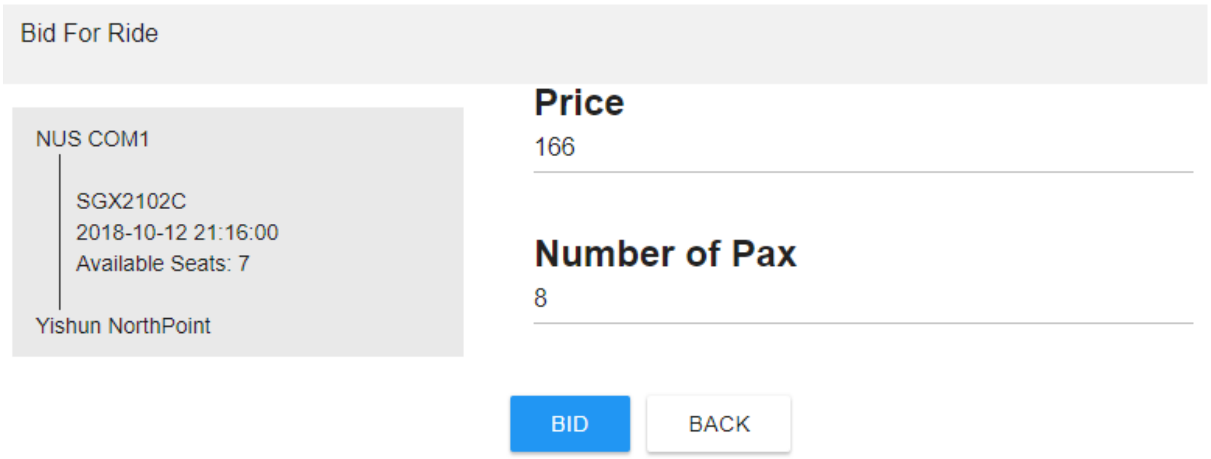
3.3.3.2 Simple Update Query

```
UPDATE ride
   SET origin = %s, destination = %s, status = %s
   WHERE reg_no = %s AND start_time = %s
```

4 Assertion

4.1 Rejection of bid

When the user is bidding for seats which is greater than current available seats, the system will reject the transaction via `trigger`



Bid For Ride

NUS COM1

SGX2102C
2018-10-12 21:16:00
Available Seats: 7

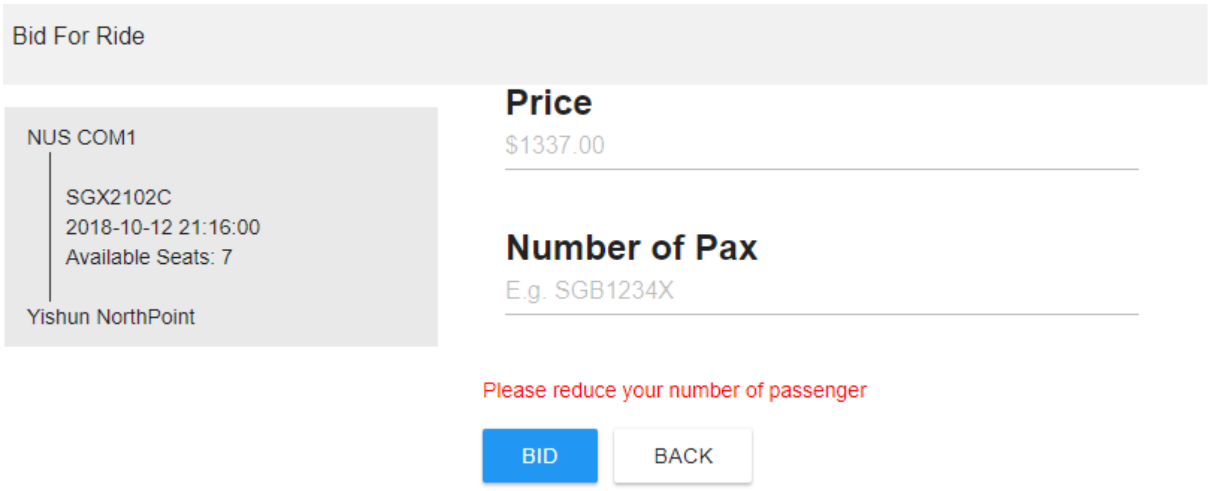
Yishun NorthPoint

Price

166

Number of Pax

8



Bid For Ride

NUS COM1

SGX2102C
2018-10-12 21:16:00
Available Seats: 7

Yishun NorthPoint

Price

$1337.00

Number of Pax

E.g. SGB1234X

Please reduce your number of passenger

4.1.1 Trigger behind

```
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()
;
```

4.1.2 Simple Insert Query

```
INSERT INTO ride_bid (email,start_time,reg_no,no_pax,bid_price) VALUES (%s,%s,%s,%s,%s)
```

4.2 Bid Approval

For every `successful` ride bid that is approved by the `driver` , there will be a trigger to update the current capacity of the ride and reject all pending bid which exceed the current available seats.

