NUS School of Computing
CS2102 Database Systems
AY20/21 Semester 1
Course Coordinator - A/P Adi Yoga Sidi Prabawa

Group Project Report - Pet Caring Services (PCS) Application

Group Number : 21
Group Members:

- Enelton Satria (A0173350M)
- Dorcas Tan (A0190356A)
- Kelvin Wong (A0201706U)
- Wincent Tjoi (A0201480W)
- Yang Jiyu (A0199476E)

Table of Contents

1. Project Responsibilities

Group Member	Project Responsibilities and Contributions
Enelton Satria	• Contributed to the preliminary ER diagram and listing of functional requirements and data constraints • Contributed to the SQL Create Table Queries • In-charge of the Final Report and ensuring that all the project requirements are met • Created all the records for data initialization covering the different business scenarios
Dorcas Tan	• Contributed to the preliminary ER diagram and listing of functional requirements and data constraints • Led the initial configuration (code structure, code quality) and integration of front-end and back-end • Developed the following application features: authentication, creation of user accounts, "My Pets" page, public profiles for Pets, SQL query for finding caretakers
Kelvin Wong	• Contributed to the preliminary ER diagram and listing of functional requirements and data constraints • Completed the preliminary wireframe and mock-up for front-end development https://www.figma.com/file/qttaFp0xmUFwjlqSEast17/PCS-v0.9?node-id=0%3A1 • Developed the following application features: Admin Dashboard, Profile Settings for Users and Admins, public profiles for Users and Admins, Performance Tab for Caretakers, SQL query for leaderboard and trigger to reject conflicting bids.
Wincent Tjoi	• Contributed to the preliminary ER diagram and listing of functional requirements and data constraints • Helped with setting up the front-end and back-end environment, and system set-up such as Git • Configuration for hosting in Heroku • Delegate tasks and organize weekly sprints • Developed the following application features: Leave and availability application, hosting on Heroku, triggers implementation.
Yang Jiyu	• Contributed to the preliminary ER diagram and listing of functional requirements and data constraints • Revised the preliminary ER diagram • Contributed to the SQL Create Table Queries • Developed the following application features: Browsing for caretakers, bookings for Pet Owners and Jobs for Caretakers.

Link to application: https://petcaringservices-group21.herokuapp.com/

2. Functional Requirements and Data Constraints

2.1 Application Functionalities

Pet Caring Services (PCS) is a three-tier web application that enables dog and cat pet owners to match with the right caretaker for their pets. PCS supplements our customers' busy lifestyles when they wish to temporarily free themselves of pet caring responsibilities. There are three broad categories of PCS users: Caretakers, Pet Owners and PCS Administrators. The application is scalable and supports the creation, deletion, and updating of relevant data.

Pet Owner

Upon the creation of their PCS account, Pet Owners will be able to write and edit their profile information such as a short biography, address, and email. Login is done via password and email or username. They also have the option of adding their credit card information for convenience[1]. Before selecting any services for their pet, they will first be invited to fill their Pet's information along with any requirements such as the pet's dietary restrictions. They must also select a Pet Category for their Pet because it will determine the daily price for the pet services. The 5 Pet Categories and their respective daily prices are: Cat (All) with a daily price of $25, Small Dog (0kg – 10kg) with a daily price of $25, Medium Dog (11kg – 20kg) with a daily price of $30, Large Dog (21kg – 40kg) with a daily price of $35 and Giant Dog (41kg and above) with a daily price of $40.

[1] Even though Pet Owners are given the option to pay either through cash or credit card in PCS, the application does not support actual transaction (payment) between users as this feature is out-of-scope of our project.

At the browsing page, Pet Owners can search for available[2] Caretakers by indicating the start and end date that they need their services, and which registered Pet it is for. Pet Owners can filter the Caretakers based on their average rating. They can also view the Caretaker's profile and past reviews to make an informed choice. After finding suitable caretakers, Pet Owners can make bids for one or more of the caretakers. To make a bid, a Pet Owner must provide additional information about the preferred mode of pet transfer (Delivery, Pick-up, On-site transfer). The bid will also calculate and reflect the invoice amount (daily price for the pet category multiplied by the number of days) which will be charged to the Pet Owners should the bid be successful. If any of the bids for a particular pet is accepted by a caretaker, any other bids for the same pet in the same time period will be automatically rejected by the system.
After a bid is accepted, Pet Owners must indicate what method of payment they have used (Cash or Credit Card). The Caretaker can then mark the job as "Completed". When a job has been completed, Pet Owners can also write a review and give a rating to that Caretaker. Reviews given by a Pet Owner should still be visible if he/she decides to delete their account or remove a pet.

Caretaker

Like Pet Owners, Caretakers can view and edit their profile information such as their biography, address, password, and email. Caretakers must indicate the Pet Categories that they can care for because this will affect the bidding process mentioned above. Caretaker can also view all the bids categorized by status (Pending, Accepted, Completed, Rejected)[3]

There are 2 sub-categories of caretakers, namely, Full Time Employees and Part Time Employees. Part Time Employees must indicate their availability dates for their names to appear in the browsing page. Otherwise, they will not be able to receive any bids. When they receive a bid, they have the option of accepting or rejecting the bid. On the other hand, when a Full Time Employee receives a bid, the bid is automatically accepted. Full Time Employees are deemed as available all year-round unless their leave application is approved by a PCS Administrator. When applying for leave, they must ensure that the following requirements are met[4]:

- They must be able to work for 2 sets of 150 hours for that particular year. The start date begins from the day their account is created.
- They are free of any responsibilities during the period they are applying for (no accepted bids).
- The new leave period does not overlap with an existing leave application (whether approved or not) for the same period.

All employees have a maximum workload: they can care for up to 5 pets at any point of time. Part-time caretakers can care for a maximum of 2 pets unless they have an outstanding performance rating of more than 4 stars (out of 5). All Caretakers are also able to view the leaderboard for the previous month, which shows the top performing caretakers. The leaderboard ranks the top 5 full-time employees and top 5 part-time employees based on their performance, which consists of customer satisfaction, efficiency and popularity[5]. Finally, Caretakers are also able to view a summary of their work for the previous month: Total number of bids received, Total number of jobs (bids with "Completed" status), Total number of pet days worked, Base Salary, Variable Salary, Total Salary, Average Rating and Performance.

PCS Administrator

PCS Administrator accounts can only be created by any other PCS Admin for security purposes. A PCS Admin can also approve the leave applications submitted by the Full Time Employee. PCS Administrators can also view the caretaker leaderboard for the previous month. Most importantly, Admins can see individual caretakers' summary information and the amount of profit (or loss) that each caretaker has contributed to the company[6].

2.2 Data Constraints

A. Accounts

A.1. There are two types of Accounts: User accounts and PCS Administrator accounts. An account must be either a user or an admin account (covering constraint), but not both (no overlap constraint). There are two types of Users: Pet Owner and Caretakers. A user must be either a Pet Owner or a Caretaker (covering constraint), or both (overlap constraint).
A.2. An Account is identified by its username. Each account must have a password digest, email, the person's name and the date of account registration.

[2] Refer to Section 6.2 for more information on how the bids constraint affects the Caretaker's availability.
[3] Refer to Section 6.3 for more information on the trigger that rejects conflicting bids
[4] Refer to Section 6.1 for more information on the trigger for leave application
[5] Refer to Section 7.2 for more information on the complex query that computes the caretaker's leaderboard of the month
[6] Refer to Section 7.3 for more information on the complex query for the admin's dashboard

A.3. A User account stores the user's profile which comprises of a bio, phone number, address, postal code and the timestamp when the account was deleted. The timestamp of deletion is recorded when the user account is deleted (user account is not deleted from the database to allow previous reviews to be seen). Once an account is deleted, the same email cannot be used to register for a new account. On the other hand, PCS Administrator accounts use hard deletion (removed from the database).

