CS2102 Database Systems Project

August 2020 Semester 1

Team 54 Final Report

Name	Matric. No.
Liu Zechu	A0188295L
Ng Guangren, Ryan	A0140017Y
Ooi Xi Yi	A0171638Y
Sarah Taaher Bonna	A0171268B

Our website can be accessed at https://pet-care-service.herokuapp.com/

Our demo video can be accessed at https://youtu.be/WvYvcUh5QSs

Note: After logging out of an account, please refresh the page before logging in. This is because the browser remembers the cookies even though the cookie is supposed to be deleted.

Section 1 Project Responsibilities

Liu Zechu:

1. Design of database schema and triggers
2. Database initialisation and troubleshooting
3. Generation of mock data:
    a. pet
    b. bid_transaction
    c. leave_days
    d. availabilities
4. Frontend and Backend of:
    a. Account creation (Sign Up) for Pet Owners, Full-time Caretakers and Part-time Caretakers
    b. User authentication (Log In) for Pet Owners, Full-time Caretakers and Part-time Caretakers
    c. View a Caretaker's past reviews
    d. Submit and edit a review for a Caretaker with regards to a job
    e. PCS Admin sign up and log in
    f. Set and edit base daily prices for different animal types by a PCS admin
    g. Leave Application for full-time Caretakers
    h. Viewing and deleting leaves for full-time Caretakers

Ng Guangren, Ryan:

1. Design of Database Schema and triggers
2. Frontend and Backend of:
    a. Display number of pet-days, number of pets taken care of by each caretaker and salary for each caretaker for the current month.

Ooi Xi Yi:

1. Design of Database Schema and triggers
2. Generation of mock data:
    a. can_take_care
3. Frontend and Backend of:
    a. Bidding of caretaker by pet owner and transaction
    b. Full-time caretaker summary information in the home page
    c. Part-time caretaker summary information in the home page

Sarah Taaher Bonna:

1. Design of Database Schema and triggers
2. Generation of Mock Data:
    a. users
    b. pet_owners
    c. caretakers
    d. full_time_caretaker
    e. part_time_caretaker
3. Frontend and Backend of:
    a. View and edit Pet Owner profile
    b. View, edit, and delete Pet profile by Pet Owner
    c. Pet Owner view past, ongoing and upcoming jobs
    d. Pet Owner browse/search caretakers
    e. View and edit Part-time Caretaker profile
    f. Part-time Caretaker view past, ongoing and upcoming jobs
    g. Part-Time Caretaker set availability
    h. View and edit Full-time Caretaker profile
    i. Full-time Caretaker view past, ongoing and upcoming jobs

j. View and Edit PCS Admin Profile
k. Display summary information for PCS Admin

Section 2 Description of Application

Section 2.1 Data Requirements and Data Constraints

1. There are two types of users: Pet Owner and Caretaker. A Caretaker can be either full-time or part-time (but not both at the same time).
2. In addition to the users, there are PCS administrators.
3. A user can be both a Pet Owner and a Caretaker at the same time.
4. Each Pet Owner can have any number of Pets but each Pet can only belong to one owner.
5. Each Pet Owner/Caretaker/Pet has a profile.
6. A Caretaker should not take care of pets they cannot care for.
7. When a bid is made, Pet Owner's proposed start date and proposed end date should fall between the availability period of the Caretaker and the Pet Owner's Pet must satisfy the Caretaker's requirements regarding which animal types they can take care of.
8. The cost of caring for a Pet is the number of days × the daily price of that specific pet type for the Caretaker. A Pet Owner needs to pay this cost upfront, which is the amount reflected in a transaction.
9. Each full-time Caretaker must work for a minimum of 2 × 150 consecutive days a year.
10. A full-time Caretaker cannot apply for leave if there is at least one Pet under their care.
11. A full-time Caretaker has a limit of up to 5 pets at any one time.
12. When bid by any Pet Owner, a full-time or part-time Caretaker will always accept the job immediately if possible.
13. The Pet Owner that bids for a specific time slot for a specific Caretaker the earliest is successful.
14. For each part-time Caretaker, they should be able to specify their availability for the current year and the next year.
15. At any single point in time, a part-time Caretaker can take care of up to 2 pets if their average rating is less than 4.0. Otherwise, they can take care of up to 4 pets.
16. Current daily price for a Caretaker increases with their rating but will never be below the base price for the animal type.
17. A full-time Caretaker will receive a salary of $3000 per month for up to 60 pet-days. For any excess pet-day, they will receive 80% of their current daily price as bonus.
18. For a part-time Caretaker, the PCS will take 25% of their price as payment (i.e., the Caretaker receives 75% of their stated price).
19. A Caretaker's rating is the average of all of his/her past ratings.
20. A Caretaker's current daily price increases with their average rating.

Section 2.2 Functionalities

The following are what users can do when using our application:

1. Sign up to be a pet owner only, full-time caretaker only, part-time caretaker only, both pet owner and full-time caretaker, both pet owner and part-time caretaker or PCS admin.
2. Log in as a pet owner, full-time caretaker, part-time caretaker, or PCS admin (depending on what they signed up as)
3. Pet Owners are able to:
    a. View current and upcoming events on the home page
    b. View past, ongoing, and upcoming jobs
    c. Submit reviews and ratings of caretakers under the past jobs page
    d. Browse/Search Caretakers (Filter according to caretaker commitment level, date range (starting date and ending date of caretaking planned), caretaker rating, minimum price, maximum price, pet type and search by caretaker username) Limited to 20 (or less depending on number of caretakers found fitting the pet owner's specifications), caretaker profiles fitting

the conditions specified (if any) are shown randomly. If a username is searched for, the other conditions specified (if any) are not looked at.