4.2.1 Trigger as shown below

```
create trigger approval_update
after update
on ride_bid
for each row
execute procedure on_approval_update_pax()
;
```

4.2.2 on_approval_update_pax function

```
create or replace function on_approval_update_pax() returns trigger
language plpgsql
as $$
BEGIN
IF NEW.status = 'successful' and OLD.STATUS <> 'successful'
THEN
UPDATE ride
SET current_pax = current_pax + NEW.no_pax
WHERE reg_no = NEW.reg_no
AND start_time = NEW.start_time;
-- The following part update all current biddings which have their number of passenger
above the current available seats
-- for example total pax is 4, current pax is 2
-- all bidding which is >2 will be automatically changed to unsuccessful
 UPDATE ride_bid rb
   SET status = 'unsuccessful'
 FROM ride r, car c, model m
 WHERE r.reg_no = rb.reg_no
 AND r.start_time = rb.start_time
 AND r.reg_no = c.reg_no
 AND c.make = m.make
 AND c.model = m.model
 AND rb.reg_no = NEW.reg_no
 AND rb.start_time = NEW.start_time
 AND rb.status = 'pending'
 AND rb.no_pax > (m.capacity - r.current_pax);
END IF;
-- return NULL as trigger is called after update
RETURN NULL;
END
$$
;
```

4.3 Rejecting Car Updates

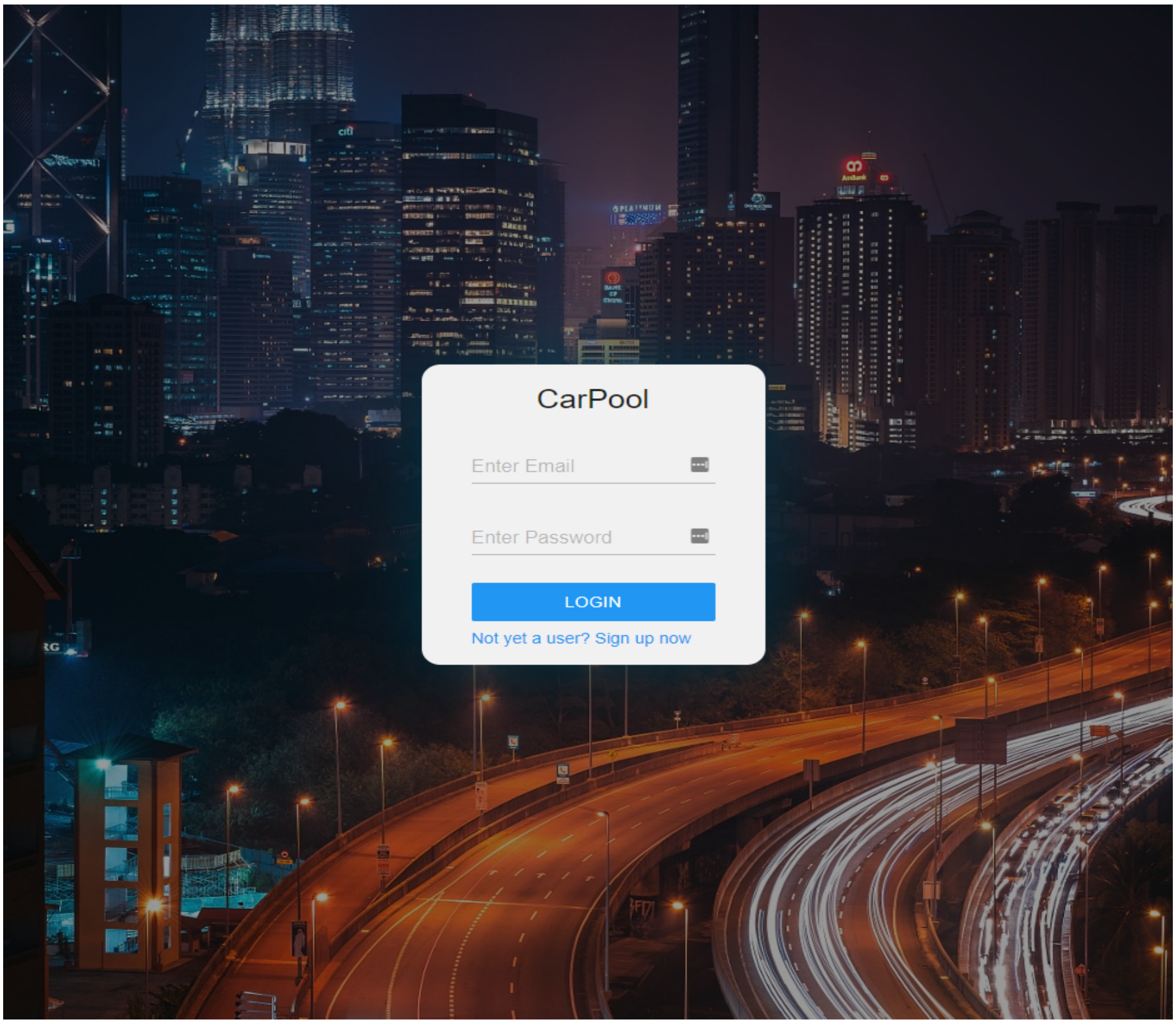
There cannot be any updates to the car entry because if there is a ride in progress with full capacity, changing to another car model might result in not enough seats for the passengers.