B. Pet Owners, Pets and Pet Requirements

B.1. Each Pet Owner can store information for at most one credit card. Credit card information includes the credit card number, cardholder name, credit card expiry date and CVV code.
B.2. A Pet is identified by its name and owner. Each Pet has a year of birth, gender and breed and time of deletion.
B.3. A Pet Owner can own zero or more Pets. A Pet must be owned by exactly one Pet Owner. If the Pet Owner is deleted, his/her associated Pet needs to be deleted too (soft deletion; the pet is not actually deleted in the database). We restrict that once a pet is deleted, the pet owner cannot create another pet with the same pet name as the deleted one.
B.4. A Pet can have many requirements, such as its dietary requirements and regular walk timings. Each Pet Requirement must belong to exactly one Pet.
B.5. A Pet Requirement is identified by the pet it belongs to and a requirement type. It also has a description. The requirement type can be selected from a predefined list or defined by the user.

C. Pet Categories

C.1. Each Pet belongs to exactly one category. Each Pet Category can have zero or more Pets and is identified by its category name.
C.2. Each Pet Category have a daily price associated with it (listed above in Section 2.1).

D. Caretakers: Full Time Employees and Part Time Employees

D.1. There are two types of Caretakers: Part Time Employees and Full Time Employees. A Caretaker must be either a part-time or full-time employee (covering constraint), but not both (no overlap constraint).
D.2. A caretaker can care for zero or more pet categories.
D.3. A Part Time Employee can indicate one or more Availability Periods. Each Availability Period is identified by its start date, end date, and the associated employee.
D.4. A Full Time Employee can apply for one or more Leave Periods. Each Leave Period is identified by its start date, end date, approval status, and the associated employee. A Leave Period also indicates whether it is an emergency leave or not. A Leave Period should be deleted if its associated employee is deleted.
D.5. A Leave Period can only be added if there will be at least two 150-day gaps between all Leave Periods (including the one to be added) from the account registration date to a year after - in the relevant year(s).
D.6. The start and end dates of different Leave Periods of an employee cannot overlap.
D.7. To apply for leave, employee cannot have accepted bids (taking care of a pet) in any day during the leave period.
D.8. A Full Time Employee cannot have more than 5 jobs at the same time. A Part Time Employee cannot have more than 2 jobs unless they have an average rating of at least 4 out of 5 (then they can have at most 5 simultaneous jobs).

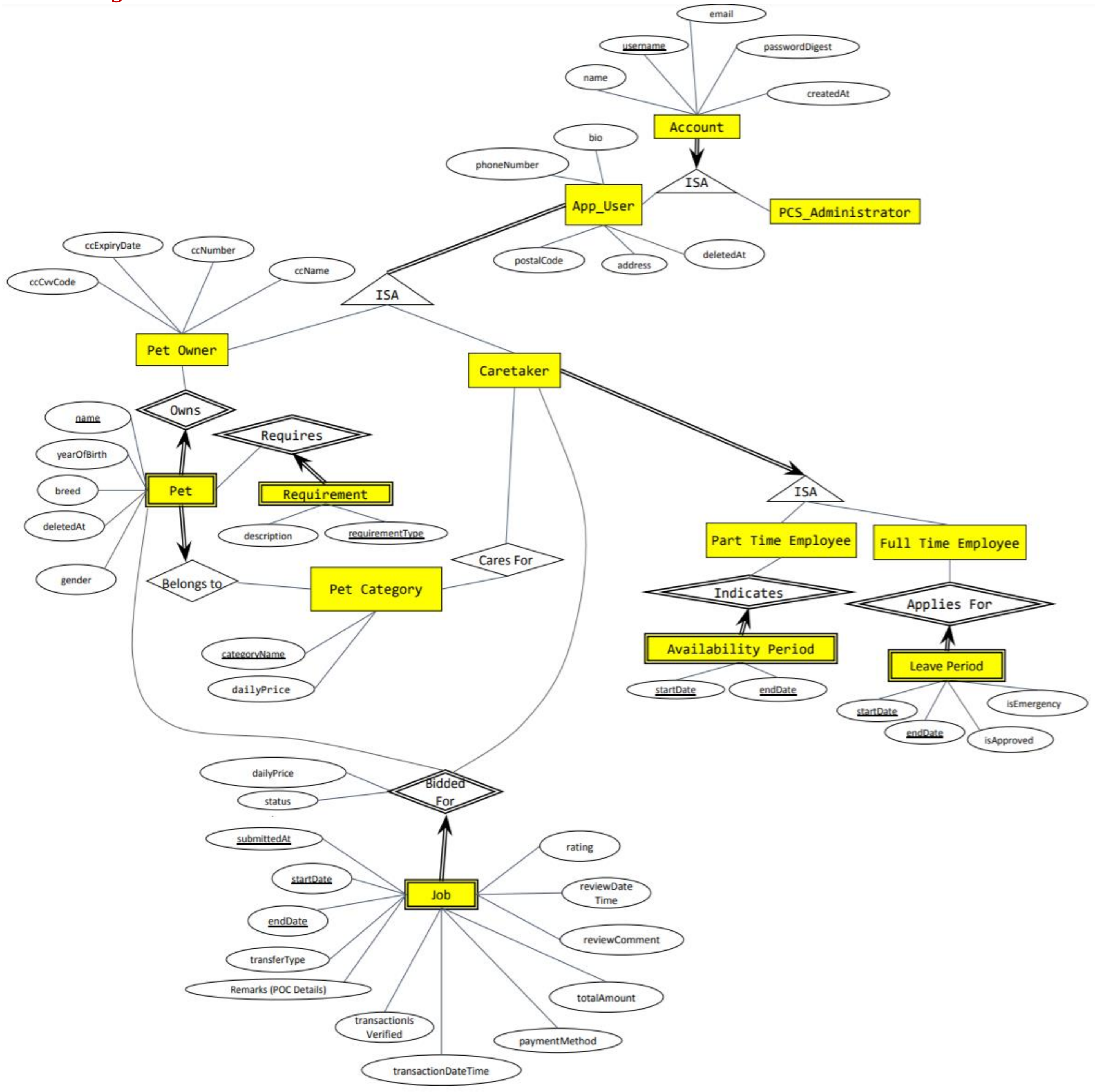
E. Bidding

E.1. When the Pet Owner bids for an available Caretaker, the Pet Owner specifies the selected Pet that he wants the Caretaker to take care for. The Pet Owner also must indicate the start and end date. The Pet Owner can only bid for caretakers who have not reached their maximum job quota (as explained in D.8. above) and are available during that period (as explained in Section 2.1- Caretaker). Each Bid must also include the daily price, a status and a submission date and time. The five possible bid statuses are: Pending, Accepted, Completed, Expired and Rejected. This is further elaborated in Section 7.1- Filtering for available Caretakers.
E.2. Each accepted Bid is identified by the Caretaker who undertakes it, the Pet involved in it, the time of bidding and its start and end date.
E.3. A Bid can only be added if its start date and end date are within one of the Caretaker's availability periods (for part-time employees), or if its start date and end date do not cross into any of the Caretaker's leave periods (for full-time employees).
E.4. A Part-Time Caretaker can select or reject the bids that he wants to undertake by changing the bid status to Approved or Rejected respectively.
E.5. A Pet Owner can choose to cancel a Pending bid.
E.6. A Bid can only contain the time of transaction and payment method (cash or credit card) if its status is Accepted or Completed.