e. View reviews of individual caretakers
f. Bid for a caretaker
g. View and edit their own profile
    i. Change password
    ii. Change name, gender, email address, phone number
    iii. Add, edit, or delete credit card details
h. View, add, edit, and delete pet profile
    i. Pet name (cannot edit)
    ii. Birthdate (cannot edit)
    iii. Gender (cannot edit)
    iv. Breed (cannot edit)
    v. Type of animal (cannot edit)
    vi. Medical history
    vii. Special requirements

4. Full-time caretakers are able to:
    a. View number of pets, the corresponding number of pet-days and the amount they have earned so far in the month on the home page, as well as the current and upcoming events.
    b. View past, ongoing and upcoming jobs
    c. View and edit their own profile
        i. Change password
        ii. Change name, gender, email address, phone number
        iii. View the pet types they can take care of and the current price for that pet type (cannot edit)
    d. View their leave days
    e. Apply for leave
    f. Delete a leave application
5. Part-time caretakers are able to:
    a. View number of pets, the corresponding number of pet-days and the amount they have earned so far in the month on the home page, as well as the current and upcoming events.
    b. View past, ongoing and upcoming jobs
    c. View and edit their own profile
        i. Change password
        ii. Change name, gender, email address, phone number
        iii. View the number of pets, pet types they can take care of, and the current price for that pet type (cannot edit)
    d. View, add, edit and delete their availabilities
6. PCS Admins are able to:
    a. View and Edit their own profile
    b. Set/Edit the base daily price of each pet type
    c. View summary information for the current month (number of pet-days, the amount earned from caretakers' jobs, and total caretaker salary so far)
    d. View the number of pet-days, number of pets taken care of, and expected salary for each caretaker in the current month in a table
    e. View each caretaker's profile by clicking on the row in the table in part d containing the caretaker's name
    f. View reviews of the caretaker from the caretaker's profile

Section 2.3 Interesting/Non-trivial aspects of the Functionalities

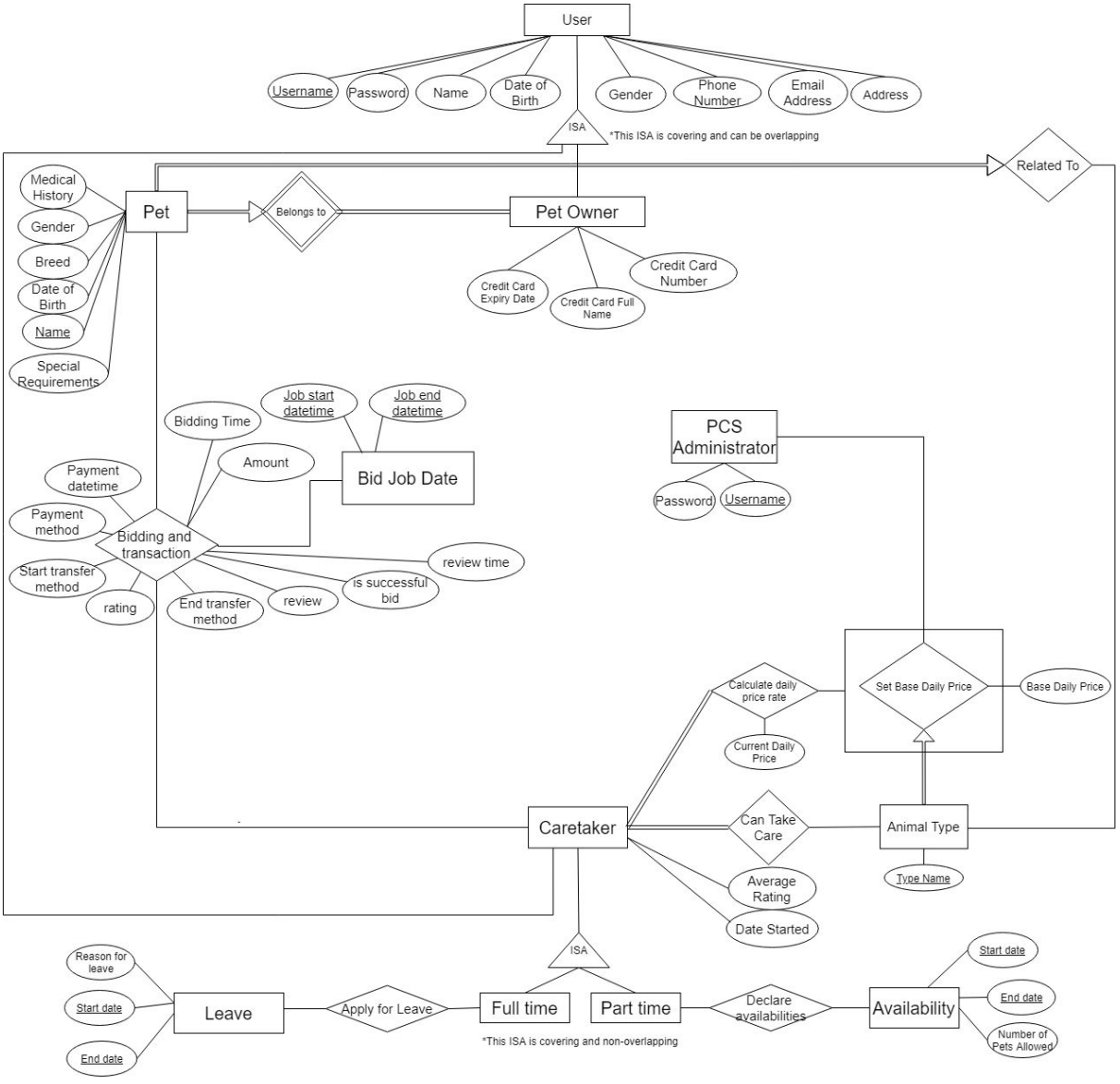
Some interesting aspects of the functionalities are as follows:

1. The caretakers in the Caretakers' Profiles page (for pet owners to browse/search caretakers) are shown randomly, 20 each time. Pet owners are able to filter the caretakers according to their needs using the metrics provided, for example, by caretaker commitment level, date range (starting date and ending date of caretaking planned), caretaker rating, minimum price, maximum price, search by caretaker username and pet type. Results shown are limited to 20 (or less depending on the number of caretakers found fitting the pet owner's specifications) and caretaker profiles fitting the conditions specified are shown randomly. If a username is searched for, the other conditions specified (if any) are not looked at.
2. PCS admins are able to view the number of pet-days, salary and number of pets taken care of by each caretaker in a table for easy viewing. A click on a row in the table leads to the caretaker's profile. From the profile, the PCS admin is able to read the caretaker's reviews by pet owners (if any).