4.3.1 Update Query is shown below

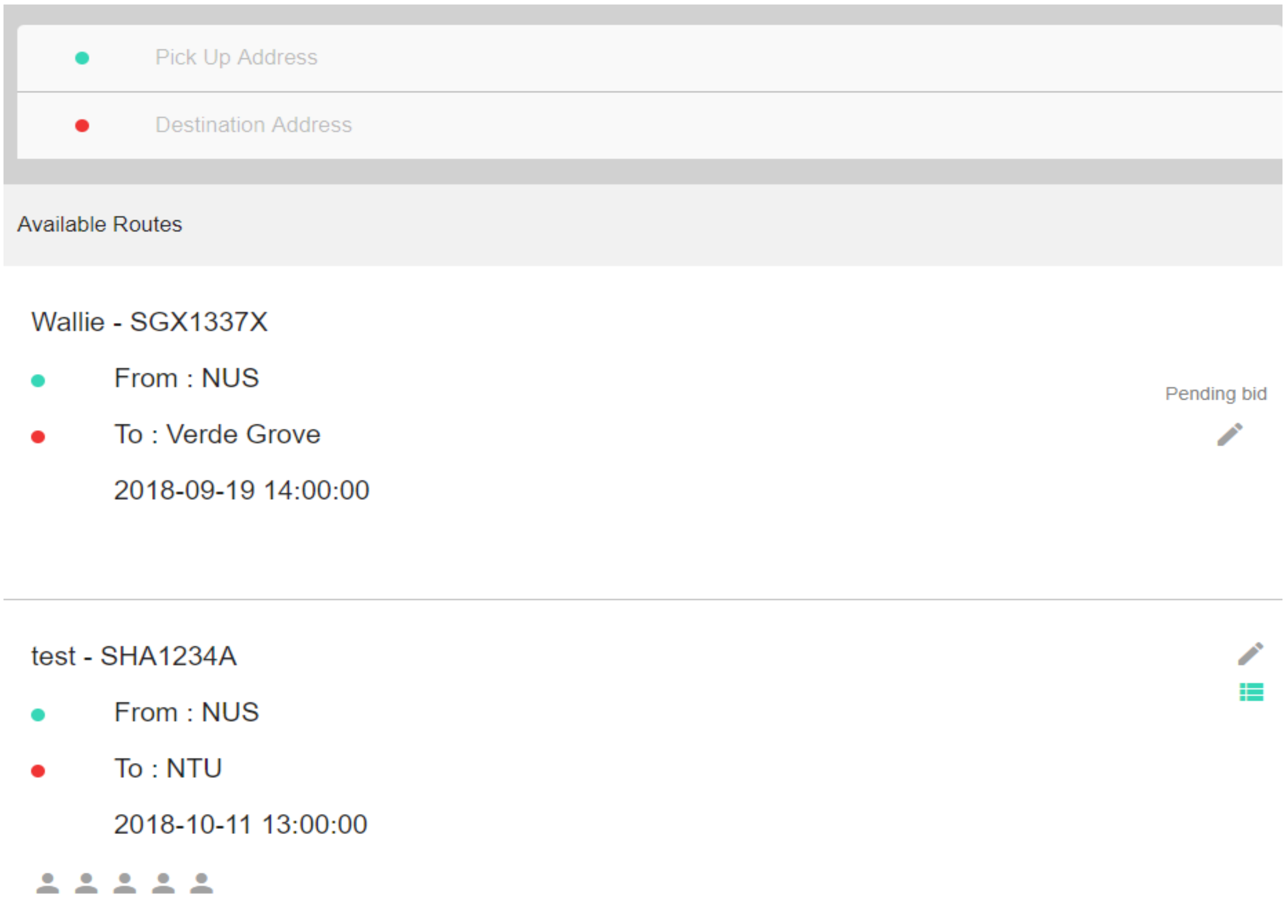
```
UPDATE "car" SET "make" = %s, "model" = %s, "colour" = %s
WHERE "reg_no" = %s AND NOT EXISTS (
        SELECT * FROM "ride" r WHERE r."status" = 'in progress' AND r."reg_no" = %s
) RETURNING "reg_no"
```

