

CS2102 Database Systems
AY 2020/21 Semester 1



Pet Care System Project
Final Report

Group CS2102_2021_S1_41

Chan Wei Qiang Jason	A0184165B
Chew Zhao En	A0190128J
Lim Jia Ying	A0184190E
Ko Gi Hun	A0185313J
Putra Mohammad Danish Bin Mohd Rafee	A0164802J

1. Project Responsibilities

Project distribution and responsibilities are as follows;

Ko Gi Hun	<ul style="list-style-type: none"> - SQL Query Design - Web App development support - Report Writeup
Chew Zhao En	<ul style="list-style-type: none"> - Report Writeup - 1 Trigger - SQL Query Design
Lim Jia Ying	<ul style="list-style-type: none"> - Front-End Development & Design - Back-End Development & Design - Web App Deployment - Report Writeup
Putra Mohammad Danish Bin Mohd Rafee	<ul style="list-style-type: none"> - Application data constraints - 2 Triggers
Chan Wei Qiang Jason	<ul style="list-style-type: none"> - ER Diagram/ Constraints - Table Creation - Report Writeup - Web App development support

2. Data requirements and functionalities

2.1. Notable Application Data Constraints

Table Name	Attribute	Data Type	Constraints	Keys	Possible options
users	username	varchar(50)	not null	primary key	
	password_hash	varchar(64)	not null		*generated in web app
	salt	varchar(16)	not null		*generated in web app
pcs_administrators	username	varchar(50)	not null	primary key	
				foreign key references <i>users</i> on delete cascade	
pet_owners	username	varchar(50)	not null	primary key	
				foreign key references <i>users</i> on delete cascade	
care_takers	username	varchar(50)	not null	primary key	

				foreign key references <i>users</i> on delete cascade	
	employee_type	varchar(50)	not null		'part-time'
					'full-time'
	area	varchar(50)			'north'
					'south'
					'central'
					'east'
					'west'
base_prices	pet_type	varchar(50)	not null	primary key	
	price	numeric(4, 2)	not null		
			check (price>0)		
pets	username	varchar(50)	not null	foreign key references <i>users</i> on delete cascade	
	pet_type	varchar(50)	not null	foreign key references <i>base_prices</i> on delete cascade	
	other keys			primary key (username, pet_name)	
bids	pet_owner	varchar(50)	not null	foreign key references <i>pet_owners</i> on delete cascade	
	care_taker	varchar(50)	not null	foreign key references <i>care_takers</i> on delete cascade	
	transfer_mode	varchar(50)	not null		'PCS Meet-up'
					'Deliver'
	daily_price	numeric(4,2)	not null		
			check (daily_price>0)		
	payment_type	varchar			'credit card'
					'NETS'

					'cash'
	other constraints and keys		check (end_date >= start_date)	primary key (pet_owner, care_taker, pet_name, start_date, end_date)	
			check (((rating between 0 and 5) or rating IS null))	foreign key (pet_owner, pet_name) references <i>pets</i> on delete cascade	
prices	care_taker	varchar(50)	not null	foreign key references <i>care_takers</i> on delete cascade	
	pet_type	varchar(50)	not null	foreign key references <i>base_prices</i> on delete cascade	
	price	numeric(4,2)	not null		
			check (price>0)		
	other keys			primary key (care_taker, pet_type)	
availabilities	care_taker	varchar(20)	not null	foreign key references <i>care_takers</i> on delete cascade	
	other constraints		check (end_date >= start_date)		
leaves	care_taker	varchar(20)	not null	foreign key references <i>care_takers</i> on delete cascade	
	other constraints		check (end_date >= start_date)		*leaves start_date and end_date must fall between start_date and end_date in <i>availabilities</i>

Note 1: boolean types are implicitly defined to have true or false as the only options

Note 2: date types are in the form 'YYYY-MM-DD'

Note 3: All other attributes not described in the table above have either no constraints or only 'not null' as the only constraint. They are further described in the SQL implementation under section 3

2.2. Application Functionalities

2.2.1 Pet Owner

Pet owners will be able to add pets and delete pets from the petOwner dashboard. To add a pet the pet owner can simply click on the + button and input the pet_name and pet_type in the form accordingly. To delete a pet the pet owner can simply click on the trash can on the row that the pet is on.

petOwner Dashboard:

Welcome Admin

Your Pets (+)

Name	Type	
Catto	cat	🗑️
Doggo	dog	🗑️

Quick Access

[Find a Caretaker](#)

Your Bids

Pet	Care Taker	Start	End	Confirmed	Price	Payment	Review
Catto	Philip Tang	19-11-2020	20-11-2020	✗	18.00	<button>Pay</button>	<button>Review</button>
Catto	Russel Tang	09-11-2020	10-11-2020	✓	18.00	<button>Pay</button>	<button>Review</button>
Catto	Austin Tan	06-11-2020	06-11-2020	✓	22.00	<button>Paid</button>	<button>Review</button>

Add Pet:

Add New Pet ✕

Pet Name

Pet Type

Special Requirements

Add Pet Cancel

Pet owners will also be able to do a search on the caretakers using our search function using pet_type, min price and max price.