Section 3 ER Model, Relational Schema and NF

Section 3.1 ER Model

The following is the final ER Model of our application:



The constraints not captured by our ER model are as follows:

1. For bid transaction:
    a. The end date-time of the job should be greater than or equal to the start date-time of the job.
    b. The date-time of the payment should be less than or equal to the job start date-time.
    c. The bidding time should be less than or equal to the payment date-time.
    d. A pet cannot have overlapping bids.
2. For availabilities, the number of pets allowed for each part-time caretaker is either 2 or 4.
3. For leave days, the start date must be less than or equal to the end date.
4. A part-time caretaker is able to take care of 2 or 4 pets at any one time, depending on their average rating (if average rating < 4.0, then 2 pets, else 4 pets).
5. A full-time caretaker can take care of 5 pets at any one point.
6. A caretaker's average rating changes based on the ratings provided by the pet owners.
7. A caretaker's current daily price rate changes according to the caretaker's average rating, but never goes below the base daily price.
8. When a bid is made, the pet owner's proposed start date and proposed end date should fall between the availability period of the caretaker (if part-time) and outside of leave dates (if full-time) and the pet owner's pet must satisfy the caretaker's requirements (pet type should match the pet type the caretaker can take care of).
9. The cost of caring for a pet is the number of days * the daily price of that specific pet type for the caretaker. A pet owner needs to pay this cost upfront, which is the amount reflected in a transaction.
10. Each full-time caretaker must work for a minimum of 2 × 150 consecutive days a year.
11. When bid by any pet owner, a full-time caretaker will always accept the job immediately if possible.
12. For each part-time Caretaker, they should be able to specify their availability for the current year and the next year.
13. A full-time Caretaker will receive a salary of $3000 per month for up to 60 pet-days. For any excess pet-day, they will receive 80% of their current daily price as bonus.
14. For a part-time Caretaker, the PCS will take 25% of their price as payment (i.e., the Caretaker receives 75% of their stated price).

Section 3.2 Relational Schema

```
CREATE TABLE animal_type (
   type_name VARCHAR PRIMARY KEY
);

CREATE TABLE users (
   username VARCHAR PRIMARY KEY,
   password VARCHAR NOT NULL,
   name VARCHAR NOT NULL,
   birth_date DATE NOT NULL,
   gender VARCHAR NOT NULL,
   phone INT NOT NULL,
   email VARCHAR NOT NULL,
   address VARCHAR NOT NULL
);

CREATE TABLE pet_owner (
  username VARCHAR PRIMARY KEY REFERENCES users(username) ON DELETE cascade,
  credit_card_number VARCHAR UNIQUE,
  credit_card_full_name VARCHAR,
```

```
  credit_card_expiry_date VARCHAR(5),
  CHECK ((credit_card_number IS NULL AND credit_card_full_name IS NULL
        AND credit_card_expiry_date IS NULL)
  OR (credit_card_number IS NOT NULL AND credit_card_full_name IS NOT NULL
      AND credit_card_expiry_date IS NOT NULL))
);

CREATE TABLE caretaker (
  username VARCHAR PRIMARY KEY REFERENCES users(username) ON DELETE cascade,
  average_rating NUMERIC(2, 1) NOT NULL DEFAULT 0.0,
  date_started DATE NOT NULL
);

CREATE TABLE full_time_caretaker (
  username VARCHAR PRIMARY KEY REFERENCES caretaker(username) ON DELETE cascade
);

CREATE TABLE part_time_caretaker (
  username VARCHAR PRIMARY KEY REFERENCES caretaker(username) ON DELETE cascade
);

CREATE TABLE pet (
  username VARCHAR NOT NULL REFERENCES pet_owner(username) ON DELETE cascade,
  pet_name VARCHAR NOT NULL,
  birth_date DATE NOT NULL,
  breed VARCHAR NOT NULL,
  type_of_animal VARCHAR NOT NULL REFERENCES animal_type(type_name),
  gender VARCHAR NOT NULL,
  med_hist VARCHAR,
  special_req VARCHAR,
  PRIMARY KEY (username, pet_name)
);

CREATE TABLE bid_transaction (
  pusername VARCHAR,
  pet_name VARCHAR,
  cusername VARCHAR REFERENCES caretaker(username),
  bidding_time TIMESTAMP NOT NULL,
  job_start_datetime TIMESTAMP,
  job_end_datetime TIMESTAMP,
  payment_datetime TIMESTAMP,
  amount NUMERIC(7, 2),
  payment_method VARCHAR ,
  start_transfer_method VARCHAR,
```

```
    end_transfer_method VARCHAR,
    is_successful_bid BOOLEAN,
    review_time TIMESTAMP,
    review VARCHAR,
    rating INT,
    FOREIGN KEY (pusername, pet_name) REFERENCES pet(username, pet_name),
    PRIMARY KEY (pusername, cusername, pet_name, job_start_datetime, job_end_datetime),
    CHECK (job_end_datetime >= job_start_datetime),
    CHECK (payment_datetime <= job_start_datetime),
    CHECK (bidding_time <= payment_datetime)
);

CREATE TABLE pcs_administrator (
    username VARCHAR PRIMARY KEY,
    password VARCHAR NOT NULL
);

CREATE TABLE set_base_daily_price (
    type_name VARCHAR PRIMARY KEY REFERENCES animal_type(type_name),
    base_daily_price INT NOT NULL,
    admin_username VARCHAR NOT NULL REFERENCES pcs_administrator(username)
);

CREATE TABLE can_take_care (
    username VARCHAR REFERENCES caretaker(username),
    type_name VARCHAR REFERENCES animal_type(type_name),
    PRIMARY KEY (username, type_name)
);

CREATE TABLE daily_price_rate(
    username VARCHAR REFERENCES caretaker(username),
    type_name VARCHAR REFERENCES set_base_daily_price(type_name),
    current_daily_price NUMERIC(7, 2),
    PRIMARY KEY(username, type_name)
);

CREATE TABLE availabilities(
    username VARCHAR REFERENCES part_time_caretaker(username),
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    number_of_pets_allowed INTEGER NOT NULL ,
    PRIMARY KEY (username, start_date, end_date),
    CHECK (number_of_pets_allowed = 2 OR number_of_pets_allowed = 4)
);
```

```
CREATE TABLE leave_days(
  username VARCHAR NOT NULL REFERENCES full_time_caretaker(username),
  reason_for_leave VARCHAR NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  PRIMARY KEY(username, start_date, end_date),
  CHECK (start_date <= end_date)
);
```

