National University of Singapore

School of Computing

CS2102

Database Systems

PROJECT REPORT

PetCare

Group 18

Chen Penghao (A0122017Y)
Kuang Ming (A0148043L)
Xia Rui (A0148000Y)
Xie Peiyi (A0141123B)

Acknowledge

We would like to thank Associate Professor Bressan Stephane, Professor Lee Mong Li and the teaching team of CS2102 for bringing us into the realm of Database System, for designing this project and for providing guidance in various areas. We would also like to express our appreciation to our tutor, Ms Zheng Kaiping, for her patience in replying to our queries and for giving us timely feedbacks.

CONTENT

1. Introduction

Pets are often treated as important as a member of the family of their owners. However, sometimes pet owners might be unable to take care of their pets for various reasons. It would be a worrying time if the pet was not properly taken care of, such as not timely fed, or the feces not properly cleaned. It would be helpful if the pets could be taken care of by another caregiver, such that the owner would be less worried of their pets' situations.

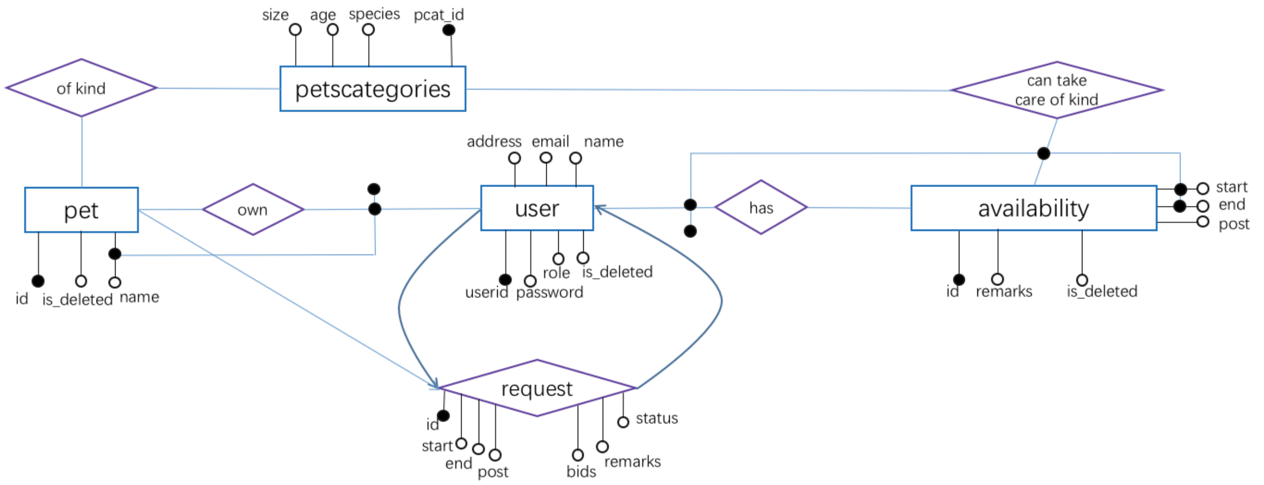
PetCare, our web application, made use of PHP and PostgreSQL to address to this problem, by connecting pet owners and care givers in an interactive and real-time manner.

2. Project Overview

Here are the technical details of our project:

Web server	Bitnami
Server Language	PHP
Database management system used	PostgreSQL
External packages installed	Bootstrap CSS, JQuery

3. Entity Relation Diagram



4. Relational Schema

4.1 Tables

```
CREATE TABLE petcategory(
    pcat_id INT PRIMARY KEY DEFAULT nextval('pcat_seq'),
    age VARCHAR(10),
    size VARCHAR(20),
    species VARCHAR(30)
);

CREATE TABLE pet_user(
    user_id INT PRIMARY KEY DEFAULT nextval('user_id_seq'),
    name VARCHAR(64) NOT NULL,
    password VARCHAR(64) NOT NULL,
    email VARCHAR(64) UNIQUE,
    address VARCHAR(64),
    role VARCHAR(10) DEFAULT 'normal' CONSTRAINT CHK_role CHECK (role in ('admin',
'normal')),
    is_deleted BOOLEAN DEFAULT FALSE
);

CREATE TABLE pet(
    pets_id INT PRIMARY KEY DEFAULT nextval('pets_id_seq'),
    owner_id INT REFERENCES pet_user(user_id) ON DELETE CASCADE,
    pcat_id INT REFERENCES petcategory(pcat_id) ON DELETE CASCADE ON UPDATE CASCADE,
    pet_name VARCHAR(64),
    is_deleted BOOLEAN DEFAULT FALSE,
    UNIQUE (owner_id, pet_name)
);

CREATE TABLE availability(
    avail_id INT PRIMARY KEY DEFAULT nextval('avail_id_seq'),
    post_time timestamp NOT NULL DEFAULT current_timestamp,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP NOT NULL,
    pcat_id INT REFERENCES petcategory(pcat_id) ON DELETE CASCADE ON UPDATE CASCADE,
    taker_id INT REFERENCES pet_user(user_id) ON DELETE CASCADE,
    remarks VARCHAR(64) DEFAULT 'No',
    is_deleted BOOLEAN DEFAULT FALSE,
    UNIQUE (start_time, end_time, pcat_id, taker_id),
    CONSTRAINT CHK_start_end CHECK (end_time > start_time),
    CONSTRAINT CHK_post CHECK (start_time > post_time)
);

CREATE TABLE request(
    request_id INT PRIMARY KEY DEFAULT nextval('request_id_seq'),
    owner_id INT REFERENCES pet_user(user_id) ON DELETE CASCADE,
    taker_id INT REFERENCES pet_user(user_id) ON DELETE CASCADE,
    post_time TIMESTAMP NOT NULL DEFAULT current_timestamp,
    care_begin TIMESTAMP NOT NULL,
    care_end TIMESTAMP NOT NULL,
    remarks VARCHAR(64) DEFAULT 'No',
    bids NUMERIC NOT NULL,
    pets_id INT REFERENCES pet(pets_id) ON DELETE CASCADE ON UPDATE CASCADE,
    slot VARCHAR(64),
    totaltime DOUBLE PRECISION,
    status VARCHAR(20) CHECK (status IN ('pending', 'failed', 'successful', 'cancelled'))
DEFAULT 'pending',
    CONSTRAINT CHK_start_end CHECK (care_end > care_begin),
    CONSTRAINT CHK_post CHECK (care_begin > post_time)
);

CREATE VIEW requesttime AS
    SELECT SUM(r.bids)/SUM(r.totaltime)*60 AS avgbids, r.taker_id AS taker_id
    FROM request r
    WHERE r.status = 'successful'
    GROUP BY r.taker_id;
```

4.2 Functions & Triggers

```sql
--According to the timing of the inserted request entry, set the slot attribute
--to the corresponding value

CREATE OR REPLACE FUNCTION timeslot(requestNum INTEGER)
RETURNS VARCHAR(64) AS $$
DECLARE slot VARCHAR(64); hours DOUBLE PRECISION; beginTime timestamp;
BEGIN
SELECT care_begin INTO beginTime FROM request WHERE request_id = requestNum;
hours = extract(HOUR FROM (beginTime));
IF hours BETWEEN 6 AND 11 THEN slot = 'Morning';
ELSE IF hours BETWEEN 12 AND 17 THEN slot = 'Afternoon';
ELSE IF hours BETWEEN 18 AND 23 THEN slot = 'Evening';
ELSE slot = 'Before Dawn';
END IF;
END IF;
END IF;
RETURN slot;
END; $$
LANGUAGE PLPGSQL;

--Calculate total length of time period of the request entry inserted

CREATE OR REPLACE FUNCTION calculateTotalTime(requestNum INTEGER)
RETURNS DOUBLE PRECISION AS $$
DECLARE totalmins DOUBLE PRECISION; days DOUBLE PRECISION; hours DOUBLE PRECISION; mins
DOUBLE PRECISION;
startTime timestamp; endTime timestamp;
BEGIN
SELECT care_begin, care_end INTO startTime, endTime FROM request WHERE request_id =
requestNum;
mins = extract(MINUTE FROM (endTime - startTime));
days = extract(DAY FROM (endTime - startTime));
hours = extract(HOUR FROM (endTime - startTime));
totalmins = mins + 60 * (hours + 24 * days);
RETURN totalmins;
END; $$
LANGUAGE PLPGSQL;

--Add time slot and total time attribute to the request entry

CREATE OR REPLACE FUNCTION addRequestInfo()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE request
    SET slot= timeslot(new.request_id), totaltime = calculateTotalTime(new.request_id)
    WHERE request_id = new.request_id;
    RETURN NULL;
END; $$
LANGUAGE PLPGSQL;

--Trigger that activate slot and totaltime adding methods when inserting new request
--entry

CREATE TRIGGER addSlot
AFTER INSERT
ON request
FOR EACH ROW
EXECUTE PROCEDURE addRequestInfo();
```

```sql
--Take out the availability slots that has end_time already before current time, set
--is_deleted to TRUE

CREATE OR REPLACE FUNCTION cleanOutdatedAvail()
RETURNS TRIGGER AS $$
BEGIN(
  UPDATE availability SET is_deleted = TRUE
  WHERE end_time <= CURRENT_TIMESTAMP
  AND is_deleted = FALSE;
  RETURN NULL;
END; $$
LANGUAGE PLPGSQL;

--Take out the request entries that have no corresponding availability slots to hold the
--request, as well as the end time already before current time, set status to be failed.

CREATE OR REPLACE FUNCTION cleanOutdatedAndNotMatchingReq()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE request
  SET status = 'cancelled'
  WHERE (care_begin <= CURRENT_TIMESTAMP
  AND status = 'pending')
  OR (request_id NOT IN (SELECT r.request_id
                      FROM request r INNER JOIN pet p ON r.pets_id = p.pets_id
                                     INNER JOIN availability a ON a.pcat_id = p.pcat_id
                      WHERE r.taker_id = a.taker_id
                      AND a.is_deleted = FALSE      AND p.is_deleted = FALSE
                      AND r.care_end <= a.end_time  AND r.care_begin >= a.start_time)
  AND status = 'pending');
  RETURN NULL;
END; $$
LANGUAGE PLPGSQL;

--Activate the cleaning function on availability after each insertion

CREATE TRIGGER changeAvail
AFTER INSERT ON availability
FOR EACH STATEMENT
EXECUTE PROCEDURE cleanOutdatedAvail();

--Activate the cleaning function on request after each insertion

CREATE TRIGGER changeReq
AFTER INSERT ON request
FOR EACH STATEMENT
EXECUTE PROCEDURE cleanOutdatedAndNotMatchingReq();

--View that is used in request page to short list the averate bids

CREATE VIEW requesttime AS
    SELECT SUM(r.bids)/SUM(r.totaltime)*60 AS avgbids, r.taker_id AS taker_id
    FROM request r
    WHERE r.status = 'successful'
    GROUP BY r.taker_id;
```

5. SQL Code Snippets & Screenshot of Webpages

5.1 Pet Owner Page

- Show all ongoing/unsuccessful/pending request:
- r.status = ‘successful’/ ‘failed’/ ‘pending’ accordingly in ongoing/unsuccessful/pending

```
SELECT u.name, u.email, r.care_begin, r.care_end, r.bids, p.pet_name
FROM request r, pet_user u, pet p
WHERE r.owner_id = $user_id AND r.status = 'successful'
      AND r.care_end > current_timestamp AND r.taker_id = u.user_id
      AND r.pets_id = p.pets_id AND p.is_deleted = false
ORDER BY care_begin;
```

- Actions for cancelling pending request and reading unsuccessful requests:

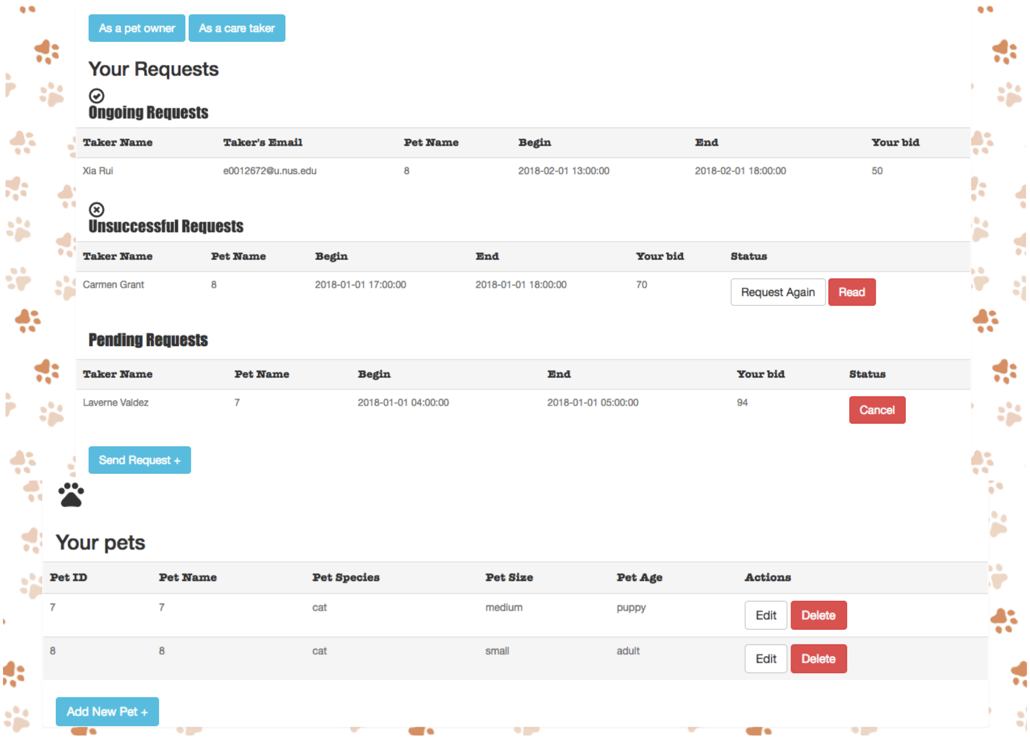
```
UPDATE request SET status = 'cancelled' WHERE request_id =$request_id;
```

- Show information for pets:

```
SELECT p.pets_id, p.name, c.species, c.size, c.age FROM pet p, petcategory c WHERE
p.owner_id =$user_id p.pcat_id = c.pcat_id AND p.is_deleted=false ORDER BY pets_id;
```

- Update/Add information for pets:

```
UPDATE pet SET pcat_id = $pcat_id, pet_name = '$pet_name' WHERE pets_id = $pet_id;
INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES ($pcat_id,$user_id,'$pet_name');
```



Your Requests

Ongoing Requests

Taker Name	Taker's Email	Pet Name	Begin	End	Your bid
Xia Rui	e0012672@u.nus.edu	8	2018-02-01 13:00:00	2018-02-01 18:00:00	50

Unsuccessful Requests

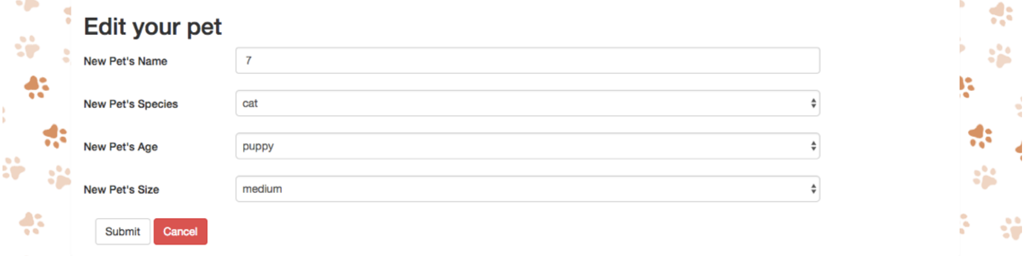
Taker Name	Pet Name	Begin	End	Your bid	Status
Carmen Grant	8	2018-01-01 17:00:00	2018-01-01 18:00:00	70	Request Again Read

Pending Requests

Taker Name	Pet Name	Begin	End	Your bid	Status
Laverne Valdez	7	2018-01-01 04:00:00	2018-01-01 05:00:00	94	Cancel

Your pets

Pet ID	Pet Name	Pet Species	Pet Size	Pet Age	Actions
7	7	cat	medium	puppy	Edit Delete
8	8	cat	small	adult	Edit Delete



Edit your pet

New Pet's Name	7
New Pet's Species	cat
New Pet's Age	puppy
New Pet's Size	medium

5.2 Pet Taker Page

- Show all ongoing/pending request:
- r.status = ‘successful’ or ‘pending’ accordingly in ongoing/pending requests

```
SELECT r.request_id, u.name, u.email, r.care_begin, r.care_end,
       r.remarks, r.bids, p.pet_name, c.age, c.size, c.species
FROM request r, pet p, petcategory c, pet_user u
WHERE r.taker_id = $user_id AND r.status = 'pending'
      AND r.care_begin > CURRENT_TIMESTAMP AND p.pets_id = r.pets_id
      AND p.pcat_id = c.pcat_id AND u.user_id = r.owner_id
      AND p.is_deleted = false
ORDER BY r.bids DESC; (ORDER BY r.care_begin; in ongoing request)
```

- Reject/Accept:

```
UPDATE request SET status = 'failed' WHERE request_id =$reject_id;
UPDATE request SET status = 'successful' WHERE request_id =$accept_id;
```

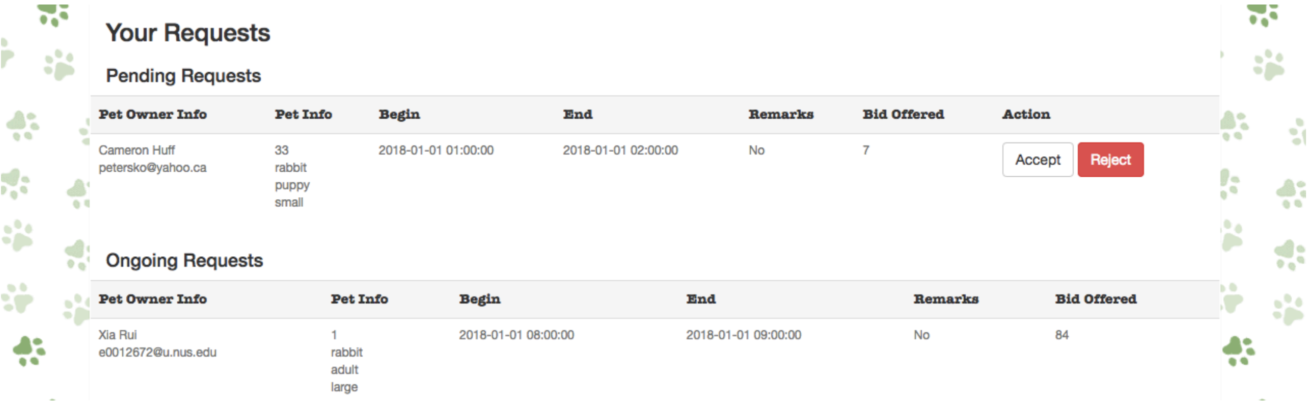
- Before Accept, check #overlap request:

AGGREGATE

```
SELECT COUNT(*) FROM request r1,request r2
WHERE r1.request_id = $accept_id AND r2.taker_id = $user_id
      AND r2.status = 'successful'
      AND r1.care_begin < r2.care_end AND r1.care_end > r2.care_begin;
```

- After Accept, cancel all request with same pet and time overlap:

```
UPDATE request SET status = 'cancelled'
WHERE request_id <> $accept_id AND pets_id = $pets_id
      AND '$start' < care_end AND '$end' > care_begin;
```



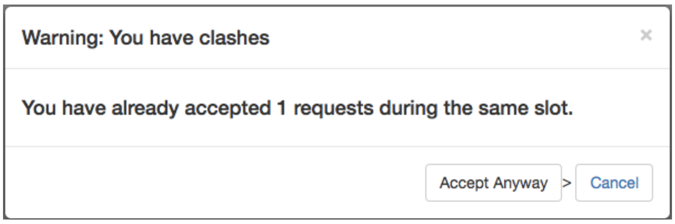
Your Requests

Pending Requests

Pet Owner Info	Pet Info	Begin	End	Remarks	Bid Offered	Action
Cameron Huff petersko@yahoo.ca	33 rabbit puppy small	2018-01-01 01:00:00	2018-01-01 02:00:00	No	7	Accept Reject

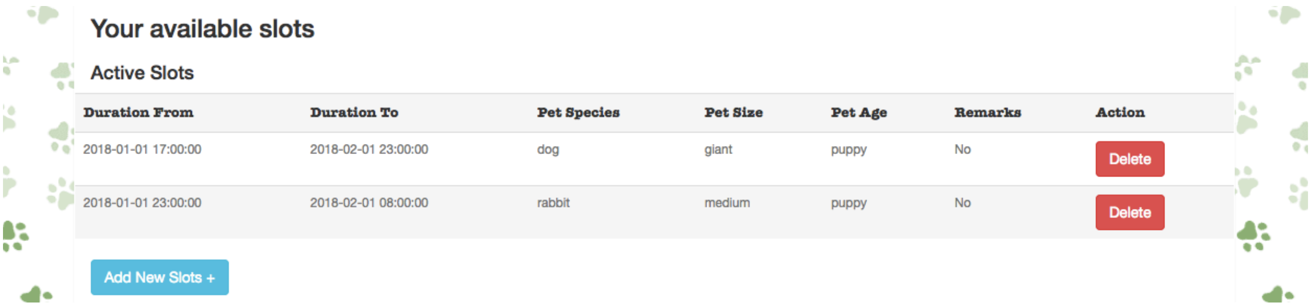
Ongoing Requests

Pet Owner Info	Pet Info	Begin	End	Remarks	Bid Offered
Xia Rui e0012672@u.nus.edu	1 rabbit adult large	2018-01-01 08:00:00	2018-01-01 09:00:00	No	84



Warning: You have clashes

You have already accepted 1 requests during the same slot.