Search Form:

Apply for new caretaker

Firstly, please search by pet type

Pet Type

E.g. dog, cat, etc

Min Price

0

Max Price

100

Search

Search Results:

Apply for new caretaker
Please select a care taker

Care Taker	Price (\$)	Apply
Austin Tan	22.00	Bid
Eugene Tan	24.00	Bid
Full Timer	18.00	Bid
Part Timer	15.00	Bid
Philip Tang	18.00	Bid
Russel Tang	18.00	Bid

[Back](#)

Pet owners will then be able to bid from the search results which will bring them to the bids form.

Bids Form:

Apply for new caretaker
Enter your bid info
Austin Tan's rate: \$22.00

Your Pet Name

Pet Type

Your Bid

Pet Transfer Method

Start Date

End Date

[Submit Bid](#) [Back](#)

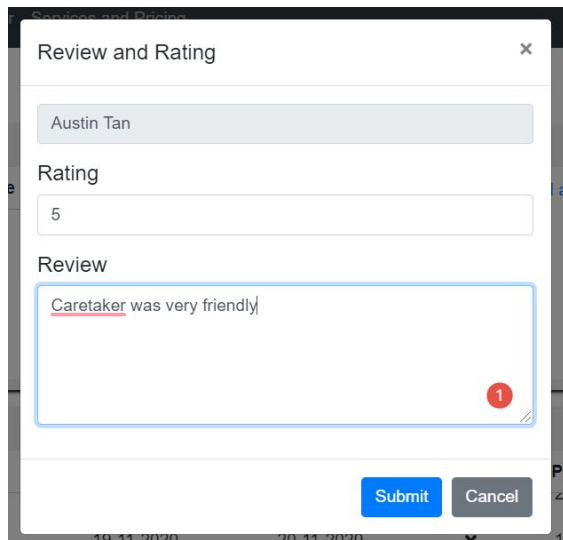
After successfully bidding for the caretaker, the pet owner will now wait for the caretaker to accept the bid. After the caretaker has accepted the pet owner's bid, the pet owner can now proceed to pay using the pay button. The pay button will change to paid after payment.

petOwner Dashboard (pay button and paid indication):

Your Bids						
Pet	Care Taker	Start	End	Confirmed	Price	Payment Review
Doggo	Full Timer	06-11-2020	06-11-2020	✓	20.00	Paid Done
Catto	Philip Tang	19-11-2020	20-11-2020	✗	18.00	Pay Review
Catto	Russel Tang	09-11-2020	10-11-2020	✓	18.00	Pay Review
Catto	Austin Tan	06-11-2020	06-11-2020	✓	22.00	Paid Review

After the service is complete, the pet owner will now be able to give rating and review to the caretaker on the specific transaction. There will be a trigger to update the avg_rating of the caretaker once the rating is submitted.

petOwner Dashboard (rating/review form):



Review and Rating

Austin Tan

Rating: 5

Review: Caretaker was very friendly

Submit Cancel

2.2.2.1 Caretaker (Part-time)

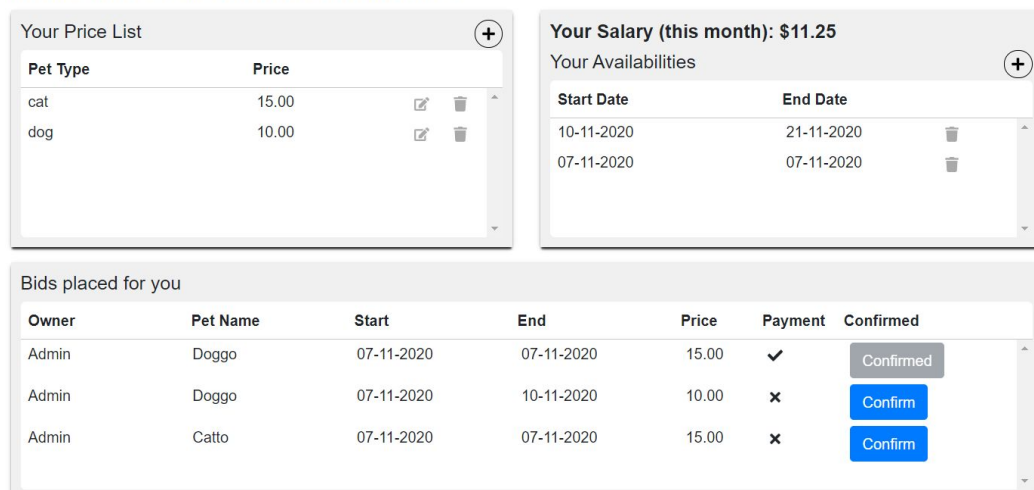
Part-time caretakers can add, edit or delete their individual prices for each pet_type.