E.7. A Bid can only have a rating, comment, and time of review if its status is Completed and a transaction time is present.
E.8. A Bid must include a daily price[7] and transfer type. Transfer type can be delivery, pick-up or on-site transfer.
E.9. Rating must be an integer between 1 and 5 inclusive.
E.10. When a Pending bid is accepted by a Caretaker, all other Pending bids with the same pet name and pet owner that fall within the period of the latest accepted bid, are rejected.

3. Entity Relational Model

3.1. ER Diagram



[7] This daily price may differ from the daily price of the associated Pet Category if a future PCS Administrator decides to change the Pet Category's daily price.

3.2. Non-trivial design decisions

(a) ISA relationships

- An account can either be a User or PCS Admin (not both).
- A user can be a Pet Owner, Caretaker, or both.
- A caretaker can either be part-time or full-time employee (not both).

(b) Weak entities

- A Pet has an identity dependency on their Pet Owner and are existentially dependent on their Pet Category.
- A Job has an identity dependency on the Pet and Caretaker involved in it.
- Availability period has an identity dependency on Part Time Employee.
- Leave period has an identity dependency on Full Time Employee.
- A Requirement has an identity dependency on the Pet it is tagged to.

3.3. Uncaptured constraints

The following constraints are not captured in the ER Diagram:

Constraint	Description	Where is the constraint captured?
Unique Constraint	• Every **PCS Admin**, **Caretaker** and **Pet Owner** must have a unique email	SQL Create Table
Not Null Constraint[8]	• Every **PCS Admin**, **Caretaker** and **Pet Owner** must have an email, name, and password. • Every **Caretaker** and **Pet Owner** must have a phone number, address and postal code. • Every **Pet** must have a year of birth, breed, and gender. • Every **Requirement** must have a description. • Every **Pet Category** must have a daily price. • Every **Leave Application** must indicate if it is an emergency. • Every **Bid** must have a daily price and transfer type.	SQL Create Table
Check Constraint	• When a caretaker is indicating the **Availability Period** or **applying for Leave**, the start date must be earlier than the end date specified. Furthermore, the latest end date for an **availability period** is 31 December of the following year. • When a pet owner is giving a feedback rating to the pet owner, the **rating** must be an integer between 1 and 5 inclusive. • A **Bid** with Pending/Rejected/Expired status must have an empty transaction date time and payment method. For the **Bid's** rating, comment and review date time to be filled, the status of Bid must be Completed, and transaction date time is not empty.	SQL Create Table
Covering constraints	• All **Accounts** must either be an **App User or PCS Administrator**. (Covering) • All **App Users** must either be a **Pet Owner or Caretaker.** (Covering) • All Caretakers must either be a Part Time or Full Time Employee. (Covering)	Triggers
Overlapping constraints	• An **App User** cannot be a **PCS Administrator**, and vice versa. (Non-overlapping) • A **Pet Owner** can also be a **Caretaker,** and vice versa. (No restriction) • A Part-Time employee cannot be a Full-Time employee, and vice versa. (Non-overlapping)	Triggers

[8] Excluding attributes in the primary key, which must be non-null by default.

Insertion constraints	To insert a new **leave application**, the caretaker must not be taking care of any pets during the leave duration, and caretaker must be able to fulfill 2 sets of 150 consecutive working days after the application. • To insert a new **bid**, a full-time caretaker cannot have more than 5 jobs for that bid's duration. For part-time caretaker, he or she can have up to 2 jobs, unless if rating is more than 4 — then the employee can also have up to 5 jobs.	Triggers

Other constraints that are not captured in the ER Diagram are: **<u>Section 2.2 - B.5, D.5, D.6, D.8, E.3, E.4 and E.5</u>**.

4. Relational schema

4.1. SQL CREATE TABLE statements

```
CREATE TABLE PCS_Administrator(
  username VARCHAR PRIMARY KEY,
  email VARCHAR NOT NULL UNIQUE,
  passwordDigest VARCHAR NOT NULL,
  name VARCHAR NOT NULL,
  deletedAt TIMESTAMP,
  createdAt DATE NOT NULL
);

CREATE TABLE App_User(
  username VARCHAR PRIMARY KEY,
  email VARCHAR NOT NULL UNIQUE,
  passwordDigest VARCHAR NOT NULL,
  name VARCHAR NOT NULL,
  deletedAt TIMESTAMP,
  bio VARCHAR,
  phoneNumber VARCHAR NOT NULL,
  address VARCHAR NOT NULL,
  postalCode VARCHAR NOT NULL,
  createdAt DATE NOT NULL
);

CREATE TABLE Pet_Owner(
  ccNumber VARCHAR,
  ccName VARCHAR,
  ccExpiryDate DATE,
  ccCvvCode VARCHAR,
  petOwnerUsername VARCHAR PRIMARY KEY
    REFERENCES App_User(username)
    ON DELETE CASCADE
);

CREATE TABLE Caretaker(
  caretakerUsername VARCHAR PRIMARY KEY
    REFERENCES App_User(username)
    ON DELETE CASCADE
);

CREATE TABLE Full_Time_Employee(
```

```
CREATE TABLE Indicates_Availability_Period(
  caretakerUsername VARCHAR REFERENCES
    Part_Time_Employee(caretakerUsername)
    ON DELETE CASCADE,
  startDate DATE,
  endDate DATE,
  CHECK (startDate <= endDate),
  -- Latest availability application is end of next
year
  CHECK (extract(year from endDate) <=
    extract(year from current_date) + 1),
  PRIMARY KEY(caretakerUsername, startDate,
    endDate)
);

CREATE TABLE Applies_For_Leave_Period(
  caretakerUsername VARCHAR REFERENCES
    Full_Time_Employee(caretakerUsername)
    ON DELETE CASCADE,
  startDate DATE,
  endDate DATE,
  isEmergency[9] BOOLEAN NOT NULL,
  isApproved BOOLEAN DEFAULT FALSE,
  CHECK (startDate <= endDate),
  PRIMARY KEY(caretakerUsername,
    startDate, endDate)
);

CREATE TYPE STATUS AS ENUM (
  'Accepted',
  'Completed',
  'Pending',
  'Expired',
  'Rejected'
);

CREATE TYPE PAYMENT_METHOD AS ENUM (
  'Cash',
  'Credit Card'
```