Your available slots

Active Slots

Duration From	Duration To	Pet Species	Pet Size	Pet Age	Remarks	Action
2018-01-01 17:00:00	2018-02-01 23:00:00	dog	giant	puppy	No	Delete
2018-01-01 23:00:00	2018-02-01 08:00:00	rabbit	medium	puppy	No	Delete

- Show all available slot

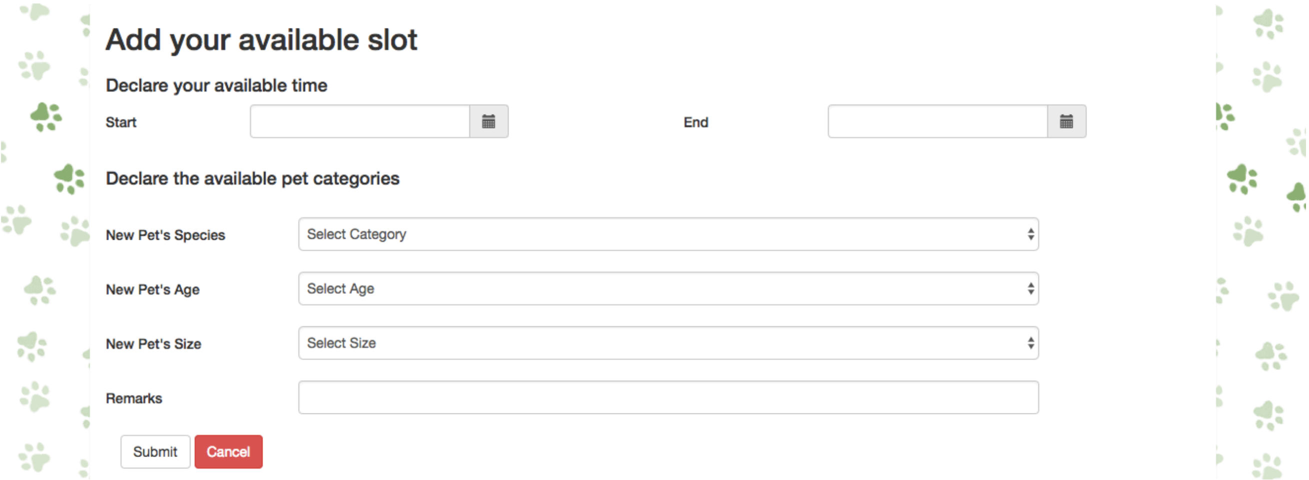
```
SELECT a.avail_id, a.start_time, a.end_time, a.remarks, p.species,
       p.size, p.age
FROM availability a, petcategory p
WHERE a.pcat_id = p.pcat_id AND a.taker_id =$user_id
      AND a.is_deleted = FALSE AND a.start_time > CURRENT_TIMESTAMP
ORDER BY a.start_time;
```

- Add/delete available slot

```
INSERT INTO availability(start_time, end_time, pcat_id, taker_id, remarks)
                   VALUES ('$start_time', '$end_time', $pcat_id, $user_id, '$remarks');
UPDATE availability SET is_deleted=true WHERE avail_id=" . $avail_id . ";
```

- After deletion of available slot, all related requests’ status changed to failed

```
UPDATE request SET status='failed' WHERE status='pending' AND request_id NOT IN(
        SELECT r.request_id FROM request r,availability a,pet p
        WHERE a.pcat_id=p.pcat_id AND r.pets_id=p.pets_id AND a.is_deleted=false
              AND r.taker_id=" . $user_id . " AND a.taker_id=" . $user_id . "
              AND a.start_time<=r.care_begin AND r.care_end<=a.end_time);
```



Add your available slot

Declare your available time

Start

End

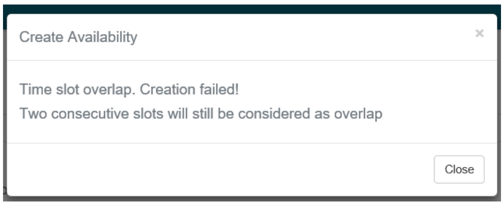
Declare the available pet categories

New Pet's Species: Select Category

New Pet's Age: Select Age

New Pet's Size: Select Size

Remarks



Create Availability

Time slot overlap. Creation failed!
Two consecutive slots will still be considered as overlap

- Cannot create available slot with time overlap

5.3 Send Request

LEFT JOIN

USING VIEW

- Search without any constraint:

```
SELECT a.avail_id, a.start_time, a.end_time, a.taker_id, p.name,
       (CASE WHEN t.avgbids is NULL THEN 0 ELSE t.avgbids END) AS avgbids,
       a.remarks
FROM (availability a INNER JOIN pet_user p ON p.user_id = a.taker_id
      AND a.is_deleted = FALSE AND p.is_deleted = FALSE)
      LEFT JOIN requesttime AS t ON a.taker_id = t.taker_id
WHERE a.taker_id <> '$user_id'
```

- When pet/start-time/end-time specified:

```
AND a.pcat_id = $pcat_id
AND a.start_time <= '$start_time'
AND a.end_time >= '$end_time'
```

- When all the above three specified:

```
AND a.taker_id NOT IN (SELECT r.taker_id FROM request r WHERE r.care_end >
                       '$start_time' AND r.care_begin < '$end_time' AND r.pets_id
                       = $pet_id AND r.status='pending')
```

- When preferred taker_name specified:

```
AND UPPER(p.name) LIKE UPPER('%$taker_name%')
```

- When average bids/hour range specified:

```
AND (t.avgbids <= $upperbound OR t.avgbids is NULL)
AND (t.avgbids >= $lowerbound OR t.avgbids is NULL)
```

AGGREGATE

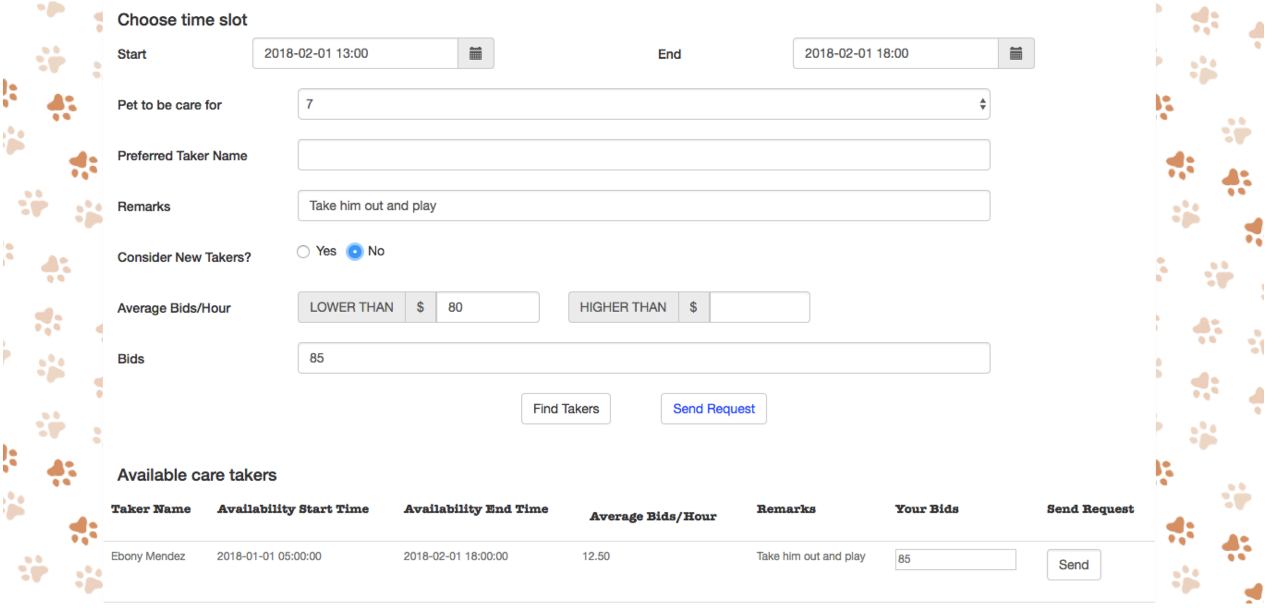
- When owner do not want new takers: `AND t.avgbids is NOT NULL`
- Added in the end: `ORDER BY avgbids ASC;`

- Before sending requests, check the pet has not been taken care of in the specific period:

```
SELECT * FROM request r WHERE r.care_begin <'$end_time' AND r.care_end >
'$start_time' AND r.pets_id = $pet_id AND r.status = 'successful';
```

- Finally send the request:

```
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids,
pets_id) VALUES ($user_id, $taker_id, '$start_time', '$end_time', '$remarks',
$bids, $pet_id);
```



Choose time slot

Start	2018-02-01 13:00	End	2018-02-01 18:00
Pet to be care for	7		
Preferred Taker Name			
Remarks	Take him out and play		
Consider New Takers?	○ Yes ● No		
Average Bids/Hour	LOWER THAN $ 80	HIGHER THAN $	
Bids	85		

Available care takers

Taker Name	Availability Start Time	Availability End Time	Average Bids/Hour	Remarks	Your Bids	Send Request
Ebony Mendez	2018-01-01 05:00:00	2018-02-01 18:00:00	12.50	Take him out and play	85	Send

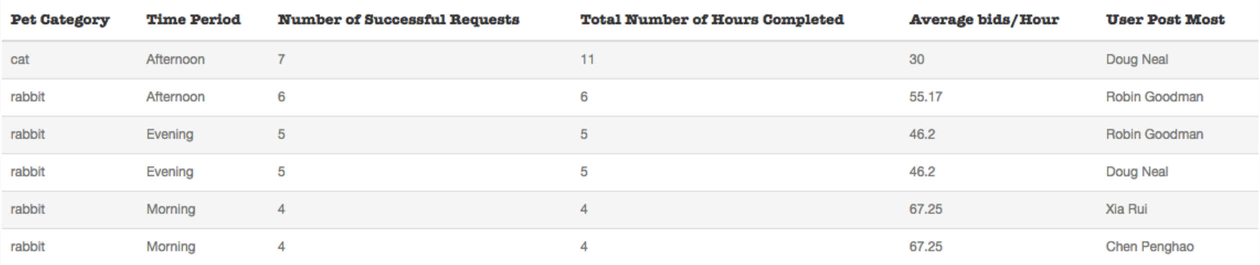
5.4.1 Admin-stats

- Group all the successful request by timeslots(morning/afternoon/evening) and pet species
- For each group, find total #successful request, average bids/hour, and users that posts most

```
SELECT k.species, k.timeslot, k.RequestNum, k.average, r1.owner_id, k.totaltime
FROM (SELECT c.species AS species, r.slot AS timeslot,
             COUNT(r.request_id) AS RequestNum,
             (SUM(r.totaltime)/60) AS totaltime,
             (SUM(r.bids)/SUM(r.totaltime)*60) AS average
      FROM petcategory c, pet p, request r
      WHERE r.pets_id = p.pets_id AND c.pcat_id = p.pcat_id
            AND r.status = 'successful'
      GROUP BY r.slot, c.species) AS k, request r1, petcategory c1, pet p1
WHERE r1.pets_id = p1.pets_id AND c1.pcat_id = p1.pcat_id
      AND r1.status = 'successful' AND c1.species = k.species
      AND r1.slot = k.timeslot
GROUP BY r1.owner_id, k.species, k.timeslot, k.RequestNum, k.average, k.totaltime
HAVING COUNT(*) >= ALL( SELECT COUNT(*)
                        FROM request r2, petcategory c2, pet p2
                        WHERE r2.pets_id = p2.pets_id AND c2.pcat_id = p2.pcat_id
                              AND r2.status = 'successful'
                              AND c2.species = k.species AND r2.slot = k.timeslot
                        GROUP BY r2.owner_id)
ORDER BY k.RequestNum DESC;
```

AGGREGATE

NESTED



Pet Category	Time Period	Number of Successful Requests	Total Number of Hours Completed	Average bids/Hour	User Post Most
cat	Afternoon	7	11	30	Doug Neal
rabbit	Afternoon	6	6	55.17	Robin Goodman
rabbit	Evening	5	5	46.2	Robin Goodman
rabbit	Evening	5	5	46.2	Doug Neal
rabbit	Morning	4	4	67.25	Xia Rui
rabbit	Morning	4	4	67.25	Chen Penghao

- Find the average bids/hour… of takers who have taken care of all pet species

```
SELECT u.name, (SUM(r1.bids)/SUM(r1.totaltime)*60) AS average, SUM(r1.totaltime)
FROM request r1, pet_user u
WHERE r1.taker_id = u.user_id AND r1.status = 'successful'
      AND NOT EXISTS (SELECT c1.species
                      FROM petcategory c1
                      WHERE NOT EXISTS (SELECT *
                                        FROM request r2, pet p, petcategory c2
                                        WHERE r2.taker_id = r1.taker_id
                                              AND r2.pets_id = p.pets_id
                                              AND p.pcat_id = c2.pcat_id
                                              AND c2.species = c1.species
                                              AND r2.status = 'successful'))
GROUP BY r1.taker_id, u.name
ORDER BY average DESC;
```