To add a price, the caretaker can simply click on the + button and input the pet_type and the corresponding price in the form accordingly. The pet_type chosen must be in the base price list declared by the admin and the value must not be lower than the base price or a corresponding error will be displayed.

To edit or delete a price the caretaker can simply click on the edit icon or the trash can on the row that the pet_type is on. Part-time caretakers have to manually accept bids from pet owners.

Caretaker Dashboard (Part-time) + manual bid accept:

Part-time Caretaker Dashboard



Your Price List

Pet Type	Price		
cat	15.00		
dog	10.00		

Your Salary (this month): \$11.25

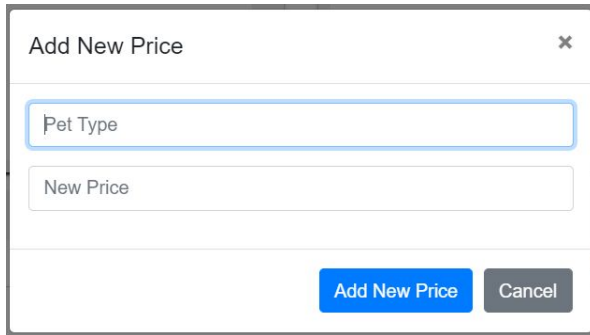
Your Availabilities

Start Date	End Date	
10-11-2020	21-11-2020	
07-11-2020	07-11-2020	

Bids placed for you

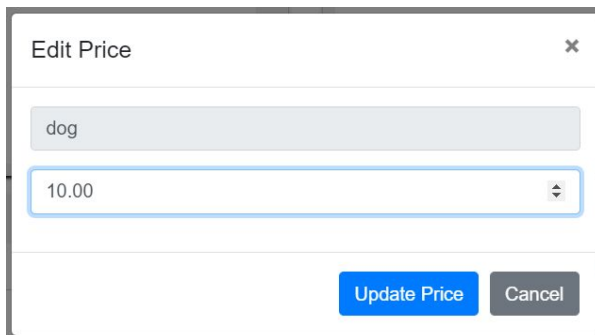
Owner	Pet Name	Start	End	Price	Payment	Confirmed
Admin	Doggo	07-11-2020	07-11-2020	15.00	✓	Confirmed
Admin	Doggo	07-11-2020	10-11-2020	10.00	✗	Confirm
Admin	Calto	07-11-2020	07-11-2020	15.00	✗	Confirm

Add Price:



A dialog box titled "Add New Price" with a close button (X) in the top right corner. It contains two input fields: "Pet Type" and "New Price". At the bottom, there are two buttons: "Add New Price" (blue) and "Cancel" (grey).

Edit Price:



A dialog box titled "Edit Price" with a close button (X) in the top right corner. It contains two input fields: the first is a text field with "dog" entered, and the second is a numeric field with "10.00" entered. At the bottom, there are two buttons: "Update Price" (blue) and "Cancel" (grey).

Part-time caretakers will have to declare their availability. Part-time caretakers will not be able to accept jobs on dates that they are not declared available. They will also be able to view their salary and the pets they have taken care of for the month. (Trigger 1: salary for caretakers)

Caretaker Dashboard (Part-time) + salary + pets taken care of:

The Part-time caretaker will then await payment and the date of transaction. After the transaction is completed, the caretaker may get a rating and review from the pet owner. (Trigger 2: avg_rating)

2.2.2.2 Caretaker (Full-time)

Full-time caretakers can add or delete which pet_type they are able to take care of.

To add a pet_type, the caretaker can simply click on the + button and input the pet_type. The corresponding price will be added by the system through a trigger. The corresponding price for the full-time caretaker will be the base price if his avg_rating is < 4 else the corresponding price will $1.2 \times \text{base price}$ if his avg_rating is ≥ 4 . (Trigger 4: price computation for full-time caretakers when they add a pet type)

The full-time caretaker will not be able to edit his prices as it is determined by the system.

To delete a price the caretaker can simply click on the trash can on the row that the pet_type is on.