[9] Due to time constraints, the logic for isEmergency is not implemented for this project.

```
  caretakerUsername VARCHAR PRIMARY KEY
    REFERENCES
Caretaker(caretakerUsername)
    ON DELETE CASCADE
);

CREATE TABLE Part_Time_Employee(
  caretakerUsername VARCHAR PRIMARY KEY
    REFERENCES
Caretaker(caretakerUsername)
    ON DELETE CASCADE
);

CREATE TYPE GENDER AS ENUM (
  'Male',
  'Female'
);

CREATE TABLE Pet_Category(
  categoryName VARCHAR PRIMARY KEY,
  dailyPrice DECIMAL(10,2) NOT NULL
);

CREATE TABLE Pet(
  petOwnerUsername VARCHAR REFERENCES
    Pet_Owner(petOwnerUsername)
    ON DELETE CASCADE,
  name VARCHAR,
  yearOfBirth DATE NOT NULL,
  breed VARCHAR NOT NULL,
  deletedAt TIMESTAMP,
  gender GENDER NOT NULL,
  categoryName VARCHAR NOT NULL
    REFERENCES Pet_Category(categoryName),
    PRIMARY KEY(petOwnerUsername, name)
);

CREATE TABLE Requirement(
  requirementType VARCHAR,
  description VARCHAR NOT NULL,
  petName VARCHAR,
  petOwnerUsername VARCHAR,
  PRIMARY KEY(requirementType, petName,
    petOwnerUsername),
  FOREIGN KEY(petName,petOwnerUsername)
    REFERENCES Pet(name,
      petOwnerUsername)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
);

CREATE TYPE TRANSFER_TYPE AS ENUM (
  'Delivery',
  'Pick-up',
  'On-site transfer'
);

CREATE TABLE Bids(
  petName VARCHAR,
  petOwnerUsername VARCHAR,
  caretakerUsername VARCHAR REFERENCES
    Caretaker(caretakerUsername) ON DELETE CASCADE,
  dailyPrice DECIMAL(10,2) NOT NULL,
  status STATUS,
  submittedAt TIMESTAMP,
  startDate DATE,
  endDate DATE,
  transferType TRANSFER_TYPE NOT NULL,
  remarks VARCHAR,
  transactionDateTime TIMESTAMP,
  paymentMethod PAYMENT_METHOD,
  totalAmount DECIMAL(10,2),
  rating INTEGER CHECK (rating BETWEEN 1 AND 5),
  comment VARCHAR,
  reviewDateTime TIMESTAMP,
  CONSTRAINT statusAccepted CHECK(status IN
    ('Accepted', 'Completed') OR
    (transactionDateTime IS NULL AND
    paymentMethod IS NULL)),
  CONSTRAINT transactionCompleted
    CHECK((transactionDateTime IS NOT NULL AND
      status = 'Completed') OR
    (rating IS NULL AND comment IS NULL AND
       reviewDateTime IS NULL)),
  PRIMARY KEY(petName, petOwnerUsername,
    caretakerUsername, submittedAt, startDate,
    endDate),
  FOREIGN KEY(petName, petOwnerUsername) REFERENCES
    Pet(name, petOwnerUsername) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE Cares_For(
  categoryName VARCHAR REFERENCES
    Pet_Category(categoryName),
  caretakerUsername VARCHAR REFERENCES
    Caretaker(caretakerUsername),
  PRIMARY KEY(categoryName, caretakerUsername)
);
```

4.2. Unenforced Constraints

All the constraints that are enforced by triggers instead of relational schema are listed in Section 3.3 above.

In addition, we wanted to automatically reject Pending bids with start times that have already passed. However, this could not be enforced with relational schema nor triggers and requires the use of Heroku scheduler (not implemented due to time constraints).

5. Database Normalisation

The database is in neither BCNF nor 3NF. Below are the highest normal form for each of the tables in the database.

Table	**Highest Normal Form (up to BCNF) & Explanations**
PCS_Adminstrator	BCNF. All attributes are directly dependent on all keys: { username }, { email }.
App_User	BCNF. Similar to PCS_Administrator.
Pet_Owner	BCNF. PetOwnerUsername is the only candidate key. All the attributes are directly dependent on PetOwnerUsername.
Caretaker	BCNF. There is no non-trivial FD.
Full_Time_Employee	BCNF. There is no non-trivial FD.
Part_Time_Employee	BCNF. There is no non-trivial FD.
Pet	BCNF. All attributes are directly dependent on the key {PetOwnerUsername, PetName}.
Pet_Category	BCNF. All attributes are directly dependent on the key {categoryName}.
Requirement	BCNF. All attributes are directly dependent on the key {requirementType, petName, petOwnerUsername}.
Cares_For	BCNF. All attributes are directly dependent on the key {categoryName, caretakerUsername}.
Indicates_Availability_Period	BCNF. All attributes are directly dependent on the key {caretakerUsername, startDate, endDate}.
Applies_For_Leave_Period	BCNF. All attributes are directly dependent on the key {caretakerUsername, startDate, endDate}.
Bids	Neither 3NF nor BCNF. There is a partial dependency in the FD = { Daily Price, Start Date, End Date => Total Amount } The LHS of F is not a superkey and RHS of F (Total Amount) is not a prime attribute, so it is not in 3NF (or BCNF). Nonetheless, we chose to store the total amount for easy retrieval of information. Realistically, these 4 attributes (Daily Price, Start Date, End Date, Total Amount) do not change after the bid is created (since Pet Owners and Caretakers are not allowed to edit these attributes). Therefore, there will not be any update anomaly, which 3NF aims to prevent.

6. Interesting SQL Triggers

6.1. Interesting Trigger 1: Applying for a New Leave for Full Time Employees

Business Case:
For a new leave application to be inserted, **ALL** the following conditions must hold true:

- The Full Time Employee must **not** have an existing leave application for any day in the leave application (no overlapping dates).
- Full Time Employee must **not** be taking care of any pet during all the days in the leave application. An employee is taking care of a pet only if the bid associated to it on that date has been accepted; pending bids does not count.

- The Full Time Employee must still be able to complete 2 sets of 150 consecutive working days if the leave is applied for and approved by the Admin.

In the 3rd criteria mentioned above, every employee will have a different starting date depending on when they created their PCS account. We assume that if a user registers as Full Time Employee on 5th January 2020, then that employee needs to fulfill 2 sets of 150 consecutive working days starting from 5th January 2020 to 5th January 2021. Once this iteration is done, the next set will be from 5th January 2021 to 5th January 2022.

```
CREATE OR REPLACE FUNCTION leave_constraints() RETURNS TRIGGER AS $$
DECLARE
  overlapping_date INTEGER;
  pet_under_care INTEGER;
  total_set INTEGER;
BEGIN
  -- Check no overlapping date of leave
  SELECT COUNT(*) INTO overlapping_date FROM (
  SELECT (generate_series(NEW.startdate, NEW.enddate, '1 day'::interval))::date
  INTERSECT
  SELECT generate_series(startdate, enddate, '1 day')
  FROM applies_for_leave_period where caretakerusername = NEW.caretakerusername)
      AS intersect_dates;

  IF overlapping_date > 0 THEN
    RAISE EXCEPTION 'You cannot apply for overlapping leaves';
  END                                                                                IF;

  -- Check no pet under care
  SELECT COUNT(*) INTO pet_under_care FROM (
  SELECT (generate_series(NEW.startdate, NEW.enddate, '1 day'::interval))::date
  INTERSECT
  SELECT generate_series(startdate, enddate, '1 day')
  FROM Bids WHERE caretakerusername = NEW.caretakerusername AND status = 'Accepted')
      AS pet_being_cared;

  IF pet_under_care > 0 THEN
    RAISE EXCEPTION 'You cannot apply leave you are / will be in charge of one or more pets during
the leave';
  END IF;

  -- Check caretaker can still meet 2x150 working days criteria starting from when account was
registered
  -- Generate "consecutive_leave_groups" of dates by subtracting the date's row number (no gaps)
from the date itself (with potential gaps). Whenever there is a gap, there will be a new group
  WITH dates(date) AS (
    -- This table contains all the distinct date instances in the data set
    SELECT (generate_series(
      concat(extract(year from current_date)::text, substring(createdAt::text from 5))::date,
      concat((extract(year from current_date) + 1)::text,
        substring(createdAt::text from 5))::date,
      '1 day'::interval))::date
    FROM App_User WHERE username = NEW.caretakerusername
    EXCEPT
    -- EXCEPT the insert of new row here
    SELECT(generate_series(NEW.startdate, NEW.enddate, '1 day'::interval))::date
    EXCEPT
    -- DATA that are already existing in the table beforehand
    SELECT generate_series(startdate, enddate, '1 day')
    FROM applies_for_leave_period
    WHERE caretakerusername = NEW.caretakerusername
  )

  SELECT sum(sets) INTO total_set FROM (
    SELECT COUNT(*) / 150 AS sets
    FROM (
      SELECT
        ROW_NUMBER() OVER (ORDER BY date) AS rn,
        date + (-ROW_NUMBER() OVER (ORDER BY date)) * INTERVAL '1 day' AS grp,
        date
      FROM dates
    ) AS groups
    GROUP BY grp
  ) AS consecutive;

  IF (total_set <> 2) THEN
    RAISE exception 'You cannot take this leave because you will not be able to fulfill minimum
requirements of consecutive working days';
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_leave_constraints
BEFORE INSERT ON applies_for_leave_period
FOR EACH ROW EXECUTE PROCEDURE leave_constraints();
```