The constraints not captured by the Relational Schema above are as follows:

1. Checking whether a potential job start date-time and end date-time fall between the available dates for a part-time caretaker or outside of leave dates for a full-time caretaker before a bid is confirmed. (Enforced using triggers)
2. Updating the average rating of a caretaker when a pet owner submits a new rating. (Enforced using triggers).
3. Update the number of pets a part-time caretaker can take care of once the rating changes (if less than 4.0, then 2 pets allowed, else, 4 pets allowed). (Enforced using triggers)
4. Update the current daily price of a caretaker for all the pet types he/she can take care of if the average rating changes. (If the average rating is less than or equal to 2.0, then the current daily price for all pet types the caretaker can take care of is equal to the base daily price for those pet types. Else, the price increase is calculated by subtracting 2.0 from the new average rating, and dividing the value by 0.5, to get the number of 0.5 points the average rating increased by before multiplying by 10 ($10). This value is added to the base daily price for each of the pet types the caretaker can take care of.) (Enforced using triggers)
5. Ensuring that there is no overlap between part-time and full-time caretakers. (Enforced using triggers)
6. Updating the current daily price for a pet type for each caretaker if the base daily price of a pet type he/she can take care of changes. (The price difference between the new value and old value is calculated, and this value is added to the current daily price of the caretakers taking care of that pet type.) (Enforced using triggers)
7. Checking if a full-time caretaker has any jobs or has already applied for leaves on the dates the leaves have already been applied for. (Enforced using triggers)
8. Checking if the pet limit for a caretaker is exceeded at any point in time of the potential job start and end date-time. (Enforced using Javascript and SQL code in the backend)
9. Checking if the full-time caretaker is able to fulfill the 2 * 150 consecutive days before the leave is submitted. (Enforced using Javascript and SQL code in the backend)
10. Checking if pets cannot have overlapping bids (Enforced using Javascript and SQL code in the backend)
11. Checking if proposed job start date-time and end date-time is on or after the current time (Enforced using Javascript and SQL code in the backend)

Section 3.3 BCNF or 3NF

All the tables in our database schema are in the Boyce-Codd Normal Form (BCNF). All attributes in each of these BCNF tables are directly dependent on the primary key of the table. In particular, the pet_owner table (shown below) is also in BCNF. Although credit_card_number functionally determines the attributes credit_card_full_name and credit_card_expiry_date, credit_card_number is unique and thus can uniquely identify the other attributes, serving as a superkey. The reason for disallowing two pet owners to share the same credit card is for security considerations. Each credit card should only be used by a registered user so as to prevent potential fraudulent transactions. The pet_owner table satisfies the BCNF properties as well.

```
CREATE TABLE pet_owner (
  username VARCHAR PRIMARY KEY REFERENCES users(username) ON DELETE cascade,
  credit_card_number VARCHAR UNIQUE,
  credit_card_full_name VARCHAR,
  credit_card_expiry_date VARCHAR(5),
  CHECK ((credit_card_number IS NULL AND credit_card_full_name IS NULL
        AND credit_card_expiry_date IS NULL)
  OR (credit_card_number IS NOT NULL AND credit_card_full_name IS NOT NULL
     AND credit_card_expiry_date IS NOT NULL))
);
```

Section 4 Non-trivial/Interesting triggers

Trigger 1:

```
CREATE OR REPLACE FUNCTION not_overlap()
  RETURNS TRIGGER AS
  $$ DECLARE ctx1 NUMERIC; ctx2 NUMERIC; part_time_exists BOOLEAN; leave_start DATE; leave_end
DATE;
  BEGIN
     part_time_exists := (SELECT EXISTS (SELECT 1 FROM part_time_caretaker WHERE username =
NEW.cusername));

     IF part_time_exists THEN
        SELECT COUNT(*) INTO ctx1 FROM availabilities A
        WHERE NEW.cusername = A.username AND
        (NEW.job_start_datetime >= A.start_date AND NEW.job_end_datetime <  (A.end_date + INTERVAL
'1 day') );

        IF ctx1 = 0 THEN
           RAISE EXCEPTION 'We regret to inform you that % is unavailable from % to %', NEW.cusername,
NEW.job_start_datetime, NEW.job_end_datetime;
        ELSE
           RETURN NEW;
        END IF;

     ELSE
     SELECT COUNT(*) INTO ctx2 FROM leave_days L
     WHERE NEW.cusername = L.username AND
        (NEW.job_start_datetime, NEW.job_end_datetime) overlaps (L.start_date, (L.end_date + INTERVAL '1
day') );

     IF ctx2 > 0 THEN
        SELECT L.start_date, L.end_date INTO leave_start, leave_end
        FROM leave_days L
        WHERE NEW.cusername = L.username AND
```

```
            (NEW.job_start_datetime, NEW.job_end_datetime) overlaps (L.start_date, (L.end_date +
INTERVAL '1 day') )
          LIMIT 1;
          RAISE EXCEPTION 'We regret to inform you that % will be on leave from % to %.', NEW.cusername,
leave_start, leave_end;
       ELSE
          RETURN NEW;
       END IF;

   END IF; END; $$
LANGUAGE plpgsql;

CREATE TRIGGER check_overlap
BEFORE INSERT OR UPDATE ON bid_transaction
FOR EACH ROW EXECUTE PROCEDURE not_overlap();
```