Caretaker Dashboard (Full-time):

Full-time Caretaker Dashboard

Your Price List

Pet Type	Price
cat	18.00
dog	12.00

Your Salary (this month): \$3000

Your Leaves

Start Date	End Date
18-11-2020	19-11-2020
15-11-2020	15-11-2020
18-11-2020	09-09-2021

Bids placed for you

Owner	Pet Name	Start	End	Price	Payment
Admin	Doggo	07-11-2020	11-11-2020	20.00	✓
Admin	Doggo	07-11-2020	07-11-2020	20.00	✓
Admin	Doggo	07-11-2020	09-11-2020	20.00	✗
Admin	Doggo	07-11-2020	08-11-2020	20.00	✓
Admin	Doggo	06-11-2020	06-11-2020	20.00	✓

Add Price:

Add New Price

Add New Price

Cancel

Full-time caretakers have to declare leaves and are available on the dates where they are not on leave. Full-time caretakers automatically accept bids from pet owners if he is available on those dates and his price is met. The automatic acceptance system checks that on every date in the bid, the full-time caretaker must have less than 5 pets under his care so that he does not go past the limit. (Trigger 3: auto accepting of jobs for full-time caretakers)

Add leaves:

Apply New Leave

Start Date

End Date

Submit

Cancel

Full-time caretakers will also be able to view their salary and the pets they have taken care of for the month. (Trigger 1: salary for caretakers)

Caretaker Dashboard (Full-time) + salary + pets taken care of:

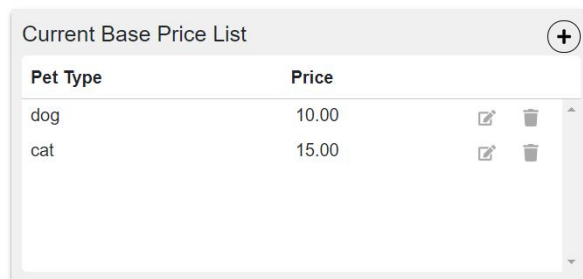
The Full-time caretaker will now await payment and the date of transaction. After the transaction, the caretaker may get a rating and review from the pet owner. (Trigger 2: avg_rating)





2.2.3 PCS Administrator

PCS Administrators can add, edit or delete the base price list. When the base price for a pet_type increases, part-time caretakers will automatically change their price to the new base price if their price becomes below the base price. Updating the base prices will also automatically update the base prices for the full-time caretakers as their pay is dependent on the base price. (Trigger 5: changing of base prices will update the prices 1. for part-time caretakers if their price becomes below the base price 2. For full-time caretakers a new price will be computed for them)

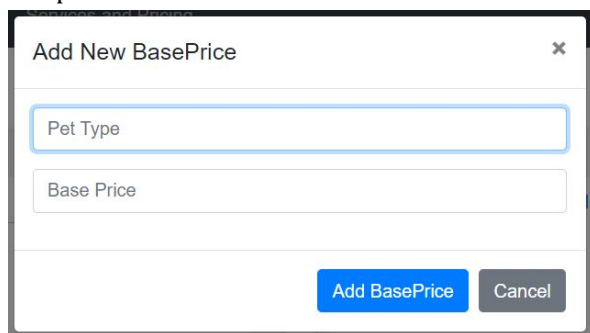
Admin Dashboard:

Admin Dashboard



Pet Type	Price	
dog	10.00	 
cat	15.00	 

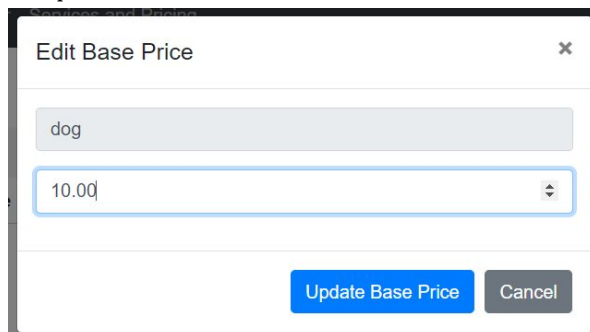
Add price:



Add New BasePrice

Add BasePrice Cancel

Edit price:



Edit Base Price

dog

Update Base Price Cancel

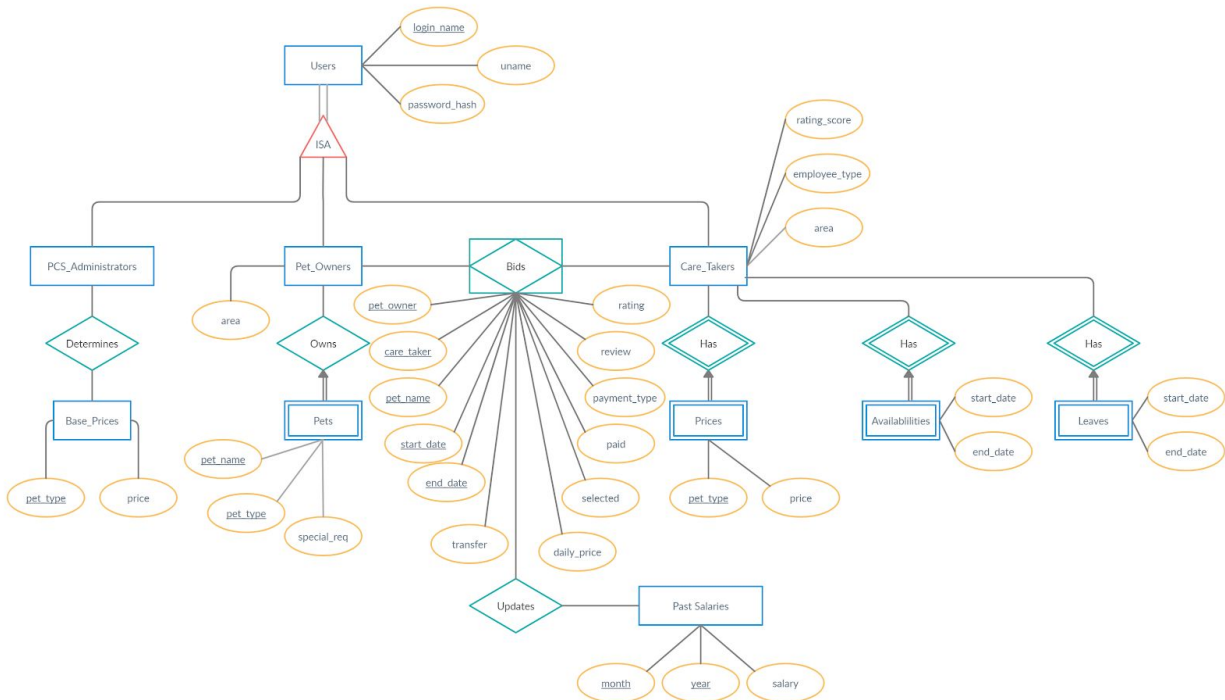
PCS Administrators also have admin access to quickly and conveniently control the user type for each user.

Admin Control Page:

All Users

Username	Display Name	Account Type
Admin	Admin	<input checked="" type="checkbox"/> Pet Owner <input type="checkbox"/> Care Taker <input checked="" type="checkbox"/> Admin
Part Timer	Part Timer	<input checked="" type="checkbox"/> Pet Owner <input checked="" type="checkbox"/> Care Taker <input type="checkbox"/> Admin
Alan Lim	Alan Lim	<input checked="" type="checkbox"/> Pet Owner <input type="checkbox"/> Care Taker <input type="checkbox"/> Admin
Philip Tang	Philip Tang	<input checked="" type="checkbox"/> Pet Owner <input checked="" type="checkbox"/> Care Taker <input type="checkbox"/> Admin
Bryan Lim	Bryan Lim	<input checked="" type="checkbox"/> Pet Owner <input type="checkbox"/> Care Taker <input type="checkbox"/> Admin

3. Database Schema (ER Model, Create Table Queries)



```
create table users
(
    username      varchar(50) not null
        constraint users_pkey
            primary key,
    display_name  varchar(50) not null,
    password_hash varchar(64) not null,
    salt          varchar(16)
);

create table pcs_administrators
(
    username varchar(50) not null
        constraint
            pcs_administrators_pkey
            primary key
        constraint
            pcs_administrators_username_fkey
            references users
            on delete cascade
);
```

```
create table bids
(
    pet_owner  varchar(50) not null
        constraint bids_pet_owner_fkey
            references pet_owners
            on delete cascade,
    care_taker varchar(50) not null
        constraint bids_care_taker_fkey
            references care_takers
            on delete cascade,
    pet_name   varchar(50) not
null,
    transfer_mode varchar(50) not
null,
    start_date  date not
null,
    end_date    date not
null,
    daily_price numeric(4, 2) not null
        constraint
            bids_daily_price_check
            check (daily_price > 0)
);
```

```

create table pet_owners
(
    username varchar(50) not null
        constraint pet_owners_pkey
            primary key
        constraint
pet_owners_username_fkey
            references users
            on delete cascade,
    area      varchar(50)
);

create table care_takers
(
    username      varchar(50) not null
        constraint care_takers_pkey
            primary key
        constraint
care_takers_username_fkey
            references users
            on delete cascade,
    employee_type varchar(50) not null,
    area          varchar(50),
    avg_rating    double precision
);

create table base_prices
(
    pet_type varchar(50) not null
        constraint base_prices_pkey
            primary key,
    price    numeric(4, 2) not null
        constraint
base_prices_price_check
            check (price > (0)::numeric)
);

create table pets
(
    username      varchar(50) not null
        constraint pets_username_fkey
            references pet_owners
            on delete cascade,
    pet_name      varchar(50) not null,
    pet_type      varchar(50) not null