AGGREGATE

NESTED

AGGREGATE

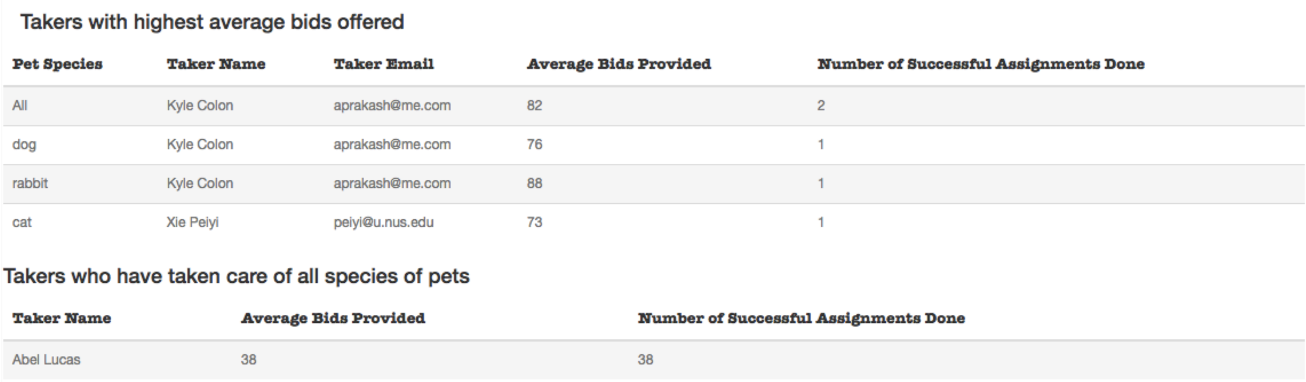
- For all takers, find the takers with highest average bids/hour

```
SELECT u.name, u.email, k.average, k.num
FROM (SELECT r.taker_id AS id, (SUM(r.bids)/SUM(r.totaltime)*60) AS average,
             (SUM(r.totaltime)/60) AS num
     FROM request r WHERE r.status = 'successful'
     GROUP BY r.taker_id) AS k, pet_user u
WHERE u.user_id = k.id AND NOT EXISTS(SELECT *
                                      FROM (SELECT
                                           (SUM(r1.bids)/SUM(r1.totaltime)*60) AS
                                            avg FROM request r1
                                      GROUP BY r1.taker_id) AS k1
                                      WHERE k.average < k1.avg);
```

NESTED

- For every species, find the takers with highest average bids/hour

```
SELECT k.species, u.name, u.email, k.average, k.num
FROM (SELECT r.taker_id AS id, (SUM(r.bids)/SUM(r.totaltime)*60) AS average,
             (SUM(r.totaltime)/60) AS num, c.species AS species
      FROM request r, pet p, petcategory c
      WHERE r.pets_id = p.pets_id AND p.pcat_id = c.pcat_id
            AND r.status = 'successful'
      GROUP BY c.species, r.taker_id) AS k, pet_user u
WHERE u.user_id = k.id AND NOT EXISTS(
                               SELECT * FROM (SELECT
                              (SUM(r1.bids)/SUM(r1.totaltime)*60) AS avg
                               FROM request r1, pet p1, petcategory c1
                               WHERE r1.pets_id = p1.pets_id
                                     AND p1.pcat_id = c1.pcat_id
                                     AND c1.species = k.species
                                     AND r1.status = 'successful'
                               GROUP BY r1.taker_id) AS k1
                               WHERE k.average < k1.avg);
```



Takers with highest average bids offered

Pet Species	Taker Name	Taker Email	Average Bids Provided	Number of Successful Assignments Done
All	Kyle Colon	aprakash@me.com	82	2
dog	Kyle Colon	aprakash@me.com	76	1
rabbit	Kyle Colon	aprakash@me.com	88	1
cat	Xie Peiyi	peiyi@u.nus.edu	73	1

Takers who have taken care of all species of pets

Taker Name	Average Bids Provided	Number of Successful Assignments Done
Abel Lucas	38	38

5.4.2 Admin usage for availability entries

- Generate the views for the availability

```
SELECT a.avail_id, a.post_time, a.start_time, a.end_time,u.user_id, u.name,
       a.is_deleted, pc.age, pc.size, pc.species, pc.pcat_id, a.remarks
FROM availability a INNER JOIN pet_user u ON a.taker_id = u.user_id
                    INNER JOIN petcategory pc ON a.pcat_id = pc.pcat_id
```

- If the button ‘Show Delete’ is activated:

```
WHERE a.is_deleted = ‘t’
```

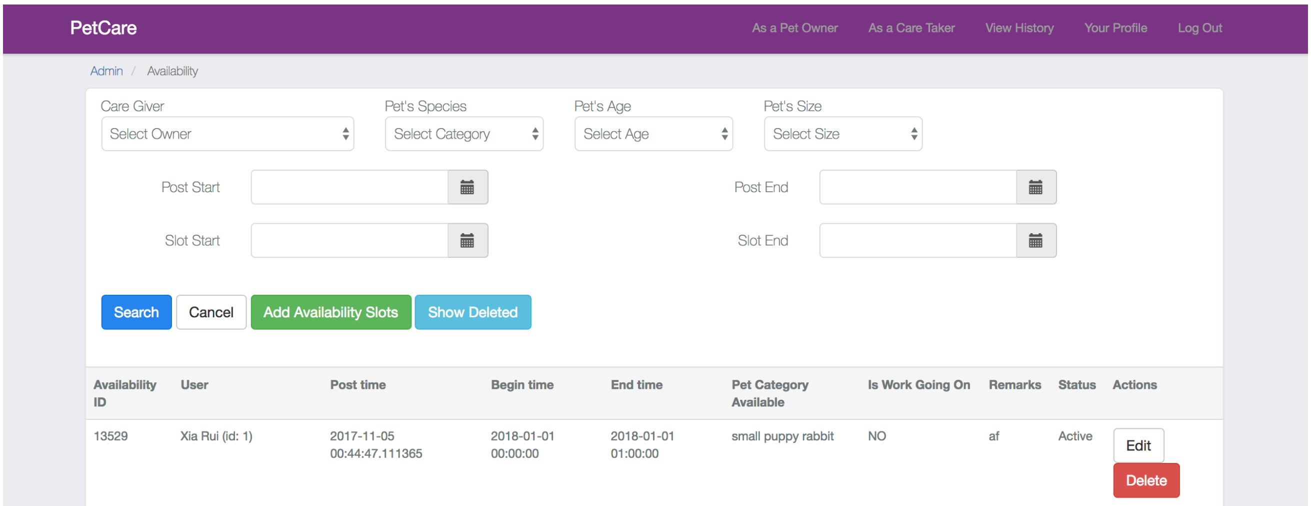
- Else:

```
WHERE a.is_deleted = ‘f’
```

- Lastly, order by the availability id

```
ORDER BY a.avail_id
```

INNER JOIN



Care Giver: Select Owner
Pet's Species: Select Category
Pet's Age: Select Age
Pet's Size: Select Size
Post Start:
Post End:
Slot Start:
Slot End:

Availability ID	User	Post time	Begin time	End time	Pet Category Available	Is Work Going On	Remarks	Status	Actions
13529	Xia Rui (id: 1)	2017-11-05 00:44:47.111365	2018-01-01 00:00:00	2018-01-01 01:00:00	small puppy rabbit	NO	af	Active	Edit Delete

INNER JOIN

- Check whether there are ongoing requests in the availability slot

```
SELECT r.request_id
FROM request r INNER JOIN availability a ON (r.taker_id = a.taker_id)
               INNER JOIN pet p ON (r.pets_id = p.pets_id)
               INNER JOIN petcategory pc ON (pc.pcat_id = a.pcat_id
                                      AND pc.pcat_id = p.pcat_id)
WHERE r.care_begin >= a.start_time AND r.care_end <= a.end_time
      AND r.status = 'successful' AND a.avail_id = ".$a_id. "
      AND a.pcat_id = ".$row[10];
```

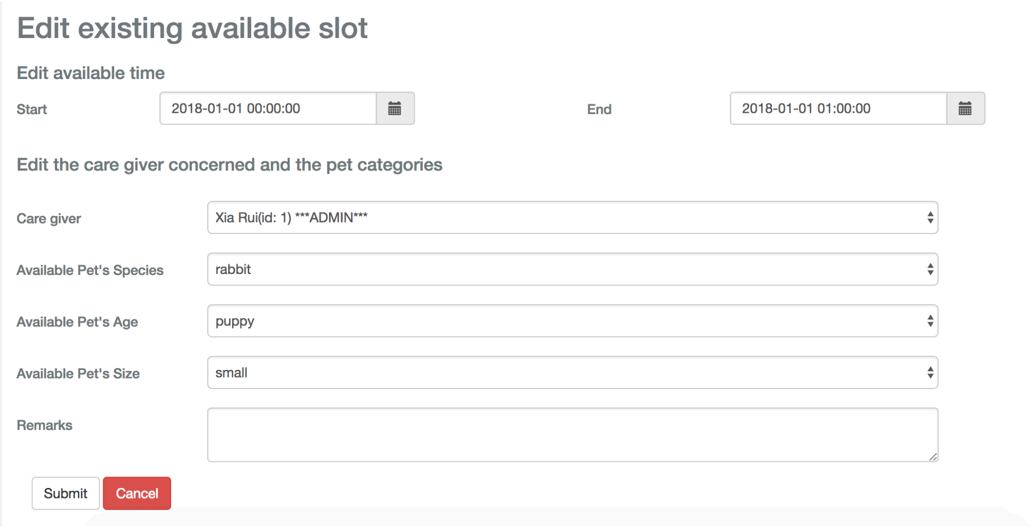
- Checking is done to make sure that the pet in the request entry matches the pet category of the availability, as well as the begin and end time of the request is nested in the availability slot, and the care giver is the same care giver for the slot.

- Update of availability slots:
- First need to check if there is overlapping slots, by checking if there exists slots that have end time after the start time of the new slot, and also the start time of such slots are before the end time of the new slot.

```
SELECT avail_id FROM availbility a
WHERE a.start_time <= '" . $a_end . "' AND a.end_time >= '" . $a_start . "'
      AND a.pcat_id = " . $pcat_id . " AND a.taker_id = " . $a_uid . ";
```

- Then carry out the updating of availbility

```
UPDATE availability
SET start_time = '" . $a_start . "', end_time = '" . $a_end . "',
    pcat_id = $pcat_id, taker_id = $a_uid, remarks = '".$a_remarks."'
WHERE avail_id = $avail_id;
```



Edit existing available slot

Edit available time

Start 2018-01-01 00:00:00

End 2018-01-01 01:00:00

Edit the care giver concerned and the pet categories

Care giver: Xia Rui(id: 1) ***ADMIN***

Available Pet's Species: rabbit

Available Pet's Age: puppy

Available Pet's Size: small

Remarks:

- Delete availability slots
  - First set the value of is_deleted of the corresponding slot to true

```
UPDATE availability SET is_deleted=true WHERE avail_id=" . $a_id . ";
```

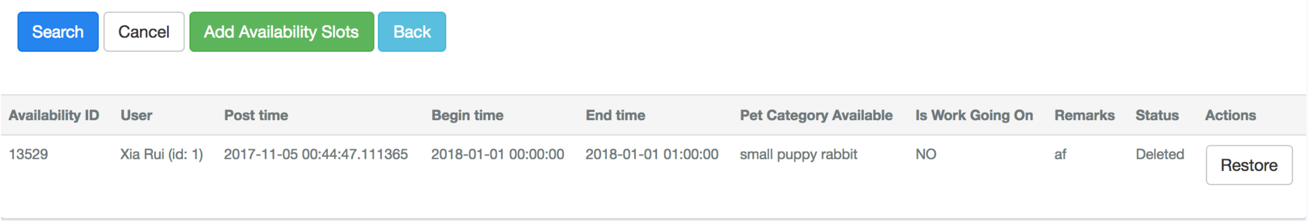
  - Then check if there are requests, by the same care giver, time periods nested in the deleted availability slots, and the category of the pet concerned matches that of the availability slot deleted. Update the status of these request to be 'failed'.

```
UPDATE request SET status='failed'
WHERE status='pending'
AND care_begin >= ALL (SELECT a.start_time FROM availability a
                       WHERE a.avail_id = " . $a_id . ")
AND care_end <= ALL (SELECT a.end_time FROM availability a
                     WHERE a.avail_id = " . $a_id . ")
AND taker_id = ALL (SELECT a.taker_id FROM availability a
                    WHERE a.avail_id = " . $a_id . ")
AND pets_id IN (SELECT p.pets_id
                FROM availability a INNER JOIN pet p ON a.pcat_id = p.pcat_id
                WHERE a.avail_id = " . $a_id . ")
```