6.2. Interesting Trigger 2: Bids Constraint

Business Case:

- To ensure that no employee is overwhelmed by too many pet requests, we only allow full-time employees to receive 5 bids for any day. For Part time employee, the limit is 2. A special consideration is given to Part time employees with excellent performance — if their average rating is more than 4 (out of 5), we allow him/her to hold up to 5 pets at any time.
- A full-time employee does not have the privilege to accept or reject bids. Bids will be automatically accepted by the system if they have a new bid. This is to ensure that no pet owner is having a hard time finding a caretaker. A part time Caretaker however, has the flexibility to choose if they want to accept the bid.
- This ensures for someone to register as a full-time employee and enjoy its perks on the potential higher salary, one must first be ready to commit their time fully.
- Furthermore, a pet owner will not be able to bid for a part-time caretaker (good rating) that has 4 Accepted and 1 Pending bids. This is to avoid the part-time caretaker from being overwhelmed with too many requests, hence a new bid can only come in after he/ she rejects the last pending bid.

```
CREATE OR REPLACE FUNCTION bids_constraint()
```

```
RETURNS TRIGGER AS $$
DECLARE max_jobs INTEGER;
BEGIN
  SELECT MAX(n) INTO max_jobs
  FROM (
    SELECT caretakerusername, as_of_date, COUNT(*) AS n
    FROM (
      SELECT d::date AS as_of_date
      FROM generate_series(NEW.startdate::date, NEW.enddate::date, '1 day') d
  ) as dates
  INNER JOIN Bids ON dates.as_of_date BETWEEN Bids.startdate AND Bids.enddate
  WHERE Bids.status = 'Accepted' OR Bids.status = ‘Pending’
  GROUP BY caretakerusername, as_of_date) T WHERE NEW.caretakerusername = T.caretakerusername;

  -- All caretakers can have a maximum of 5 pets at one time
  IF max_jobs >= 5 THEN
    RAISE EXCEPTION 'Caretaker is unable to receive more pets during this period';
  -- Part timer can hold up to 2 or 5 pets depending on rating
  ELSEIF (
    NEW.caretakerusername IN (SELECT P.caretakerusername FROM Part_Time_Employee P)
    AND ((
      SELECT totalAverageRating FROM Caretaker WHERE Caretakerusername = NEW.caretakerusername)
<= 4)
    AND max_jobs >= 2
  ) THEN
    RAISE EXCEPTION 'Part time Caretaker is unable to receive more pets during this period';
  END                                                                                    IF;

  IF (NEW.caretakerusername IN (SELECT F.caretakerusername FROM Full_Time_Employee F)) THEN
    NEW.status = 'Accepted';
  ELSEIF (NEW.caretakerusername IN (SELECT caretakerusername FROM Part_Time_Employee)) THEN
    NEW.status = 'Pending';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_bids_constraint
BEFORE INSERT ON Bids
FOR EACH ROW EXECUTE PROCEDURE bids_constraint();
```

6.3. Interesting Trigger 3: Reject Conflicting Bids

Business Case:

- Since Pet Owners can submit multiple bids for a service, there could be a scenario whereby there is more than one accepted bid for the same job, e.g., two caretakers taking care of the same pet on the same day. To prevent this from happening, PCS will update the other pending bids for the same pet and pet owner over the same period, should any one of them be accepted.
- The constraint below ensures that duplicate bids are rejected once it is impossible for them to be accepted. Hence, it reduces clutter for both parties involved (Pet Owner and Caretakers).

```
CREATE OR REPLACE FUNCTION reject_conflicting_bids()
```

```
RETURNS TRIGGER AS $$
BEGIN
    UPDATE bids SET status = 'Rejected'
    WHERE bids.status = 'Pending' AND bids.caretakerusername != NEW.caretakerusername AND bids.p
etownerusername = NEW.petownerusername AND bids.petname = NEW.petname AND
        (bids.startdate BETWEEN NEW.startdate AND NEW.enddate OR bids.enddate BETWEEN NEW.startd
ate AND NEW.enddate);
        RETURN NULL;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_conflict_bids_constraint
AFTER UPDATE ON Bids
FOR EACH ROW
WHEN  (OLD.status = 'Pending' AND NEW.status = 'Accepted')
EXECUTE PROCEDURE reject_conflicting_bids();
```

7. Complex SQL Queries

7.1. Complex Query 1: Filtering for Available Caretakers

Business Case:

- Pet owners can search for caretakers with the following filters:
    a. Have a minimum average rating
    b. Can care for a specific pet category
    c. Are available (not on leave) on all the days between the start date and end date (inclusive)
- Caretakers who cannot care for more pets on any of the given days are not returned. All full-time employees and part-time employees with a total average rating of at least 4 can care for a maximum of 5 pets per day, while all other employees can care for a maximum of 2 pets per day.
- The results are paginated and only 10 caretakers are displayed at a time.
- **Refer to Section 10.2 for screenshot**

```
SELECT username, average_rating(username) AS totalAverageRating, name, bio
FROM Caretaker CT INNER JOIN App_User ON App_User.username = CT.caretakerUsername
WHERE average_rating(CT.caretakerUsername) >= $1
AND EXISTS (
    SELECT  1  FROM  Cares_For  CF  WHERE  CF.caretakerUsername  =  CT.caretakerUsername  AND
CF.categoryName = $2
) AND NOT EXISTS (
    -- No approved Leave Applications
    SELECT 1 FROM Applies_For_Leave_Period L
    WHERE L.caretakerUsername = CT.caretakerUsername
        AND L.isApproved -- make them continue to show up until leave approved
        AND ((L.startDate < $3 AND L.endDate >= $3) OR (L.startDate >= $3 AND L.startDate <= $4))
) AND (
    -- Either Full-Time Caretaker or Part-Timer who is available from startDate to endDate
    CT.caretakerUsername IN (SELECT caretakerUsername FROM Full_Time_Employee)
    OR EXISTS (
        SELECT 1 FROM Indicates_Availability_Period A
        WHERE A.caretakerUsername = CT.caretakerUsername AND A.startDate < $3 AND A.endDate > $4
    )
) AND (
    -- Max number of pets allowed at any point
    CASE
    WHEN   EXISTS   (SELECT   1   FROM   Full_Time_Employee   FT   WHERE   FT.caretakerUsername   =
CT.caretakerUsername)
        OR average_rating(CT.caretakerUsername) >= 4
    THEN 5
    ELSE 2
    END
) >= ALL (
    -- number of accepted bids on every day in the given time period
    SELECT COUNT(*)
    FROM (
        SELECT day FROM generate_series($3, $4, '1 day'::interval) day
    ) dates
    INNER JOIN Bids ON dates.day BETWEEN Bids.startDate AND Bids.endDate
    WHERE Bids.caretakerUsername = CT.caretakerUsername
    GROUP BY caretakerUsername, day -- Note: caretakerUsername kept here for consistency
)
ORDER BY totalAverageRating DESC -- NULLS first to give chance to newer caretakers
LIMIT 10
OFFSET $5
[minRating, petCategory, startDate, endDate, offset]
```