```

```

        check (daily_price >
(0)::numeric),
    selected      boolean,
    paid          boolean,
    payment_type  varchar,
    rating        integer,
    review        varchar(200),
        constraint bids_pkey
            primary key (pet_owner,
care_taker, pet_name, start_date,
end_date),
        constraint
bids_pet_owner_pet_name_fkey
            foreign key (pet_owner,
pet_name) references pets
            on delete cascade,
        constraint bids_check
            check (end_date >= start_date),
        constraint bids_check1
            check (((rating >= 0) AND
(rating <= 5)) OR rating IS NULL)),
);

create table prices
(
    care_taker varchar(50) not null
        constraint
prices_care_taker_fkey
            references care_takers
            on delete cascade,
    pet_type   varchar(50) not null
        constraint prices_pet_type_fkey
            references base_prices
            on delete cascade,
    price      numeric(4, 2) not null
        constraint prices_price_check
            check (price >
(0)::numeric),
        constraint prices_pkey
            primary key (care_taker,
pet_type)
);

create table leaves

```

<pre> constraint pets_pet_type_fkey references base_prices on delete cascade, special_req varchar(50), constraint pets_pkey primary key (username, pet_name)); create table availabilities (care_taker varchar(50) not null constraint availablilities_care_taker_fkey references care_takers on delete cascade, start_date date not null, end_date date not null, constraint availablilities_check check (end_date >= start_date)); </pre>	<pre> (care_taker varchar(50) not null constraint leaves_care_taker_fkey references care_takers on delete cascade, start_date date not null, end_date date not null, constraint leaves_check check (end_date >= start_date)); </pre>
---	---

4. Constraints

- Users have to belong to either Admin, Pet Owners or Caretakers (Covering constraint). Users can belong to multiple groups (Overlapping Constraint).
- Price_List of Care_Takers will record down the available pet types the Caretakers are capable of handling.
- Part-time Caretakers can add pet_type and price into his prices table only if the pet_type exists in the base_prices table and the price entered is higher or equal to the price stated in the base_prices table
- Full-time Caretakers can only add pet_type into price_list as the prices will be auto generated by the system.
- Part-time Caretakers can specify their own availability for the current year + the next year.
- Full-time Caretakers have to specify their leaves for the current year + the next year and are available when they are not on leave.
- Caretakers avg_rating will be between 0 and 5
- Part-time Caretakers can take care of up to 2 pets in a day if his avg_rating <4. If avg_rating >= 4, he can take care of up to 5 pets in a day.
- Full-time Caretakers can take care of up to 5 pets in a day

- Part-time caretakers can only accept bids if the dates of the bids are within the dates in availabilities.
- Pet Owners can only pay for bids when they have been selected
- Pet Owners can only make review and rating for bids when they have been completed
- Pet Owners will be able to search Caretakers based on pet_type and min price and max price
- Existence of Pets(weak entity) depends on the existence of Pet_Owners(owning entity)
- Existence and identity of Prices(weak entity) depends on the existence of Care_Takers(owning entity)
- Existence and identity of Availability(weak entity) depends on the existence of Care_Takers(owning entity)
- Existence and identity of Leave(weak entity) depends on the existence of Care_Takers(owning entity)
- Care_Takers cannot bid or rate themselves

5. 3NF/BNF Analysis

5.1. Representing SQL schema into Functional Dependencies and R

R = (u1, u2, u3, u4, pa1, po1, ct1, bp1, p1, p2, p3, b1, b2, b3, b4, b5, b6, b7, b8, b9, pr1, av1, av2, lv1, lv2)

F = {u1->u2u3u4, pa1->u2u3u4, po1->u2u3u4, ct1->u2u3u4, p2->bp1, u1p1->p2p3, po1ct1p1b2b3->b1b4b5b6b7b8b9, ct1p2->pr1}

Relation Fragment: SQL Table	Functional Dependency
<R1(u1, u2, u3, u4): Users> <u>username</u> : u1 display_name: u2 password_hash: u3 salt: u4	u1->u2u3u4 (superkey)
<R2(pa1): PCA_Administrators> <u>username</u> : pa1	pa1->u2u3u4 (superkey)
<R3(po1): Pet_Owners> <u>username</u> : po1	po1->u2u3u4 (superkey)
<R4(ct1): Care_Takers> <u>username</u> : ct1	ct1->u2u3u4 (superkey)
<R5(p2, bp1): Base_Prices>	p2->bp1 (superkey)