NESTED

- Restore availability slots
- By simply updating the availability table, setting the attribute is_deleted from true to false. However, the affected requests are not restored after this availability is restored.

```
UPDATE request SET status='pending' WHERE request_id=" . $r_id . ";
```



Availability ID	User	Post time	Begin time	End time	Pet Category Available	Is Work Going On	Remarks	Status	Actions
13529	Xia Rui (id: 1)	2017-11-05 00:44:47.111365	2018-01-01 00:00:00	2018-01-01 01:00:00	small puppy rabbit	NO	af	Deleted	Restore

5.4.3 Admin usage for users entries

- Generate view of users
- Basic query:

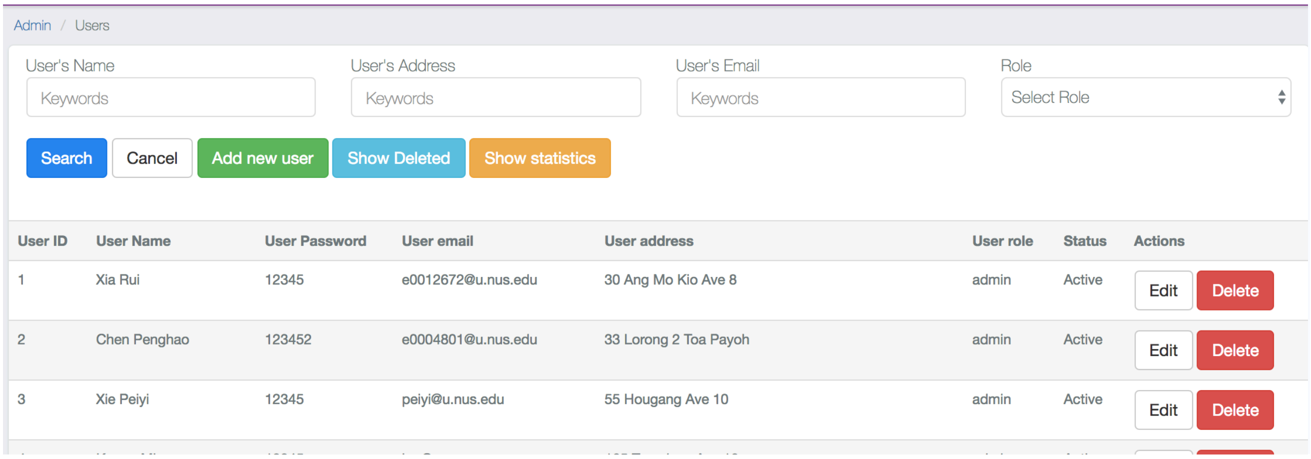
```
SELECT u.user_id,u.name,u.password,u.email,u.address,u.role,u.is_deleted FROM pet_user u
```

- Depending on if the show_deleted button is clicked, the attribute is_deleted is set to true if the button is in place, false if not.

```
WHERE u.is_deleted = " . (isset($_GET['show_deleted']) ? "true" : "false") .
```

- Then order the entries by their user id

```
ORDER BY u.user_id;
```



User ID	User Name	User Password	User email	User address	User role	Status	Actions
1	Xia Rui	12345	e0012672@u.nus.edu	30 Ang Mo Kio Ave 8	admin	Active	Edit Delete
2	Chen Penghao	123452	e0004801@u.nus.edu	33 Lorong 2 Toa Payoh	admin	Active	Edit Delete
3	Xie Peiyi	12345	peiyi@u.nus.edu	55 Hougang Ave 10	admin	Active	Edit Delete

- When deleting certain users, it generates the greatest spread out effect, such that a lot of other entries need to be deleted as well.
    - First we delete the user:

```
UPDATE pet_user SET is_deleted=true WHERE user_id=" . $u_id . ";
```

-
    - Any availability slots declared by the user must be delted as well:

```
UPDATE availability SET is_deleted=true WHERE taker_id=$u_id;
```

-
    - Any requests where the user acts as either owner or care giver must be deleted

```
UPDATE request SET status='failed'
WHERE status='pending' AND (owner_id=$u_id OR taker_id=$u_id);
```

-
    - Any pets owned by this owner must be deleted as well

```
UPDATE pet SET is_deleted=true WHERE owner_id=$u_id;
```

-
    - Any request sent out related to this pet must be set to failed as well

```
UPDATE request SET status='failed'
WHERE status='pending'
AND pets_id IN (SELECT pets_id FROM pet WHERE owner_id = $u_id);
```

- After generating the basic information view, we generate views of the statistics of users using the following query:

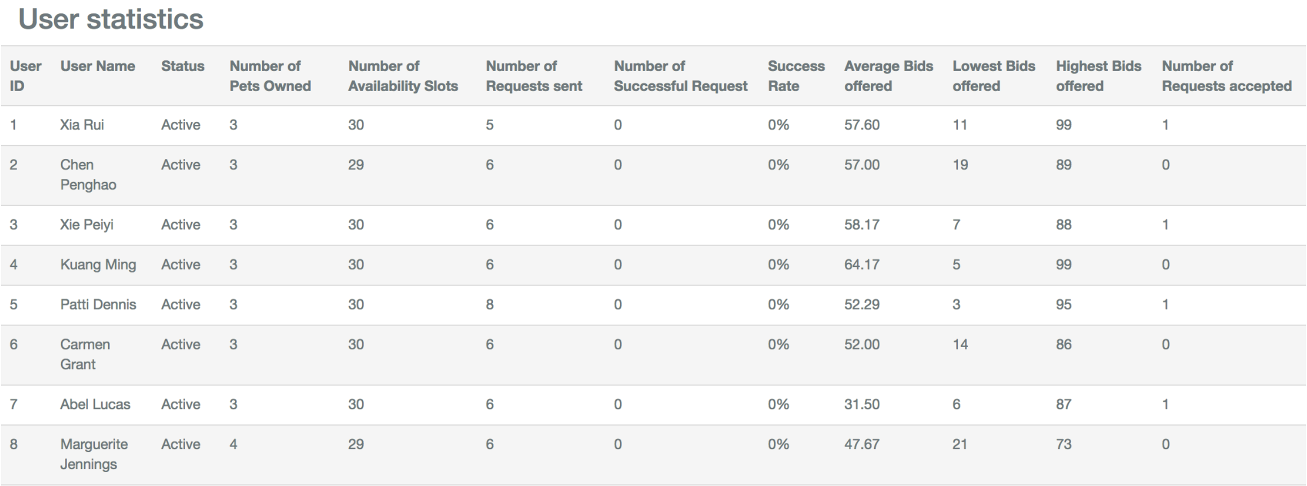
AGGREGATE

LEFT JOIN

```
SELECT u.user_id,
       COUNT(DISTINCT p.pets_id),
       COUNT(DISTINCT a.avail_id),
       COUNT(DISTINCT r1.request_id),
       COUNT(DISTINCT r2.request_id),
       COALESCE(COUNT(DISTINCT r2.request_id)::DECIMAL/NULLIF(COUNT(DISTINCT
r1.request_id),0),-1),
       COALESCE(ROUND(AVG(DISTINCT r1.bids),2),0),
       COALESCE(MIN(r1.bids),0),
       COALESCE(MAX(r1.bids),0),
       COUNT(DISTINCT r3.request_id)
FROM pet_user u LEFT OUTER JOIN pet p ON (p.owner_id = u.user_id)
                LEFT OUTER JOIN availability a ON (a.taker_id = u.user_id)
                LEFT OUTER JOIN request r1 ON (r1.owner_id = u.user_id)
                LEFT OUTER JOIN request r2 ON (r2.owner_id = u.user_id
                                          AND r2.status = 'successful')
                LEFT OUTER JOIN request r3 ON (r3.taker_id = u.user_id
                                          AND r3.status = 'successful')
GROUP BY u.user_id
ORDER BY u.user_id
```

- The outcomes of this query corresponds to :
  - the total number of pets a user is owning;
  - total number of availability slots the user has declared;
  - total number of requests sent as a pet owner;
  - number of successfully accepted requests by other care takers;
  - overall successful rate
  - average bids put
  - minimum bids put
  - maximum bids put
  - total number of successfully done or started requests as care giver

  left outer join is in place to pad in NULL value for those users who have not completed a request or never sent a request; COALESCE function is in place to convert NULL value to certain number values, so that we could process it when creating the table view.



User statistics

User ID	User Name	Status	Number of Pets Owned	Number of Availability Slots	Number of Requests sent	Number of Successful Request	Success Rate	Average Bids offered	Lowest Bids offered	Highest Bids offered	Number of Requests accepted
1	Xia Rui	Active	3	30	5	0	0%	57.60	11	99	1
2	Chen Penghao	Active	3	29	6	0	0%	57.00	19	89	0
3	Xie Peiyi	Active	3	30	6	0	0%	58.17	7	88	1
4	Kuang Ming	Active	3	30	6	0	0%	64.17	5	99	0
5	Patti Dennis	Active	3	30	8	0	0%	52.29	3	95	1
6	Carmen Grant	Active	3	30	6	0	0%	52.00	14	86	0
7	Abel Lucas	Active	3	30	6	0	0%	31.50	6	87	1
8	Marguerite Jennings	Active	4	29	6	0	0%	47.67	21	73	0

5.4.4 Admin usage for pet categories

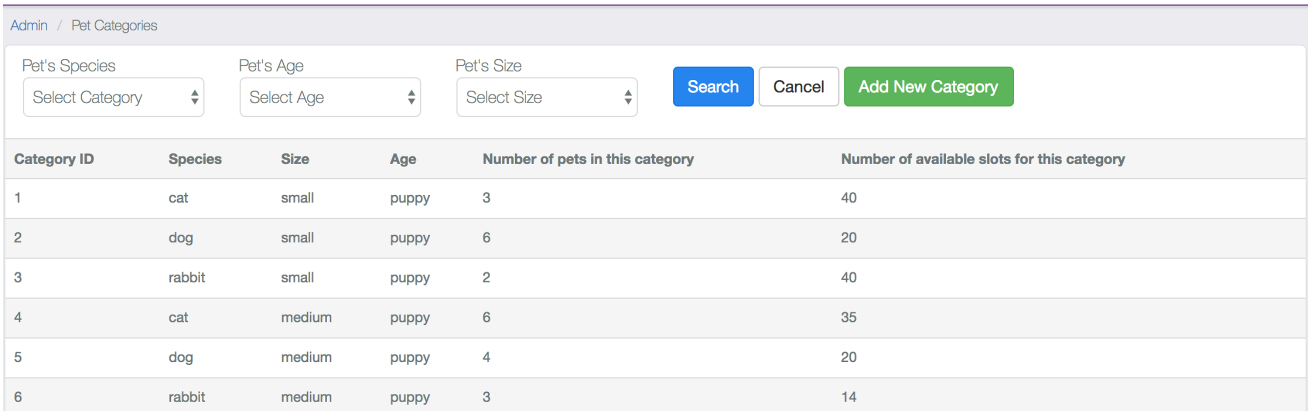
- Generate views of pet categories, while counting number of pets id and availability id present in the pet category

```
SELECT pc.pcat_id, pc.age, pc.size, pc.species,
       COUNT(DISTINCT p.pets_id), COUNT(DISTINCT a.avail_id)
FROM petcategory pc LEFT JOIN pet p ON p.pcat_id= pc.pcat_id
LEFT JOIN availability a ON a.pcat_id = pc.pcat_id
GROUP BY pc.pcat_id
ORDER BY pc.pcat_id
```