5 Screen shots

5.1 Login Page:



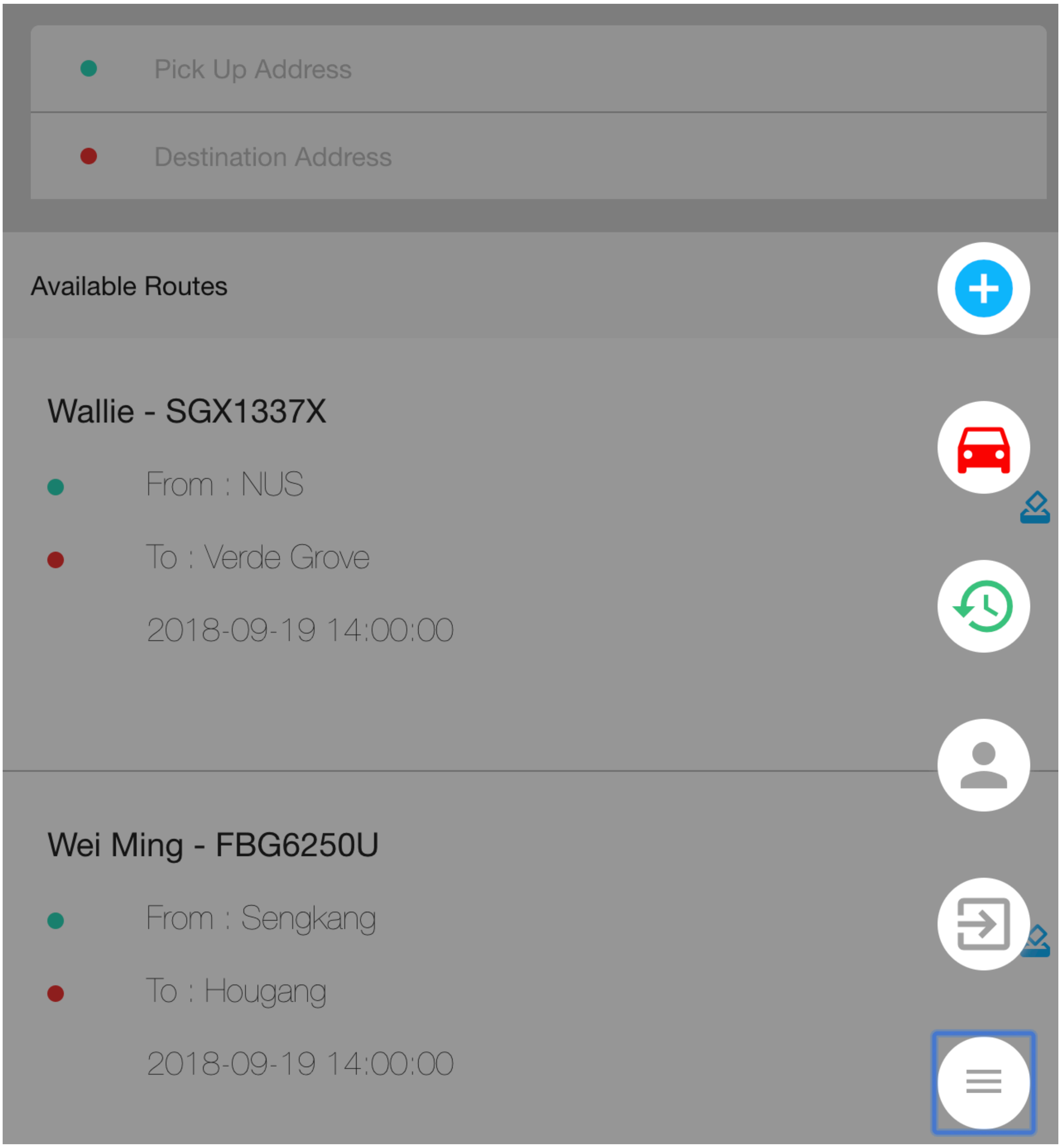
5.2 Index Page/ Home


Pick Up Address
Destination Address
Available Routes

Wallie - SGX1337X
From : NUS
To : Verde Grove
2018-09-19 14:00:00
Pending bid

test - SHA1234A
From : NUS
To : NTU
2018-10-11 13:00:00

5.3 Menu



Pick Up Address

Destination Address

Available Routes

Wallie - SGX1337X

From : NUS

To : Verde Grove

2018-09-19 14:00:00

Wei Ming - FBG6250U

From : Sengkang

To : Hougang

2018-09-19 14:00:00