<u>pet_type</u> : p2 price : bp1	
<R6(u1, p1, p2, p3): Pets> <u>username</u> : u1 <u>pet_name</u> : p1 pet_type: p2 special_req: p3	u1p1->p2p3 (superkey)
<R7(po1, ct1, p1, b1, b2, b3, b4, b5, b6, b7, b8, b9): Bids> <u>pet_owner</u> : po1 <u>care taker</u> : ct1 <u>pet_name</u> : p1 transfer_mode: b1 <u>start_date</u> : b2 <u>end_date</u> : b3 daily_price: b4 selected: b5 paid: b6 payment_type: b7 rating: b8 review: b9	po1ct1p1b2b3->b1b4b5b6b7b8b9 (superkey)
<R8(ct1, p2, pr1): Prices> <u>care taker</u> : ct1 <u>pet_type</u> : p2 price: pr1	ct1p2->pr1 (superkey)
<R9(ct1, av1, av2): Availabilities> care_taker: ct1 start_date: av1 end_date: av2	N/A
<R10(ct1, lv1, lv2): Leaves> care_taker: ct1 start_date: lv1 end_date: lv2	N/A

*<R9 Availabilities> and <R10 Leaves> do not have a key thus they are not captured in the functional dependency for our analysis.
The rest of the tables are in BCNF as the LHS of their functional dependencies all contain a super key.

6. 4 Non-trivial/interesting triggers

- 1) calculates average of ratings of caretakers in table bids of and updates the avg_rating in table care_takers (refer to the SQL snippet below)

```
CREATE OR REPLACE FUNCTION calculate_avg() RETURNS TRIGGER AS
$$ BEGIN
UPDATE care_takers SET avg_rating = (SELECT AVG(rating) FROM bids WHERE
bids.care_taker = care_takers.username); RETURN NEW; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER calculate_avg_trigger AFTER INSERT OR UPDATE
ON bids FOR EACH ROW
EXECUTE PROCEDURE calculate_avg();
```

- 2) When bid by any Pet Owner, a full-time Caretaker will always accept the job immediately if possible

```
CREATE OR REPLACE FUNCTION accept_job() RETURNS TRIGGER AS
$$ BEGIN
UPDATE bids SET selected = true
WHERE care_taker =
(SELECT care_takers.username
FROM care_takers
WHERE care_takers.username = new.care_taker
AND care_takers.employee_type = 'full-time')
AND EXISTS
(SELECT *
FROM availabilities
WHERE availabilities.care_taker= new.care_taker
AND new.start_date>= availabilities.start_date
AND new.end_date<= availabilities.end_date)
AND new.daily_price>=
(SELECT price
FROM prices
WHERE new.care_taker = prices.care_taker
AND prices.pet_type = (SELECT pets.pet_type
FROM pets
WHERE pets.username = new.pet_owner
AND pets.pet_name = new.pet_name));

RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER po_bid BEFORE INSERT
ON bids FOR EACH ROW
EXECUTE PROCEDURE accept_job();
```

- 3) If a full-time caretaker applies for leave, their availability will automatically be set to not available for the range of the dates of the leave.

```
CREATE OR REPLACE FUNCTION take_leave() RETURNS TRIGGER AS
$$ BEGIN
INSERT INTO availabilities
SELECT new.care_taker,availabilities.start_date,new.start_date - INTERVAL '1 day'
FROM availabilities
WHERE new.care_taker = availabilities.care_taker
AND availabilities.start_date<=new.start_date
AND availabilities.end_date>= new.end_date;
INSERT INTO availabilities
SELECT new.care_taker,new.end_date+ INTERVAL '1 day'
,availabilities.end_date
FROM availabilities
WHERE new.care_taker = availabilities.care_taker
AND availabilities.start_date<=new.start_date
AND availabilities.end_date>= new.end_date;
DELETE FROM availabilities
WHERE new.care_taker = availabilities.care_taker
AND availabilities.start_date<=new.start_date
AND availabilities.end_date>= new.end_date;

RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER ct_take_leave AFTER INSERT
ON leaves FOR EACH ROW
EXECUTE PROCEDURE take_leave();
```

7. 3 Most Complex SQL Queries

- 1) Table with average price by pet type and area among care-takers, joining with base price table for comparison

```
SELECT a.pet_type, area, average_price, price AS base_price,
CASE WHEN price <= average_price THEN 1 ELSE 0 END AS ishigh
FROM base_prices b LEFT JOIN
(
SELECT pet_type AS pet_type, area, ROUND(AVG(p.price)::NUMERIC,2) AS
average_price
FROM prices p, care_takers c
GROUP BY pet_type, area
) a ON a.pet_type = b.pet_type;
```