AGGREGATE

- Adding new pet category:

```
INSERT INTO petcategory(age,size,species)
VALUES ('$pcat_age','$pcat_size','$pcat_species');
```



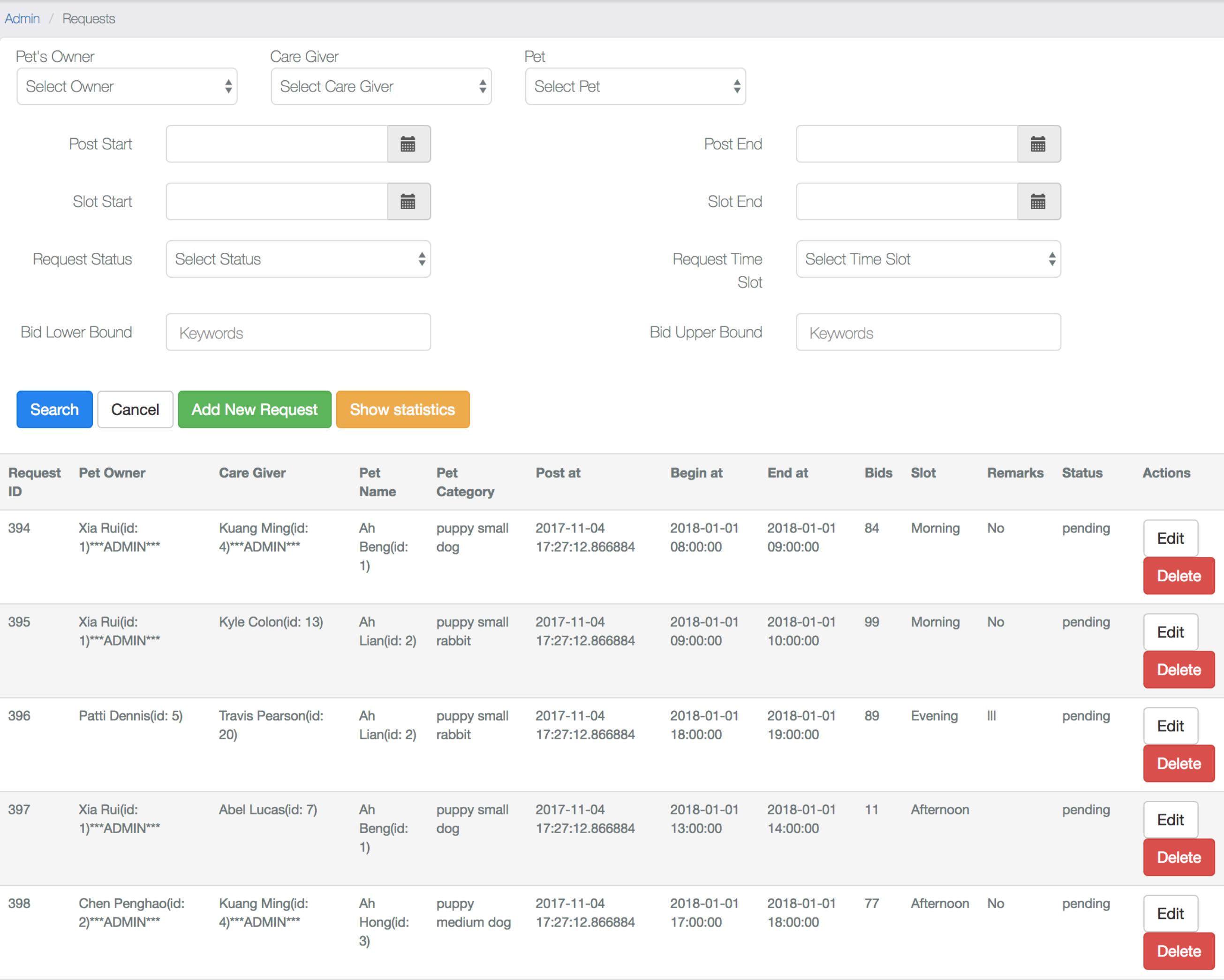
Category ID	Species	Size	Age	Number of pets in this category	Number of available slots for this category
1	cat	small	puppy	3	40
2	dog	small	puppy	6	20
3	rabbit	small	puppy	2	40
4	cat	medium	puppy	6	35
5	dog	medium	puppy	4	20
6	rabbit	medium	puppy	3	14

- We don’t allow update and deletion of pet category entries, since it is a rather static database and would create a lot of complications especially after one category of pet is deleted. Special treatment should be in place if it were really necessary to change the category database.

5.4.5 Admin usage for requests

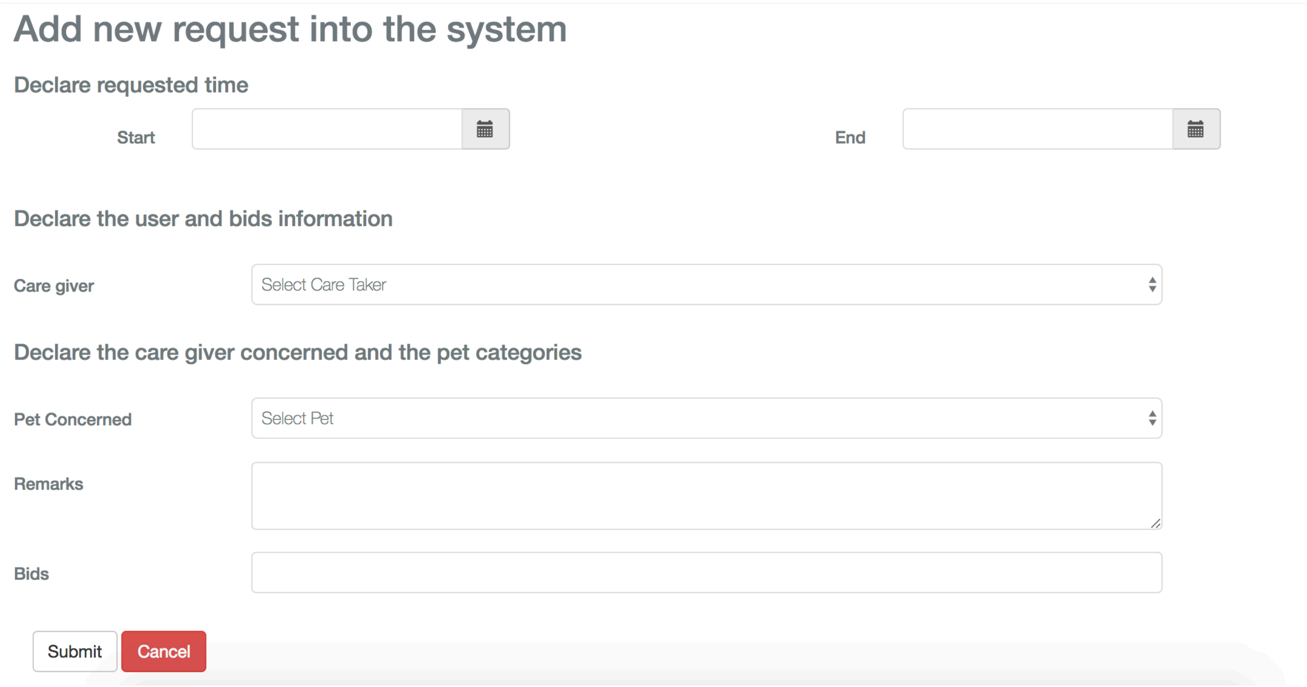
- Generating view of request, where depending on if the admin user is checking the deleted entries by clicking the “Show Deleted” button, the criterion would be different for r.status.

```
SELECT r.request_id, u1.user_id, u1.name, u1.role, u2.user_id, u2.name, u2.role,
       r.post_time, r.care_begin, r.care_end, r.bids, r.remarks, r.slot,r.status,
       p.pets_id, p.pet_name, pc.age, pc.size, pc.species
FROM request r INNER JOIN pet_user u1 ON r.owner_id = u1.user_id
               INNER JOIN pet_user u2 ON r.taker_id = u2.user_id
               INNER JOIN pet p ON r.pets_id = p.pets_id
               INNER JOIN petcategory pc ON p.pcat_id = pc.pcat_id
WHERE r.status " . (isset($_GET['show_deleted']) ? "='failed'"
                     : "IN ('pending', 'successful', 'cancelled')") .
ORDER BY r.request_id;
```



Request ID	Pet Owner	Care Giver	Pet Name	Pet Category	Post at	Begin at	End at	Bids	Slot	Remarks	Status	Actions
394	Xia Rui(id: 1)***ADMIN***	Kuang Ming(id: 4)***ADMIN***	Ah Beng(id: 1)	puppy small dog	2017-11-04 17:27:12.866884	2018-01-01 08:00:00	2018-01-01 09:00:00	84	Morning	No	pending	Edit Delete
395	Xia Rui(id: 1)***ADMIN***	Kyle Colon(id: 13)	Ah Lian(id: 2)	puppy small rabbit	2017-11-04 17:27:12.866884	2018-01-01 09:00:00	2018-01-01 10:00:00	99	Morning	No	pending	Edit Delete
396	Patti Dennis(id: 5)	Travis Pearson(id: 20)	Ah Lian(id: 2)	puppy small rabbit	2017-11-04 17:27:12.866884	2018-01-01 18:00:00	2018-01-01 19:00:00	89	Evening	lll	pending	Edit Delete
397	Xia Rui(id: 1)***ADMIN***	Abel Lucas(id: 7)	Ah Beng(id: 1)	puppy small dog	2017-11-04 17:27:12.866884	2018-01-01 13:00:00	2018-01-01 14:00:00	11	Afternoon		pending	Edit Delete
398	Chen Penghao(id: 2)***ADMIN***	Kuang Ming(id: 4)***ADMIN***	Ah Hong(id: 3)	puppy medium dog	2017-11-04 17:27:12.866884	2018-01-01 17:00:00	2018-01-01 18:00:00	77	Afternoon	No	pending	Edit Delete

- Adding new requests:



Add new request into the system

Declare requested time

Declare the user and bids information

Declare the care giver concerned and the pet categories

- Before actually inserting into the sql table, we do up two checks. The first check is on whether there are overlapped time slots between the same two users and on the same pet:

```
SELECT * FROM request
WHERE care_begin <= $proposed_end AND care_end >= $proposed_start
      AND taker_id = $care_taker AND pets_id = $pet_concerned
```

- The second check is on whether there is an availability slot corresponding to this new request, since the admin user did not go through shortlisting of availability slots advertised by the admin chosen users

```
SELECT a.avail_id
FROM request r INNER JOIN pet p ON r.pets_id = p.pets_id
INNER JOIN availability a ON p.pcat_id = a.pcat_id
WHERE a.start_time <= $proposed_start AND a.end_time >= $proposed_end
      AND a.taker_id = $care_taker AND p.pets_id = $pet_conerned
```

- If we have no results from the first check, and non empty results from the second check, then we can add in the availability slot. Notice that we only apply correspondence on pets_id of the request, since each pet only has one owner, so there is no need to locate the owner_id.
- The insertion query is:

```
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id)
VALUES ($owner_id," . $care_taker . ",'" . $proposed_start . "','" . $proposed_end .
"','" . $remarks . "'," . $bids . "," . $pet_concerned . ");
```