Explanation:
This trigger checks whether the date range of a new bid_transaction falls within the availability periods of the Caretaker involved in this bid. Since part-time caretakers' available periods are recorded in the availabilities table whereas full-time caretakers' available periods are any time periods outside the leave_days table, we need to consider both cases separately. The part_time_exists variable is a Boolean that indicates whether the given caretaker is a part-time caretaker.

If the caretaker is a part-time caretaker, we check whether the proposed job date range falls within any of the caretaker's declared availability period. If the job does not fall within any availability, the insertion/update fails and an exception is raised. Otherwise, the insertion/update will be successful.

Else, the caretaker is a full-time caretaker. In this case, we check whether the proposed job date range overlaps with any of the Caretaker's leave days. If it does, the insertion/update fails and an exception is raised (since a full-time caretaker cannot take up a job when he/she is on leave). Otherwise, the insertion/update will be successful.

We decided to raise an exception instead of returning NULL so as to provide more information to the frontend about the reason for insertion/update failure, so that the potential client knows why the bid did not go through.

Trigger 2:

```
CREATE OR REPLACE FUNCTION new_current_daily_price_rate()
RETURNS TRIGGER
AS $$
DECLARE price_increase NUMERIC;
BEGIN
  IF NEW.average_rating != OLD.average_rating THEN
    IF NEW.average_rating <= 2.0 THEN
      UPDATE daily_price_rate
      SET current_daily_price = S.base_daily_price
      FROM can_take_care AS C NATURAL JOIN set_base_daily_price AS S
```

```
      WHERE C.username = NEW.username AND daily_price_rate.username = NEW.username AND
daily_price_rate.type_name = S.type_name;

    ELSE
      price_increase := ((NEW.average_rating - 2.0)/0.5) * 10;
      UPDATE daily_price_rate
      SET current_daily_price = S.base_daily_price + price_increase
      FROM can_take_care AS C NATURAL JOIN set_base_daily_price AS S
      WHERE C.username = NEW.username AND daily_price_rate.username = NEW.username AND
daily_price_rate.type_name = S.type_name;
    END IF;
  END IF;
  RETURN NULL;
END; $$
LANGUAGE plpgsql;

CREATE TRIGGER update_daily_price
AFTER UPDATE ON caretaker
FOR EACH ROW EXECUTE FUNCTION new_current_daily_price_rate();
```

Explanation:
This trigger updates the daily price rates of a caretaker when his/her rating changes. When the average rating of a caretaker changes (due to a new/updated rating entry, which is taken care of by a separate trigger shown below), the daily price rates corresponding to each animal type will be adjusted according to the following rules: If the average rating is less than or equal to 2.0, then the current daily price for all pet types the Caretaker can take care of is equal to the base daily price for those pet types. Else, the price increase is calculated by subtracting 2.0 from the new average rating, and dividing the value by 0.5, to get the number of 0.5 points the average rating increased by before multiplying by 10 ($10). This value is added to the base daily price for each of the pet types the caretaker can take care of.

Trigger 3:

```
CREATE OR REPLACE FUNCTION update_avg_rating_of_caretaker()
RETURNS TRIGGER
AS $$
  DECLARE avg_rating NUMERIC;
  BEGIN
    avg_rating := (SELECT AVG(rating) FROM bid_transaction B WHERE B.cusername = NEW.cusername);
    IF avg_rating IS NOT NULL THEN
    UPDATE caretaker SET average_rating = avg_rating WHERE username = NEW.cusername;
    END IF;
    RETURN NULL;
END; $$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_avg_rating
AFTER INSERT OR UPDATE ON bid_transaction
FOR EACH ROW EXECUTE FUNCTION update_avg_rating_of_caretaker();
```

Explanation:
This trigger updates the average_rating attribute of a Caretaker based on the bid_transaction table where the numerical rating of each job is stored. The average value of all the ratings corresponding to a given Caretaker is first calculated, and then the average_rating attribute in the caretaker table is updated accordingly for the given Caretaker's username.

Note:
For the full list of our triggers, please refer to the file in our GitHub repository located at /seeder/sql_create_statements.sql , Line 150 onwards.