- 2) Updating of avg_rating using given input of care_taker rounded to 2d.p.

```

WITH rating_CTE (care_taker, ave)
AS (
SELECT care_taker, ROUND (AVG(rating),2) as ave
FROM (
SELECT care_taker, rating FROM bids WHERE rating is not NULL
)T1
GROUP BY care_taker
HAVING care_taker = '$1'
)

```

```

UPDATE care_takers
SET avg_rating = (SELECT ave FROM rating_CTE)
WHERE username = (SELECT care_taker FROM rating_CTE)

```

- 3) Selecting user information from pcs-admin, care-taker and owners table and determine which user type the user belongs to (CareTaker/PCS Admin/Pet Owner)

```

SELECT users.username, users.display_name, pet_owners.username as is_owner,
care_takers.username as is_care_taker, pcs_administrators.username as is_admin
FROM users left join pcs_administrators on users.username =
pcs_administrators.username
left join pet_owners on users.username = pet_owners.username
left join care_takers on users.username = care_takers.username

```

8. Specification of frameworks used in project

- Front End: CreateReactApp
- Back End: NodeJS (Express)
- Project Module Management: Yarn
- Deployment: private server
- Database Engine: Postgres-SQL (v12)

9. Screenshots of application

Pet Care

Login

Username

Password

Login Guest

New user?
[Register here](#)

<Login Page>

Pet Care

Register

Username

Display Name

Password

Confirm Password

Register Guest

Already have an account?
[Login here.](#)

<Registration Page>

PetCare Services and Pricing Log In Register

Welcome to PetCare

PetCare provides reputable, well trained care takers for your pets, to give you a peace of mind

What we provide

PetCare connects you, with care takers for your pets, when and where you need them

To view currently available care takers and rates, visit [services](#)

Please [log in](#) or [register](#) an account to make bookings

<Main Homepage>

PetCare Services and Pricing Log Out

Current Available Services

Care taker available: ● Care taker unavailable: ●

Care Taker	Employee Type	Pet Type	Price (\$)	Area/Region	Start Date	End Date
● Lily Harrington	part-time	dog	12.89	central	01-OCT-2020	31-DEC-2020
● Lily Harrington	part-time	cat	15.20	central	01-OCT-2020	31-DEC-2020
● Lily Xu	full-time	dog	14.50	east	12-SEP-2020	31-DEC-2020
● Lily Xu	full-time	cat	13.75	east	12-SEP-2020	31-DEC-2020
● Lily Tornado	full-time	dog	11.95	central	01-APR-2019	15-JAN-2021
● Lily Tornado	full-time	cat	14.00	central	01-APR-2019	15-JAN-2021
● Alex Man Yu Cheng	full-time	dog	10.50	central	12-SEP-2017	10-AUG-2020
● Alex Man Yu Cheng	full-time	cat	11.55	central	12-SEP-2017	10-AUG-2020

Average Price based on Pet Type and Area

Average price based on offered price by care takers for each pet type and available service area. Base price set by default by PetCare system.

Pet Type	Area	Average Price (\$)	Base Price (\$)
cat	central	13.63	10.00
cat	east	13.63	10.00
cat	west	13.63	10.00
dog	central	12.44	12.00
dog	east	12.44	12.00
dog	west	12.44	12.00

<Services and Pricing Page>

10. Project Summary

The overall project was a fantastic challenge and offered many learning opportunities as we worked on it, as the entire application design had to be built up from scratch, including the schema design, constraints, front and back end web application development. However, the components of the project allowed our team members to better understand about the basic knowledge of application development with relational database implementation.

We had to revise our schema design a few times as we discovered additional constraints that we wanted to include to better reflect the business processes of PetCare. Also, along the way, we discovered certain flaws about our original schema design and thus we needed to fix these flaws and re-create our schema design. From this, we learned that having a clear initial design in the form of an understandable ER diagram was critical to designing a properly working schema which could reflect our business needs.

When designing the triggers, the syntax found online was relatively different from the ones found in the lectures thus we had to reconcile these differences and test our triggers repeatedly until they worked properly. BCNF vs 3NF analysis of our tables was relatively difficult as we were unclear about the concepts and had to clarify them during tutorial and consultations. However, this analysis helped us gain insight about properly designing our schema such that redundancy is minimised.

This project also expanded our horizons as it not just taught us about database design and implementation, it reached beyond and required us to pick up new skills by self-learning along the way, such as HTML and javascript. It helped us learn the importance and tight knit integration of interactive applications and database systems, how they work together, and how a DBMS can allow a developer to easily manage a large amount of data.