7.2. Complex Query 2: Leaderboard of the Month

Business Case: We wanted to make a Caretaker Leaderboard to be made available to both Caretakers and PCS Admin. This would highlight the top 5 Caretakers of each role (Full-Time / Part-Time) so that they could be awarded or recognized. They will be ranked based on the point system below:

1) Customer Satisfaction: Average rating of reviews received by the Caretaker from jobs in the previous month. Their average rating (maximum of 5) will be multiplied by 10 to derive the customer satisfaction points.
2) Efficiency: The total number of jobs (completed bids) the caretaker took last month is computed relative to the average number of jobs completed by a caretaker during the same period. This ratio will be multiplied by 25 to compute the Efficiency points.
3) Popularity: Relative number of bids received by the Caretaker last month compared to the average of peers in the same role. This ratio is multiplied by 25 to compute the Popularity points.

Should there be a tie, they will share the same ranking. Therefore, it is possible to have more than 5 employees of the same role in the leaderboards if there is a tie. For example, the scores of the top 5 ranked Part-time Employees could be: 100, 99, 98, 97, 90, 90, 90, 90, 90 and 90.

Refer to Section 10.3 for screenshot

Considerations:

- Due to the restrictions on CTE, we initially used lateral joins. While it worked and managed to reduce the length of this query, the code was less clear and some shadowing was used, which we suppose isn't encouraged.

```
-- View for Admin Dashboard
CREATE OR REPLACE VIEW leaderboard AS (
    -- After summing the scores, rank them against their peers of the same role, and
    -- only select the caretakers with top 5 scores of each role.
  SELECT * FROM
    (SELECT *, rank() OVER (PARTITION BY role ORDER BY totalScore desc) AS rank
    FROM (
        SELECT *, satisfactionscore + efficiencyscore + popularityscore AS totalScore
        FROM (
            -- Calculate the scores using the score equation after changing nulls to zeroes
            SELECT *, COALESCE(ROUND(averageRating * 10),0) AS satisfactionscore,
                  COALESCE(ROUND(jobCount * 25 / avgJobCount),0) AS efficiencyscore,
                  COALESCE(ROUND(bidCount * 25 / avgBidCount),0) AS popularityscore
            FROM (
              SELECT *, AVG(jobCount) over (partition by role) AS avgJobCount, AVG(bidCount) over
(partition by role) AS avgBidCount FROM
              (
                SELECT  COALESCE(caretakerusername,  caretakerusername2)  AS  caretakerusername,
COALESCE(role, role2) AS role, jobcount, averageRating, bidcount
                FROM (
                    -- Find all caretakers that have completed a job in the past month, with their
roles, average rating and job count
                    (SELECT caretakerusername,
                        CASE WHEN caretakerusername IN (SELECT * FROM part_time_employee ) THEN
'PT'
                            WHEN caretakerusername IN (SELECT * FROM full_time_employee ) THEN
'FT'
                        ELSE 'null'
                        END AS role,
                    COUNT(*) AS jobCount, AVG(rating) AS averageRating
                    FROM bids
                    WHERE status = 'Completed' AND
                      enddate BETWEEN date_trunc('month', CURRENT_DATE - interval '1' month) AND
date_trunc('month', CURRENT_DATE)
                    GROUP BY caretakerusername) t1
                    FULL OUTER JOIN
                    -- Find all caretakers that have received a bid in the past month, with their
roles, and bid count
                    (SELECT caretakerusername AS caretakerusername2,
                        CASE WHEN caretakerusername IN (SELECT * FROM part_time_employee ) THEN
'PT'
                        WHEN caretakerusername IN (SELECT * FROM full_time_employee ) THEN 'FT'
                        ELSE 'null'
                        END AS role2,
                        count(*) as bidCount
                    FROM bids
                    WHERE
                      submittedat BETWEEN date_trunc('month', CURRENT_DATE - interval '1' month)
AND date_trunc('month', CURRENT_DATE)
                    GROUP BY caretakerusername) t2
                    ON t1.caretakerusername = t2.caretakerusername2
                ) AS t3
              ) AS t4
            ) AS t5
        ) AS t6
    ) AS t7
  ) AS t8
  WHERE rank BETWEEN 1 AND 5
  ORDER BY role, rank, totalScore, efficiencyscore, popularityscore, satisfactionscore
);
```

7.3. Complex Query 3 – PCS Admin Summary Table

Business Case:

- The PCS Administrator is interested in seeing all the caretakers' performances and their individual contributions to the company (profits and losses) in the previous month.
- For each record of the caretaker, the PCS Admin Summary table will display the following information:
  - **Number of Pet Day, Total Number of Completed Bids, Total Number of Bids**
  - **Invoice Amount**: Total Monthly Revenue earned, or the Total amount paid by the pet owners to PCS. Total Monthly Revenue earned by each Caretaker. It is obtained by multiplying the number of days of service and the daily price.
  - **Base Salary, Variable Salary, Variable Percentage, Total Salary:** Only Full Time Employees have $2,500 base monthly salary. - For Part Timers, the variable percentage is 80% if the total Monthly Invoice Amount exceeds $500. Else, the variable percentage is lower at 60%. For Full Timers, if their Invoice Amount exceeds $2,500, they will be entitled to 50% of the share for the excess portions. Total Salary is the sum of the Base Salary and Variable Salary.
  - **Profit Margin**: - The monthly revenue (Total Invoice) minus the total monthly salary paid to each employee (Total Salary)
  - **Performance**: "Exceeds Expectations" if the monthly Profit Margin exceeds $500. "Meets Expectations" for Profit Margin between $0 and $500. "Below Expectations" if $0 and below.
- **Refer to Section 10.3 for screenshot**

```
CREATE OR REPLACE VIEW admin_summary AS
  SELECT *,
    CASE -- Performance of the caretakers based on their profit margin level
      WHEN profitMargin >= 500 THEN 'Exceeds Expectations'
      WHEN (profitMargin > 0 AND profitMargin < 500) THEN 'Meets Expectations'
      WHEN profitMargin =< 0 THEN 'Below Expectations'
      END AS performance
    FROM
    (SELECT *, -- Compute profit margin
      invoiceAmount - t3.totalSalary AS profitMargin
      FROM
      (SELECT *, -- Compute total salary
        t2.baseSalary + t2.variableSalary AS totalSalary
        FROM
        (SELECT *, -- Compute the base and variable salary based on their roles and invoice
amount
          CASE
            WHEN t1.role = 'PT' THEN 0.0
            WHEN t1.role = 'FT' THEN 2500
            ELSE 0.0
          END AS baseSalary,
          CASE
            WHEN t1.role = 'PT' AND t1.invoiceAmount > 500 THEN t1.invoiceAmount * 0.8
            WHEN t1.role = 'PT' AND t1.invoiceAmount < 500 THEN t1.invoiceAmount * 0.6
```

```
              WHEN t1.role = 'FT' AND t1.invoiceAmount > 2500 THEN (t1.invoiceAmount - 2500) * 0.5
              ELSE 0.0
            END AS variableSalary,
            CASE
              WHEN t1.role = 'PT' AND t1.invoiceAmount > 500 THEN '80%'
              WHEN t1.role = 'PT' AND (t1.invoiceAmount < 500) and (t1.invoiceAmount > 0) THEN '60
%'
              WHEN t1.role = 'FT' AND t1.invoiceAmount > 2500 THEN '50% of excess $2,500'
              ELSE '0%'
            END AS variablePercentage
          FROM (
-- For all the caretakers, find their username, role (PT or FT), the total number of bids last
month, job count last month , average rating last month, pet day count last month, total invoice
amount (for completed bids only)
                    SELECT c4.caretakerusername,
                    CASE
                        WHEN c4.caretakerusername IN (SELECT * FROM part_time_employee) THEN 'PT
'
                        WHEN c4.caretakerusername IN (SELECT * FROM full_time_employee) THEN 'FT
'
                        ELSE 'null'
                    END AS role,
                    COALESCE((SELECT COUNT(*)
                        from bids b
                        WHERE b.caretakerusername = c4.caretakerusername
                        AND submittedAt BETWEEN date_trunc('month', CURRENT_DATE - interval '1'
month) AND date_trunc('month', CURRENT_DATE)
                        GROUP BY b.caretakerusername), 0)
                    AS bidCount,
                    COALESCE(nullableJobCount,0) AS jobCount,
                    COALESCE(nullableAverageRating,0) as averageRating,
                    COALESCE(nullablePetDayCount,0) AS petDayCount,
                    COALESCE(nullableInvoiceAmount,0) as invoiceAmount
                    FROM (caretaker c
                        LEFT JOIN
                        (SELECT b2.caretakerusername AS caretakerusername2, COUNT(*) as nullable
JobCount,
                            AVG(rating) AS nullableAverageRating,
                            SUM(endDate - startDate + 1) AS nullablePetDayCount,
                            SUM(totalAmount) AS nullableInvoiceAmount
                            FROM bids b2
                            WHERE status = 'Completed'
                            AND endDate BETWEEN date_trunc('month', CURRENT_DATE - interval '1'
month) AND date_trunc('month', CURRENT_DATE)
                            GROUP BY b2.caretakerusername) c2 ON
                        c.caretakerusername = c2.caretakerusername2)
                  AS c4
          ) AS t1
        ) AS t2
      ) AS t3
    ) AS t4
  ORDER BY role, profitMargin DESC
-- Arrange by PT or FT, then the profit margin in decreasing order
;
```