Section 5 Most Interesting SQL Queries

SQL Query 1:

```
let general_query = `SELECT X.username AS username, X.name AS name, X.age AS age, X.birth_date AS
birth_date, X.gender AS gender,
      X.phone AS phone, X.email AS email, X.address AS address, X.average_rating AS average_rating,
X.years_exp AS years_exp FROM (`;

      let request_full_time = `SELECT T.username AS username, T.name AS name, AGE(T.birth_date) AS
age,
      T.birth_date AS birth_date, T.gender AS gender, T.phone AS phone, T.email AS email,
      T.address AS address, T.average_rating AS average_rating, AGE(T.date_started) AS years_exp
      FROM (users NATURAL JOIN caretaker NATURAL JOIN full_time_caretaker NATURAL JOIN
can_take_care
       NATURAL JOIN daily_price_rate) AS T WHERE`;

      let request_part_time = `SELECT username, name, AGE(birth_date) AS age, birth_date, gender,
    phone, email, address, average_rating, AGE(date_started) AS years_exp FROM users NATURAL JOIN
    caretaker NATURAL JOIN part_time_caretaker NATURAL JOIN can_take_care
    NATURAL JOIN availabilities NATURAL JOIN daily_price_rate WHERE`;

      if (date_from != null && date_to != null) {
       let add_dates_requested_full_time = ` ((SELECT COUNT(*) FROM leave_days L1 WHERE
       L1.username = T.username) = 0 OR
       (SELECT COUNT(*) FROM leave_days L WHERE L.username = T.username
       AND (date '${date_from}', date '${date_to}') OVERLAPS (L.start_date, L.end_date)) = 0)`;

       let add_dates_requested_part_time =
        " (start_date <= date('" +
        date_from +
        "') AND end_date >= date('" +
```

```
    date_to +
    "'))";

  request_full_time += add_dates_requested_full_time + " AND";

  request_part_time += add_dates_requested_part_time + " AND";
 }

 if (rating_wanted != null) {
  let add_rating_requested_full_time =
   " (T.average_rating >= " +
   parseFloat(rating_wanted).toString() +
   ")";

  let add_rating_requested_part_time =
   " (average_rating >= " +
   parseFloat(rating_wanted).toString() +
   ")";

  request_full_time += add_rating_requested_full_time + " AND";

  request_part_time += add_rating_requested_part_time + " AND";
 }

 if (type_of_animal != null) {
  type_of_animal_full_time = type_of_animal.replace(
   /,/g,
   "' OR T.type_name = '"
  );
  type_of_animal_part_time = type_of_animal.replace(
   /,/g,
   "' OR type_name = '"
  );

  let add_animal_type_requested_full_time =
   " (T.type_name = '" + type_of_animal_full_time + "')";

  let add_animal_type_requested_part_time =
   " (type_name = '" + type_of_animal_part_time + "')";

  request_full_time += add_animal_type_requested_full_time + " AND";

  request_part_time += add_animal_type_requested_part_time + " AND";
 }
```

```
if (price_range_from != null && price_range_to == null) {
  let add_min_price_full_time =
    " (T.current_daily_price >= " +
    parseFloat(price_range_from).toString() +
    ")";

  let add_min_price_part_time =
    " (current_daily_price >= " +
    parseFloat(price_range_from).toString() +
    ")";

  request_full_time += add_min_price_full_time + " AND";

  request_part_time += add_min_price_part_time + " AND";
} else if (price_range_from == null && price_range_to != null) {
  let add_max_price_full_time =
    " (T.current_daily_price <= " +
    parseFloat(price_range_to).toString() +
    ")";

  let add_max_price_part_time =
    " (current_daily_price <= " +
    parseFloat(price_range_to).toString() +
    ")";

  request_full_time += add_max_price_full_time + " AND";

  request_part_time += add_max_price_part_time + " AND";
} else if (price_range_from != null && price_range_to != null) {
  let add_price_range_full_time =
    " (T.current_daily_price BETWEEN " +
    parseFloat(price_range_from).toString() +
    " AND " +
    parseFloat(price_range_to).toString() +
    ")";

  let add_price_range_part_time =
    " (current_daily_price BETWEEN " +
    parseFloat(price_range_from).toString() +
    " AND " +
    parseFloat(price_range_to).toString() +
    ")";
```

```
  request_full_time += add_price_range_full_time + " AND";

  request_part_time += add_price_range_part_time + " AND";
 }

 let request_full_time_split_by_space = request_full_time
  .trim()
  .split(" ");

 if (
  request_full_time_split_by_space[
   request_full_time_split_by_space.length - 1
  ] == "AND" ||
  request_full_time_split_by_space[
   request_full_time_split_by_space.length - 1
  ] == "WHERE"
 ) {
  request_full_time = request_full_time_split_by_space
   .slice(0, -1)
   .join(" ");
 } else {
  request_full_time = request_full_time_split_by_space.join(" ");
 }

 let request_part_time_split_by_space = request_part_time
  .trim()
  .split(" ");

 if (
  request_part_time_split_by_space[
   request_part_time_split_by_space.length - 1
  ] == "AND" ||
  request_part_time_split_by_space[
   request_part_time_split_by_space.length - 1
  ] == "WHERE"
 ) {
  request_part_time = request_part_time_split_by_space
   .slice(0, -1)
   .join(" ");
 } else {
  request_part_time = request_part_time_split_by_space.join(" ");
 }

 general_query +=
```

```
        request_full_time +
        " UNION " +
        request_part_time +
        ") AS X ORDER BY random() LIMIT 20;";
```

Explanation:
The above code is from pet_owner_view_caretaker_routes.js. This piece of code selects 20 (or less, depending on the number of caretakers that fit the pet owner's specifications) caretakers (both full-time and part-time) at random using inputs provided by the pet owner when trying to filter the caretakers via the metrics provided (i.e. availability, rating, minimum price, maximum price and animal type). There are 2 nested subqueries. The inner nested subquery finds full-time caretakers that match the conditions provided by the pet owner, for example, the dates the pet owner is interested in falls outside of the leave dates (if any) of the caretakers. The outer nested subquery obtains the full-time and part-time caretakers that match the pet owner's specifications and select 20 randomly. This allows the caretakers to have a chance to be featured at the top of the list of caretakers, allowing pet owners to discover them and bid for their services.