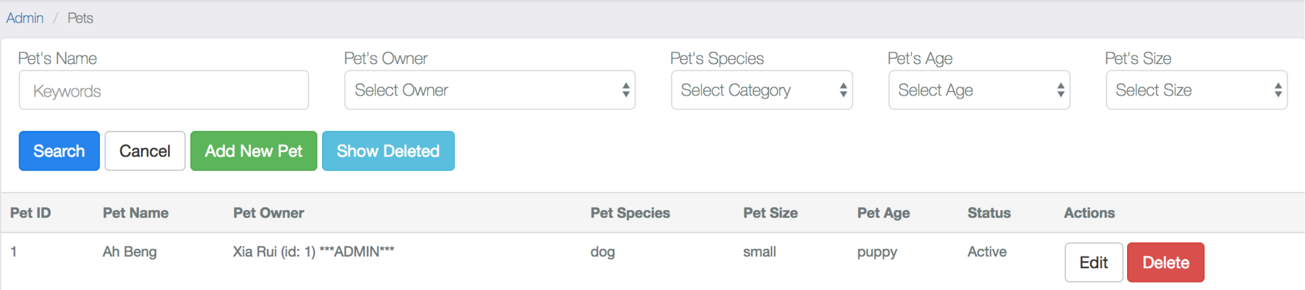
5.4.6 Admin usage for pet entries

- Generating view of all pet entries, with their species, size and age, as well as owner information.

```
SELECT p.pets_id, p.pet_name, pc.species, pc.size, pc.age,
       u.name, u.user_id, u.role, p.is_deleted
FROM pet p INNER JOIN petcategory pc ON p.pcat_id = pc.pcat_id
           INNER JOIN pet_user u ON p.owner_id = u.user_id
WHERE p.is_deleted = " . (isset($_GET['show_deleted']) ? "true" : "false") .
ORDER BY p.pets_id;
```

- Adding new pet into the system

```
INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES($pcat_id,$pet_owner,'$pet_name');
```



Pet's Name	Pet's Owner	Pet's Species	Pet's Age	Pet's Size
Keywords	Select Owner	Select Category	Select Age	Select Size

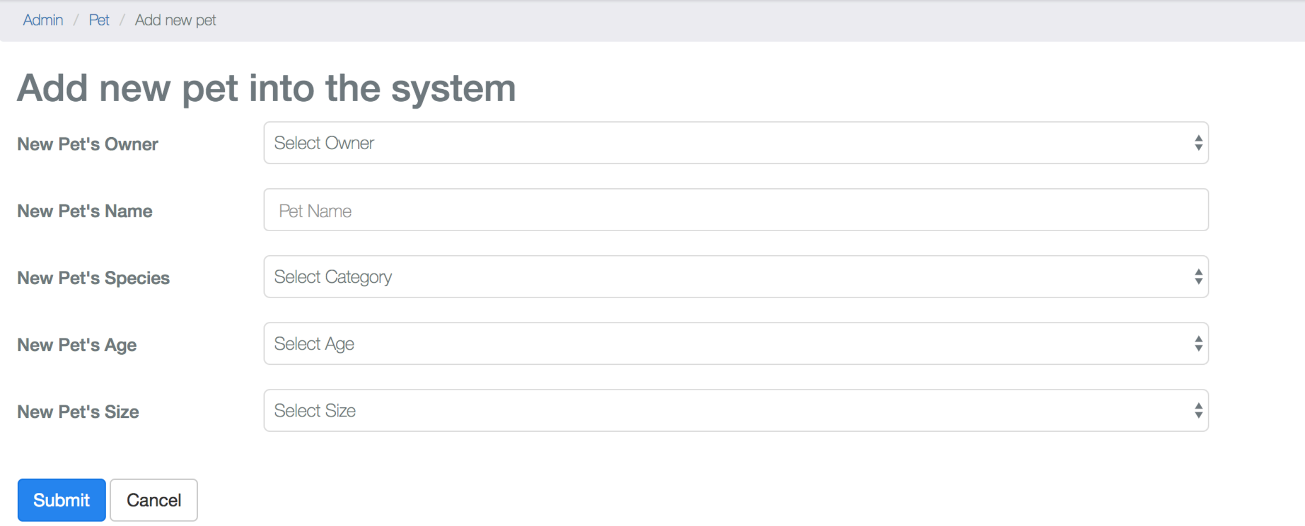
Pet ID	Pet Name	Pet Owner	Pet Species	Pet Size	Pet Age	Status	Actions
1	Ah Beng	Xia Rui (id: 1) ***ADMIN***	dog	small	puppy	Active	Edit Delete

- Before updating the request, we need to set requests on the pet to be failed if the pet category of the pet is updated.

```
UPDATE request SET status = 'failed' WHERE pets_id = $pet_id AND status = 'pending';
```

- Then we update the pet entry accordingly

```
UPDATE pet SET pcat_id = $pcat_id, pet_name = '$pet_name', owner_id = $owner_id
WHERE pets_id = $pet_id;
```



Add new pet into the system

New Pet's Owner	Select Owner
New Pet's Name	Pet Name
New Pet's Species	Select Category
New Pet's Age	Select Age
New Pet's Size	Select Size

- Deleting the pet entry would need to delete the corresponding requests that involves the pet which are pending.

```
UPDATE pet SET is_deleted=true WHERE pets_id=" . $p_id . ";”
UPDATE request SET status='failed' WHERE status='pending' AND pets_id=$p_id;
```

- However, when restoring the pet entries, these requests are not restored.

```
UPDATE pet SET is_deleted=false WHERE pets_id=" . $p_id . ";”
```

5.5 History Page for Owner (similar between taker history and owner history)

- Search for all requests

```
SELECT p.pet_name, t.name, r.post_time, r.care_begin, r.care_end, r.bids,
r.remarks, r.status FROM pet_user o, request r, pet p, pet_user t
WHERE r.owner_id = o.user_id AND r.pets_id = p.pets_id
      AND t.user_id = r.taker_id AND o.user_id = $user_id
```

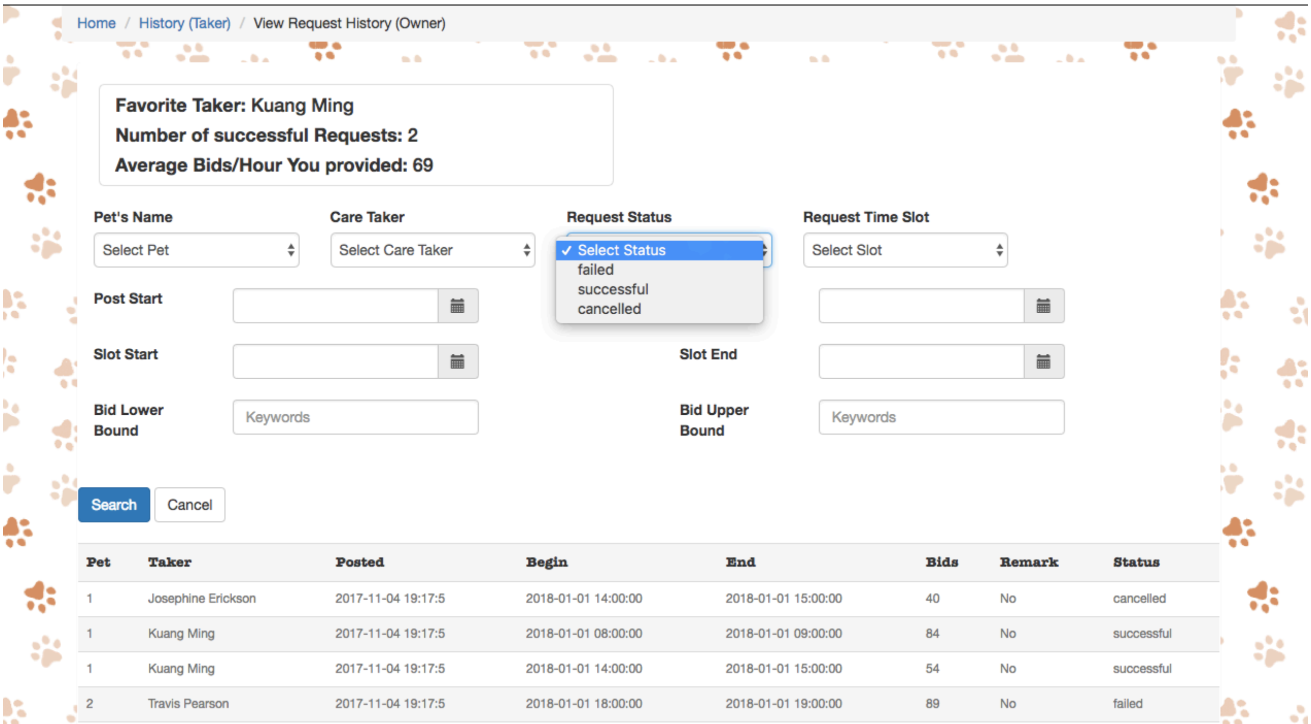
- Search by pets, taker, status, post-time, timeslot……

```
AND p.pets_id = $pet_id
AND r.taker_id =$taker_id
AND r.status = '" . $status . "'
AND r.post_time >= '" . $post_start . "'
AND r.post_time <= '" . $post_end . "'
AND r.care_begin >= '" . $slot_start . "'
AND r.care_end <= '" . $slot_end . "'
AND r.slot = '" . $req_slot . "'
AND r.bids >= $bid_low
AND r.bids <= $bid_upp
ORDER BY r.post_time;
```

AGGREGATE NESTED

- Show favorite takers, takers with the most number of successful requests, and its average bids

```
SELECT u.name, COUNT(*), (SUM(r.bids)/SUM(r.totaltime))*60 FROM request r, pet_user u
WHERE r.taker_id = $user_id AND r.status = 'successful' AND u.user_id = r.owner_id
GROUP BY r.owner_id, u.name
HAVING COUNT(*) >= ALL(SELECT COUNT(*) FROM request r1
                       WHERE r1.taker_id = $user_id AND r1.status = 'successful'
                       GROUP BY r1.owner_id)
ORDER BY (SUM(r.bids)/SUM(r.totaltime)) DESC;
```



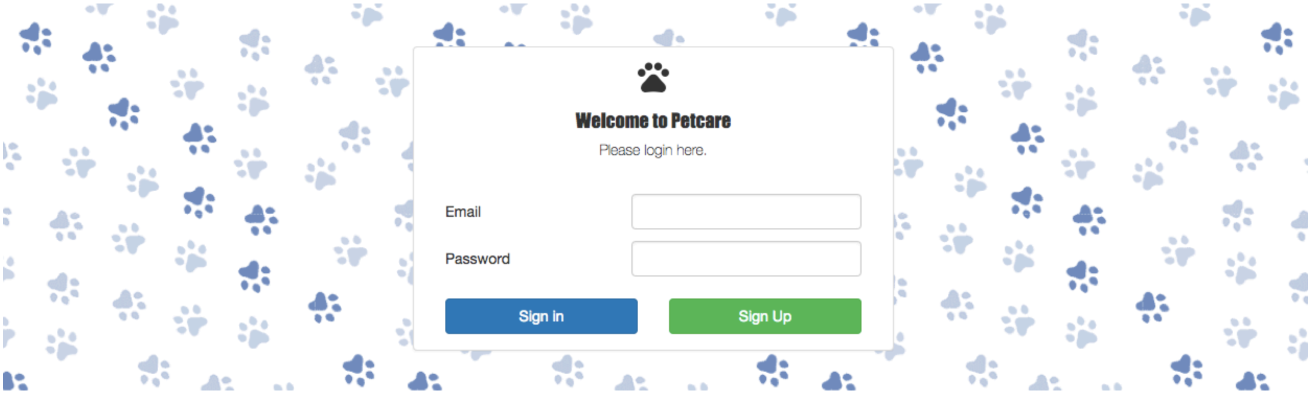
Favorite Taker: Kuang Ming
Number of successful Requests: 2
Average Bids/Hour You provided: 69

Pet	Taker	Posted	Begin	End	Bids	Remark	Status
1	Josephine Erickson	2017-11-04 19:17:5	2018-01-01 14:00:00	2018-01-01 15:00:00	40	No	cancelled
1	Kuang Ming	2017-11-04 19:17:5	2018-01-01 08:00:00	2018-01-01 09:00:00	84	No	successful
1	Kuang Ming	2017-11-04 19:17:5	2018-01-01 14:00:00	2018-01-01 15:00:00	54	No	successful
2	Travis Pearson	2017-11-04 19:17:5	2018-01-01 18:00:00	2018-01-01 19:00:00	89	No	failed