8. Software Tools and Frameworks

For the Client Application (Front-End), we used **ReactJS Framework**. The tools and libraries used for Front-End are: Material UI, Formik, Axios, Date-fns, Lodash, Yup, Js-cookie and Moment.

For the Application Server (Back End), the framework used are **NodeJS and Express.** We have also used the following tools and libraries for the back end: Passport, Express Session, Connect PG Simple, Body-parser, Cors, Dotenv, Nodemon and Lodash.

Please also note that the DBMS Language used is PostgreSQL and hosting is carried out on Heroku. Code linting was carried out using ESLint and Prettier.

9. Difficulties Encountered and Lessons Learnt

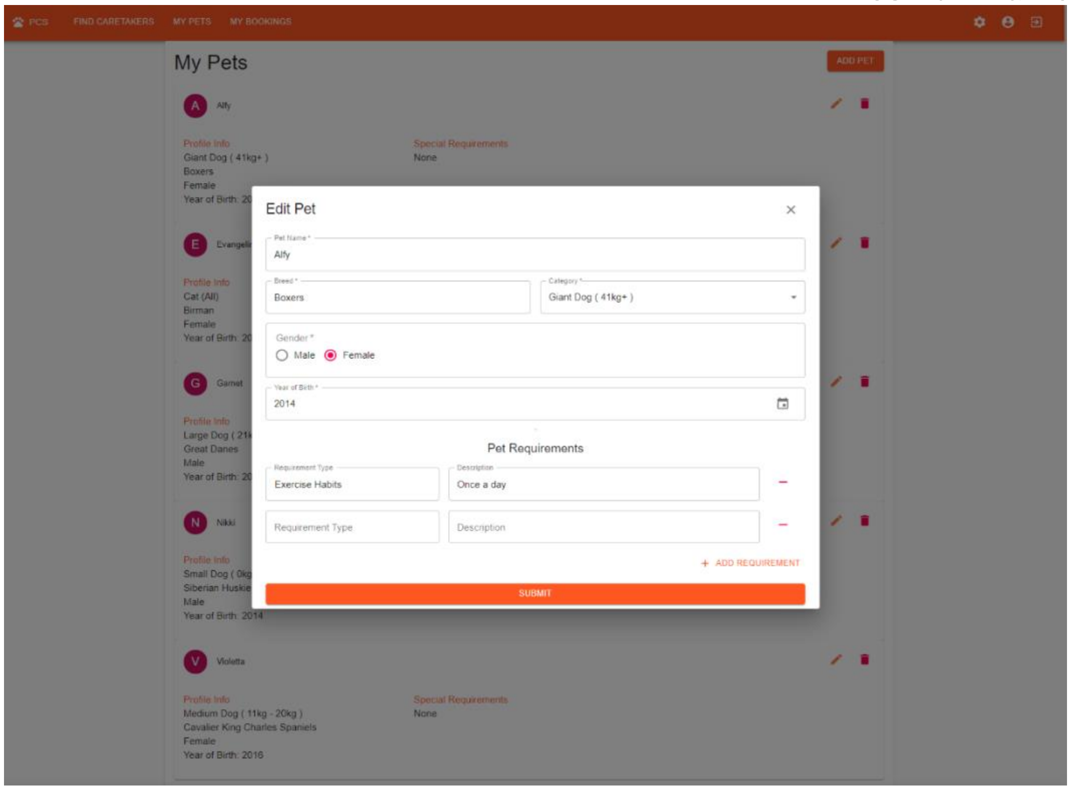
Database Systems: One lesson learnt is that it is essential to thoroughly evaluate the schema and the constraints we desired before beginning to implement them. Even after we thought we had done thorough checking, there were still minor discrepancies that led to bugs down the road that were hard to discover.

Web Development: Hosting on Heroku was done at a later stage of our implementation. We came across difficulty in logging in using our accounts on Heroku, even though it was working fine locally. Due to our lack of experience in hosting, it was difficult to debug what went wrong in the setup. After a few hours of trial and error, we realized that session table was not created on Heroku database. We feel that it would be better set up on production mode once MVP was implemented. As such, we can iteratively check that all the features are working fine in the production mode, rather than to host and test the system at the very last minute.

Timestamp data type: The TIMESTAMP data type in SQL caused several difficulties during our project. To start off, we had difficulty parsing and maintaining the format of TIMESTAMP in SQL. Later, we realized that our backend and database used different time zones. To add on, the database hosted by Heroku and the database on our local machines also used different time zones. These issues were not clear early on and caused us to spend some time searching for what went wrong.