SQL Query 2:

```
SELECT SFT.cusername,
    (SELECT COUNT(*)
     FROM bid_transaction BT
     WHERE BT.job_end_datetime >= DATE_TRUNC('MONTH', NOW()) AND
      BT.job_end_datetime <=  (SELECT (now() at time zone 'sgt'))
      AND BT.cusername = SFT.cusername)
    AS num_pets,
    COALESCE((SELECT SUM(pet_days)
            FROM pet_days_past_30_days PD
            WHERE PD.cusername = SFT.cusername), 0)
    AS num_pet_days,
    SFT.salary AS salary
FROM salary_calculation_for_full_time SFT

UNION

SELECT SPT.cusername,
    (SELECT COUNT(*)
     FROM bid_transaction BT
     WHERE BT.job_end_datetime >= DATE_TRUNC('MONTH', NOW()) AND
      BT.job_end_datetime <=  (SELECT (now() at time zone 'sgt'))
      AND BT.cusername = SPT.cusername)
    AS num_pets,
    COALESCE((SELECT SUM(pet_days)
            FROM pet_days_past_30_days PD
            WHERE PD.cusername = SPT.cusername), 0)
    AS num_pet_days,
```

```
    SUM(SPT.salary) AS salary
FROM salary_calculation_for_part_time SPT
GROUP BY SPT.cusername

ORDER BY num_pets DESC, num_pet_days DESC;
```

Explanation:
This query obtains the summary information for all caretakers for PCS administrators to view. Specifically, it displays the total number of pets taken care of, the total number of pet-days, and salary earned so far for each caretaker in the current month. This query considers the jobs whose end dates fall between the start of the current month and the present time. The results from full-time caretakers are unioned with the results from part-time caretakers, before being sorted first by the number of pet days and then by the number of pets taken care of.

SQL Query 3:

```
CREATE VIEW pet_days_past_30_days(cusername,pet_days) AS (
  SELECT cusername, SUM(pet_days)
  FROM (SELECT cusername,
            DATE_PART('DAY', job_end_datetime - job_start_datetime) + 1
            AS pet_days,
            job_end_datetime
            FROM bid_transaction)
  WHERE job_end_datetime >= DATE_TRUNC('MONTH', NOW())
      AND job_end_datetime <=  (SELECT (now() at time zone 'sgt'))
  GROUP BY cusername
);

CREATE VIEW salary_calculation_for_full_time (cusername, salary) AS (
  SELECT F.username, CASE
    WHEN PD.pet_days <= 60 OR PD.pet_days IS NULL THEN 3000
    ELSE 3000 + (PD.pet_days - 60) * (SELECT AVG(current_daily_price) FROM daily_price_rate DPR
WHERE DPR.username = F.username) * 0.8
    END AS salary
  FROM full_time_caretaker F LEFT JOIN pet_days_past_30_days PD ON F.username = PD.cusername
);
```

Explanation:
The view salary_calculation_for_full_time shows how the salary for each full-time caretaker is calculated. A full-time caretaker with not more than 60 pet-days in the current month will get $3000, and those who work more than 60 pet-days will get 80% of their average current daily price rates per pet-day as bonus. This view depends on another view pet_days_past_30_days, which calculates the total number of pet-days for each caretaker from jobs that end after the start of the current month and before the present time. Please note that pet_days_past_30_days is not a single-purpose intermediate view. Many other queries use this view too.

Section 6 Software/Tools Used

Database: PostgreSQL
Backend: Node.js with Express
Frontend: Vue.js with Vuetify UI library
Hosting: Heroku

Section 7 Representative Screenshots

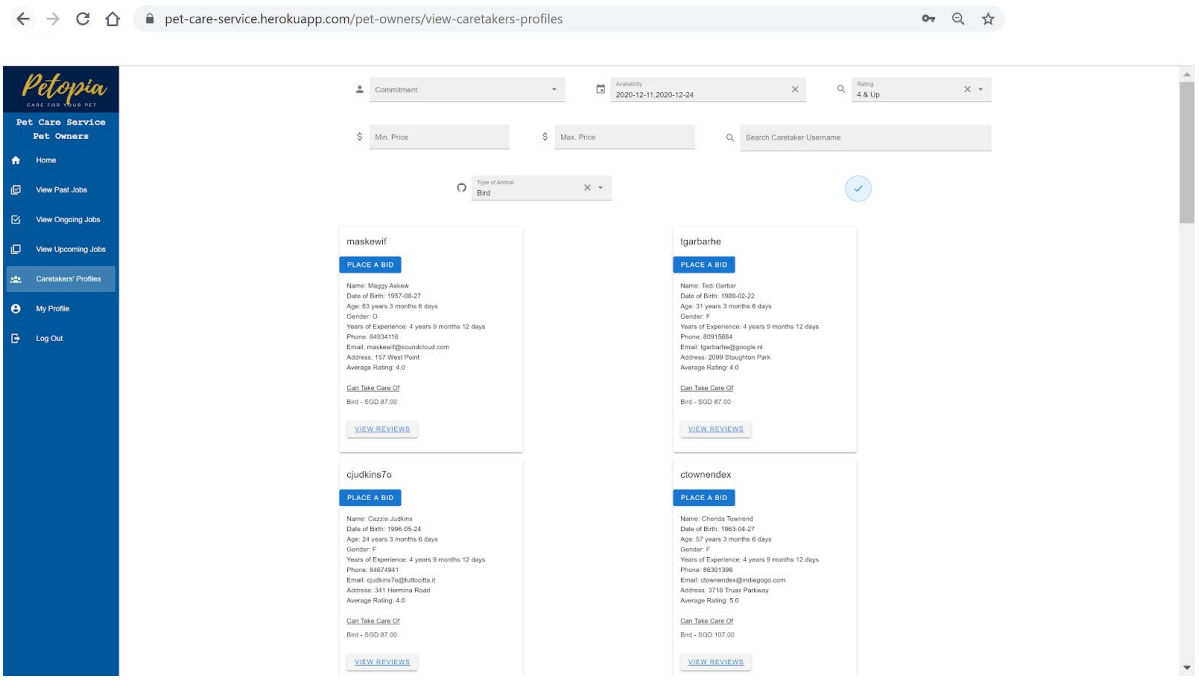


Figure 1: A zoomed-out screenshot of the Caretaker's Profiles page, for pet owners to browse/search for caretakers