5.6 Login and Signup

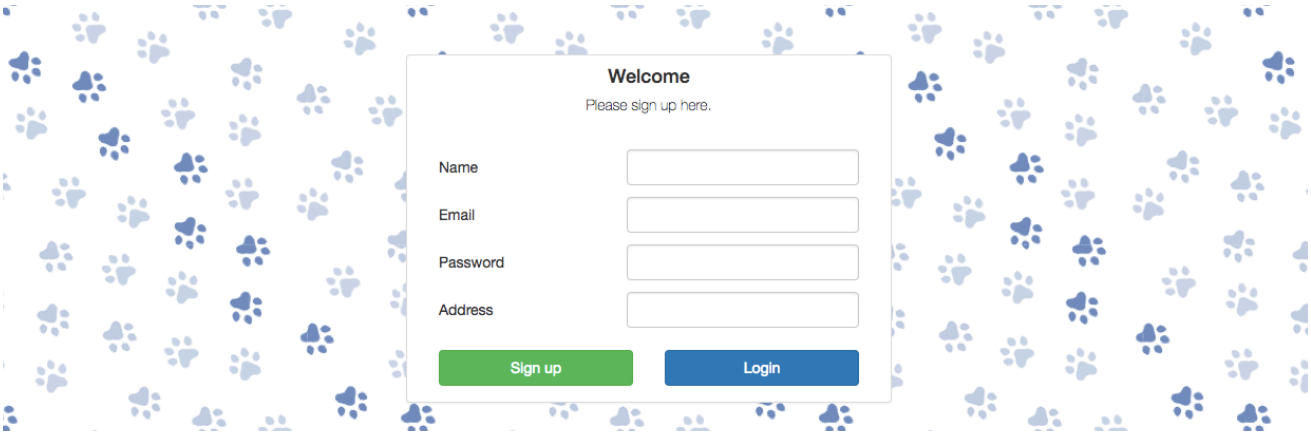
```
SELECT u.user_id, u.role FROM pet_user u
WHERE u.email = '".$email."' AND u.password = '". $_GET['password']."';

INSERT INTO pet_user (name, email, password, address)
VALUES ('".$name."', '".$email."', '".$password."','".$address."' );
```



Welcome to Petcare

Please login here.



Welcome

Please sign up here.

6. Sample Data

6.1. Pet Users (all 24 entries)

```sql
INSERT INTO pet_user(name, password, email, address, role) VALUES ('Xia
Rui',12345,'e0012672@u.nus.edu','30 Ang Mo Kio Ave 8', 'admin');
INSERT INTO pet_user(name, password, email, address, role) VALUES ('Chen
Penghao',12345,'e0004801@u.nus.edu','33 Lorong 2 Toa Payoh', 'admin');
INSERT INTO pet_user(name, password, email, address, role) VALUES ('Xie
Peiyi',12345,'peiyi@u.nus.edu','55 Hougang Ave 10', 'admin');
INSERT INTO pet_user(name, password, email, address, role) VALUES ('Kuang
Ming',12345,'km@msn.com','', 'admin');

INSERT INTO pet_user(name, password, email, address) VALUES ('Patti
Dennis',12345,'empathy@msn.com','157 Foxrun Street Newnan, GA 30263');
INSERT INTO pet_user(name, password, email, address) VALUES ('Carmen
Grant',23456,'presoff@hotmail.com','9 South Surrey Street Rockford, MI 49341');
INSERT INTO pet_user(name, password, email, address) VALUES ('Abel
Lucas',34567,'keijser@optonline.net','930 Storm Court Washington, PA 15301');
INSERT INTO pet_user(name, password, email, address) VALUES ('Marguerite
Jennings',45678,'curly@gmail.com','508 E. Longfellow Rd. Revere, MA 02151');
INSERT INTO pet_user(name, password, email, address) VALUES ('Samuel
Lawrence',56789,'squirrel@aol.com','8807 Aurora Road Ogden, UT 84404');
INSERT INTO pet_user(name, password, email, address) VALUES ('Lydia
Turner',67900,'cantu@verizon.net','29 Paradise Court Moorhead, MN 56560');
INSERT INTO pet_user(name, password, email, address) VALUES ('Eloise
Cooper',79011,'pajas@msn.com','9267 1st St. Wenatchee, WA 98801');
INSERT INTO pet_user(name, password, email, address) VALUES ('Maxine
Ramos',90122,'vertigo@aol.com','671 Liberty Dr. Ankeny, IA 50023');
INSERT INTO pet_user(name, password, email, address) VALUES ('Kyle
Colon',12334,'aprakash@me.com','49 Walt Whitman Street Apopka, FL 32703');
INSERT INTO pet_user(name, password, email, address) VALUES ('Laverne
Valdez',12344,'lishoy@verizon.net','12 Bald Hill Street Norfolk, VA 23503');
INSERT INTO pet_user(name, password, email, address) VALUES ('David
Reynolds',23455,'marnanel@hotmail.com','224 Second Drive Cocoa, FL 32927');
INSERT INTO pet_user(name, password, email, address) VALUES ('Clyde
Mack',34566,'smartfart@verizon.net','870 Addison Court Dacula, GA 30019');
INSERT INTO pet_user(name, password, email, address) VALUES ('Cameron
Huff',45677,'petersko@yahoo.ca','7834 Ann Street Quincy, MA 02169');
INSERT INTO pet_user(name, password, email, address) VALUES ('Ebony
Mendez',56788,'avalon@att.net','8789 Hart St. Ballston Spa, NY 12020');
INSERT INTO pet_user(name, password, email, address) VALUES ('Joe
Munoz',67899,'ournews@live.com','94 Meadowbrook St.Apt 36 Florence, SC 29501');
INSERT INTO pet_user(name, password, email, address) VALUES ('Travis
Pearson',79010,'chaffar@mac.com','436 E. Second Avenue Missoula, MT 59801');
INSERT INTO pet_user(name, password, email, address) VALUES ('Robin
Goodman',90121,'mdielmann@hotmail.com','11 Brewer Road Chardon, OH 44024');
INSERT INTO pet_user(name, password, email, address) VALUES ('Marcus
Gilbert',81232,'weazelman@yahoo.com','12 Summerhouse St. Hoboken, NJ 07030');
INSERT INTO pet_user(name, password, email, address) VALUES ('Doug
Neal',12343,'msloan@me.com','5 East Proctor Street Missoula, MT 59801');
INSERT INTO pet_user(name, password, email, address) VALUES ('Josephine
Erickson',23454,'goresky@msn.com','7943 East Lakeshore Street Rockford, MI 49341');
```

6.2. Pet Categories (all 24 entries)

```sql
INSERT INTO petcategory (age, size, species) VALUES ('puppy','small','cat');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','small','dog');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','small','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','medium','cat');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','medium','dog');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','medium','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','large','cat');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','large','dog');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','large','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','giant','cat');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','giant','dog');
INSERT INTO petcategory (age, size, species) VALUES ('puppy','giant','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('adult','small','cat');
INSERT INTO petcategory (age, size, species) VALUES ('adult','small','dog');
INSERT INTO petcategory (age, size, species) VALUES ('adult','small','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('adult','medium','cat');
INSERT INTO petcategory (age, size, species) VALUES ('adult','medium','dog');
INSERT INTO petcategory (age, size, species) VALUES ('adult','medium','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('adult','large','cat');
INSERT INTO petcategory (age, size, species) VALUES ('adult','large','dog');
INSERT INTO petcategory (age, size, species) VALUES ('adult','large','rabbit');
INSERT INTO petcategory (age, size, species) VALUES ('adult','giant','cat');
INSERT INTO petcategory (age, size, species) VALUES ('adult','giant','dog');
INSERT INTO petcategory (age, size, species) VALUES ('adult','giant','rabbit');
```

6.3. Pets (first 20 data)

```
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (21,1,'pet1');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (17,1,'pet2');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (6,2,'pet3');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (7,2,'pet4');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (24,3,'pet5');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (12,3,'pet6');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (4,4,'pet7');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (13,4,'pet8');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (15,5,'pet9');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (13,5,'pet10');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (13,6,'pet11');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (22,6,'pet12');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (9,7,'pet13');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (22,7,'pet14');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (12,8,'pet15');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (19,8,'pet16');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (18,9,'pet17');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (14,9,'pet18');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (10,10,'pet19');
 INSERT INTO pet(pcat_id, owner_id, pet_name) VALUES (19,10,'pet20');
```

6.4. Availabilities (first 20 data)

```
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
01:00:00','2018-01-01 04:00:00',2,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
12:00:00','2018-01-01 17:00:00',2,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
19:00:00','2018-01-02 03:00:00',2,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
10:00:00','2018-01-02 14:00:00',2,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
18:00:00','2018-01-02 19:00:00',2,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
09:00:00','2018-01-01 16:00:00',4,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
18:00:00','2018-01-01 20:00:00',4,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
23:00:00','2018-01-02 01:00:00',4,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
09:00:00','2018-01-02 10:00:00',4,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
15:00:00','2018-01-02 17:00:00',4,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
05:00:00','2018-01-01 09:00:00',6,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
10:00:00','2018-01-01 13:00:00',6,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
03:00:00','2018-01-02 07:00:00',6,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
10:00:00','2018-01-02 11:00:00',6,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
12:00:00','2018-01-02 14:00:00',6,1);
```

```
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
02:00:00','2018-01-01 11:00:00',16,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
13:00:00','2018-01-01 14:00:00',16,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-01
19:00:00','2018-01-02 04:00:00',16,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
14:00:00','2018-01-02 15:00:00',16,1);
INSERT INTO availability(start_time, end_time, pcat_id, taker_id) VALUES ('2018-01-02
16:00:00','2018-01-02 19:00:00',16,1);
```

6.5. Requests (first 20 data)

```
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,9,'2018-01-02 08:00:00','2018-01-02
09:00:00','No',48,1,'cancelled');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,3,'2018-01-01 01:00:00','2018-01-01 02:00:00','No',10,1,'pending');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,15,'2018-01-01 01:00:00','2018-01-01 02:00:00','No',53,1,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,12,'2018-01-02 15:00:00','2018-01-02
16:00:00','No',24,1,'successful');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,9,'2018-01-02 02:00:00','2018-01-02
03:00:00','No',77,1,'successful');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,9,'2018-01-01 22:00:00','2018-01-01 23:00:00','No',25,1,'pending');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,9,'2018-01-02 07:00:00','2018-01-02 08:00:00','No',1,1,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,15,'2018-01-02 10:00:00','2018-01-02 11:00:00','No',32,1,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,20,'2018-01-02 00:00:00','2018-01-02 01:00:00','No',59,1,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,12,'2018-01-02 01:00:00','2018-01-02
02:00:00','No',36,1,'successful');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,17,'2018-01-02 02:00:00','2018-01-02 03:00:00','No',99,2,'pending');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,17,'2018-01-01 07:00:00','2018-01-01
08:00:00','No',21,2,'cancelled');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,19,'2018-01-01 19:00:00','2018-01-01 20:00:00','No',49,2,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,21,'2018-01-01 07:00:00','2018-01-01 08:00:00','No',81,2,'pending');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,19,'2018-01-01 17:00:00','2018-01-01
18:00:00','No',30,2,'cancelled');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,15,'2018-01-01 11:00:00','2018-01-01
12:00:00','No',48,2,'successful');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,19,'2018-01-02 22:00:00','2018-01-02 23:00:00','No',19,2,'failed');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,19,'2018-01-02 19:00:00','2018-01-02
20:00:00','No',15,2,'successful');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,21,'2018-01-02 21:00:00','2018-01-02 22:00:00','No',47,2,'pending');
INSERT INTO request(owner_id, taker_id, care_begin, care_end, remarks, bids, pets_id,
status) VALUES (1,15,'2018-01-02 03:00:00','2018-01-02 04:00:00','No',60,2,'failed');
```