10. Application Screenshots

10.1 Pet Owner editing his Pet Profile



My Pets

Alfy

Profile Info
Giant Dog (41kg+)
Boxers
Female
Year of Birth: 20

Special Requirements
None

Evangeli

Profile Info
Cat (All)
Birman
Female
Year of Birth: 20

Garnet

Profile Info
Large Dog (21k
Great Danes
Male
Year of Birth: 20

Nikki

Profile Info
Small Dog (0kg
Siberian Huskie
Male
Year of Birth: 2014

Violetta

Profile Info
Medium Dog (11kg - 20kg)
Cavalier King Charles Spaniels
Female
Year of Birth: 2016

Special Requirements
None

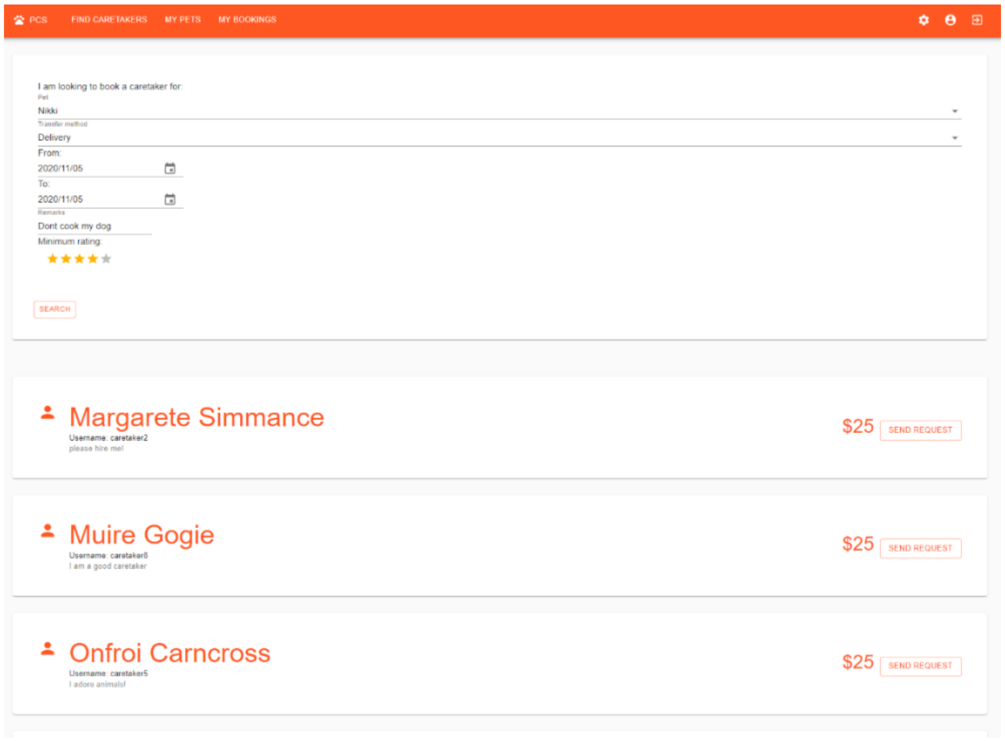
Edit Pet

Pet Name *: Alfy
Breed *: Boxers
Category *: Giant Dog (41kg+)
Gender *: Male [✓ Female]
Year of Birth *: 2014

Pet Requirements

Requirement Type	Description
Exercise Habits	Once a day
Requirement Type	Description

10.2 Pet Owner searching for a suitable Caretaker for their Pet



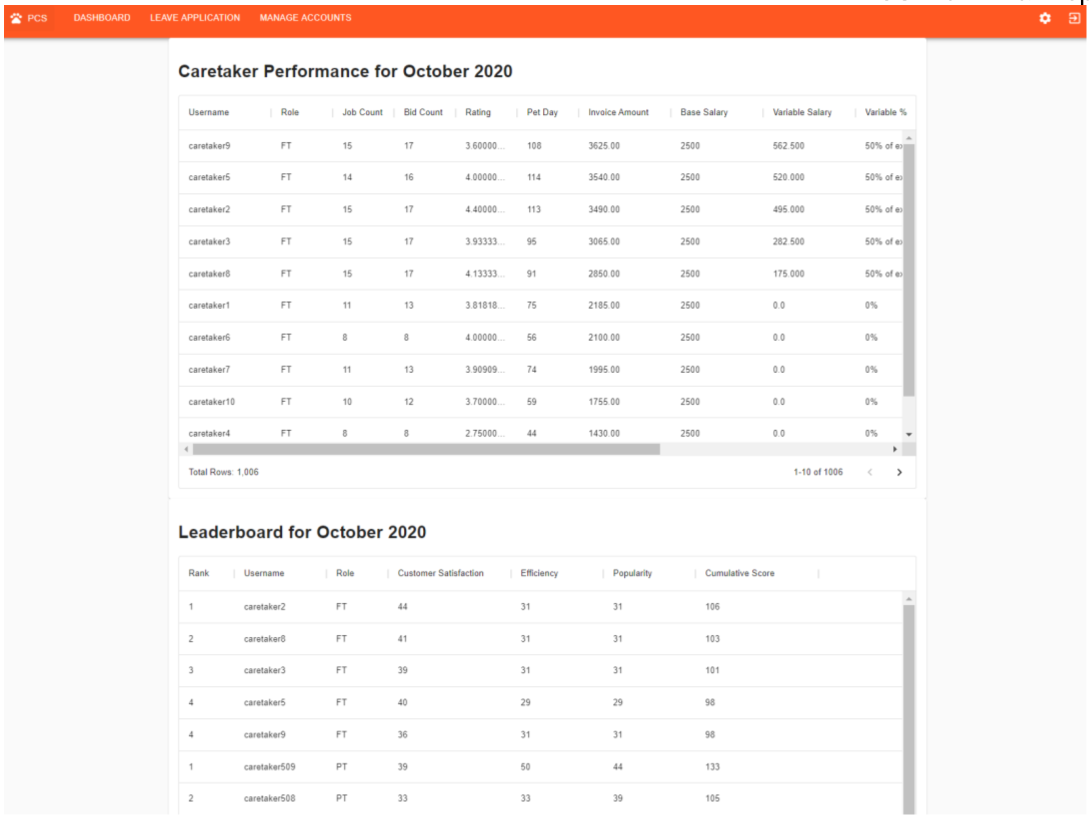
I am looking to book a caretaker for:
Pet: Nikki
Transfer method: Delivery
From: 2020/11/05
To: 2020/11/05
Remarks: Dont cook my dog
Minimum rating: ★★★★☆

Margarete Simmance
Username: caretaker2
please hire me!
$25

Muire Gogie
Username: caretaker8
I am a good caretaker
$25

Onfroi Carncross
Username: caretaker5
I adore animals!
$25

10.3 PCS Administrator's Summary Page



Caretaker Performance for October 2020

Username	Role	Job Count	Bid Count	Rating	Pet Day	Invoice Amount	Base Salary	Variable Salary	Variable %
caretaker9	FT	15	17	3.60000...	108	3625.00	2500	562.500	50% of e
caretaker5	FT	14	16	4.00000...	114	3540.00	2500	520.000	50% of e
caretaker2	FT	15	17	4.40000...	113	3490.00	2500	495.000	50% of e
caretaker3	FT	15	17	3.93333...	95	3065.00	2500	282.500	50% of e
caretaker8	FT	15	17	4.13333...	91	2850.00	2500	175.000	50% of e
caretaker1	FT	11	13	3.81818...	75	2185.00	2500	0.0	0%
caretaker6	FT	8	8	4.00000...	56	2100.00	2500	0.0	0%
caretaker7	FT	11	13	3.90909...	74	1995.00	2500	0.0	0%
caretaker10	FT	10	12	3.70000...	59	1755.00	2500	0.0	0%
caretaker4	FT	8	8	2.75000...	44	1430.00	2500	0.0	0%

Total Rows: 1,006 1-10 of 1006

Leaderboard for October 2020

Rank	Username	Role	Customer Satisfaction	Efficiency	Popularity	Cumulative Score
1	caretaker2	FT	44	31	31	106
2	caretaker8	FT	41	31	31	103
3	caretaker3	FT	39	31	31	101
4	caretaker5	FT	40	29	29	98
4	caretaker9	FT	36	31	31	98
1	caretaker509	PT	39	50	44	133
2	caretaker508	PT	33	33	39	105