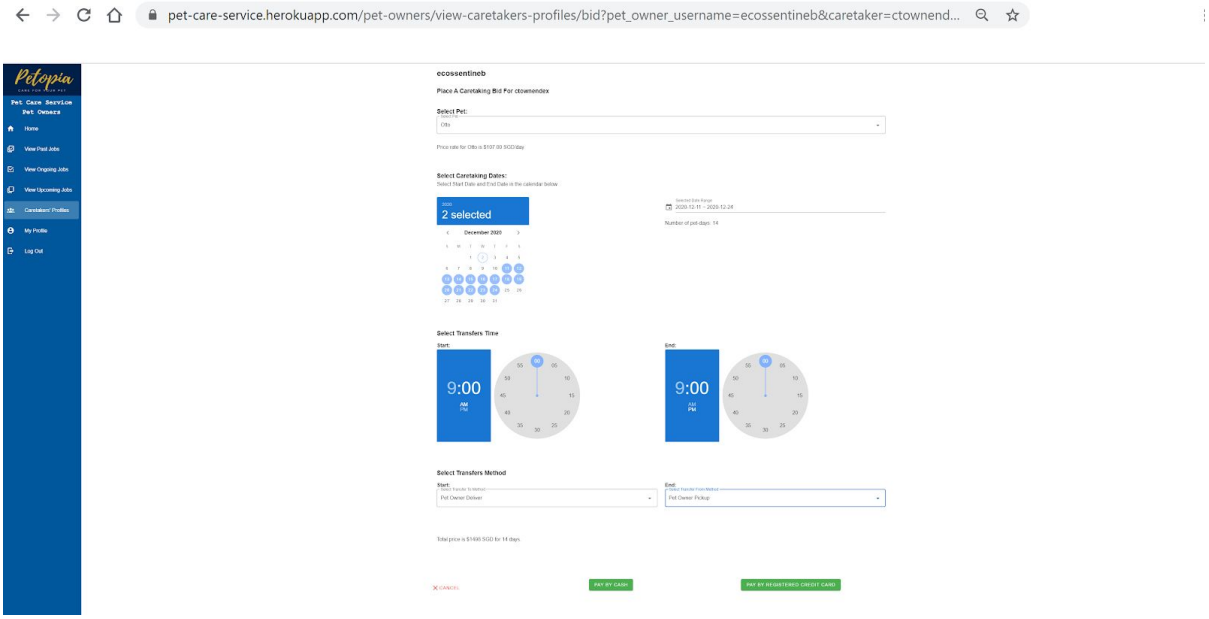


Figure 2: A zoomed-out screenshot of the bidding page

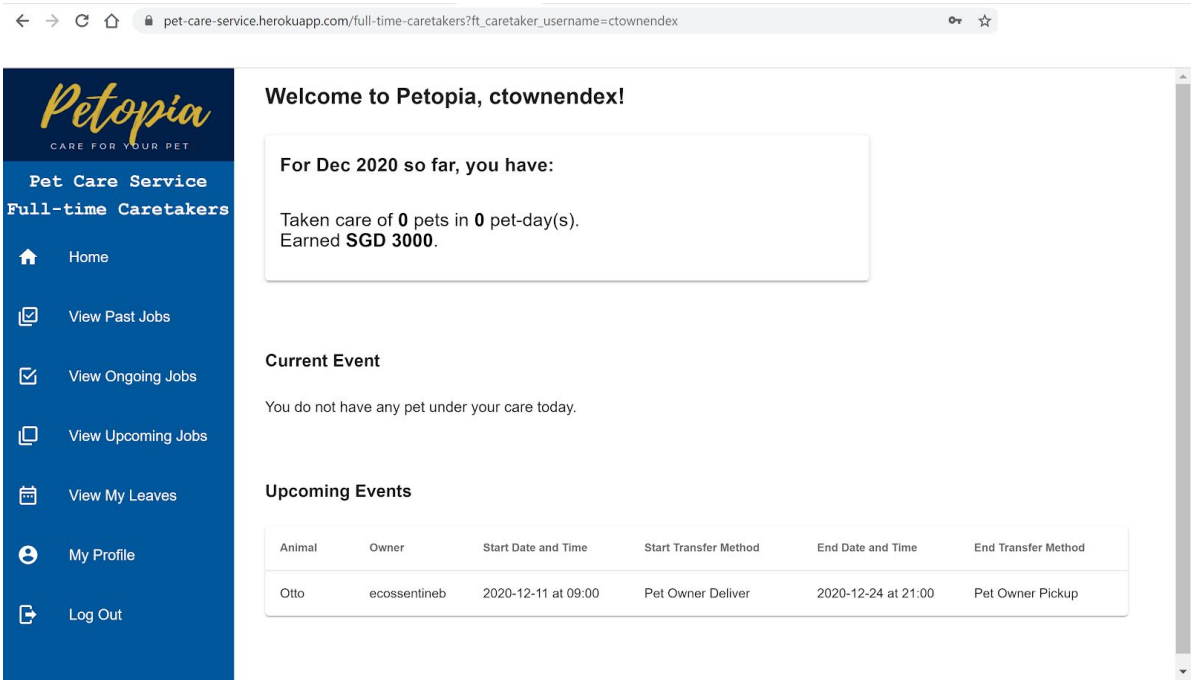


Figure 3: A screenshot of a full-time caretaker's home page, displaying any current and upcoming jobs

Section 8 Summary

Section 8.1 Difficulties Faced

The difficulties we faced are as follows:

1. Having to learn full-stack web development from scratch along the way. It takes time to resolve the technical issues we encounter as we are not familiar or perhaps not aware of the tools in web development.
2. Having to finish so many components of the web application in a short amount of time.

Section 8.2 Lessons Learnt

Here are the lessons we learnt:

1. It is important to first understand the requirement of our application and then plan the database before actually coding it. An ER diagram is helpful to achieve this.
2. Different components of the web application are interrelated. As such, although the work is split amongst team members, it is essential to come together and communicate with each other.