PET CARE MANAGEMENT SYSTEM

Submitted for:

DATABASE MANAGEMENT SYSTEM (UCS310)

Submitted by:

NIKITA	102317245
KASHIKA JINDAL	102317154
MANYA JINDAL	102317253
KUSHALI GUPTA	102317148

BE Second Year Batch – 2Q24

Submitted to: Mr. Shashank Singh

Computer Science and Engineering Department

Thapar Institute of Engineering & Technology, Patiala

Jan - June 2025

DECLARATION

We solemnly declare that the project report titled ”Petcare Management System” , submitted by our group as a part of the internal assessment for the subject Database and Management System (UCS 310), is a result of our independent and original work carried out during the academic session [2024-2025].

This report has been prepared under the supervision and guidance of Mr.Shashank Singh, and we are grateful for his constant support, encouragement, and valuable feedback throughout this project.

We further declare that this work has not been submitted, either completely or partially, to any other course, subject, or institution for academic credit or evaluation. All sources of information, data, figures, and literature used in the project have been properly acknowledged to the best of our knowledge.

This report is a genuine and honest attempt to explore and apply the concepts of Database and Management System.

ABSTRACT

This project focuses on the development of a Pet Care Management System using LiveSQL to create an efficient and well-structured database solution. The aim is to simplify and organize the management of pet care services, which include pet boarding, caretaker assignment, health monitoring, appointment scheduling, and payment tracking. The system is designed to handle data for various stakeholders such as pet owners, pets, caretakers, and veterinary staff, ensuring secure and consistent access to important information.

The project begins with identifying relevant entities and relationships, followed by the creation of an Entity-Relationship (ER) model to visualize the system structure. The database is then normalized up to the BCNF to eliminate redundancy and enhance data integrity.

By relying solely on LiveSQL, the project demonstrates how a robust backend system can be created without the use of external technologies or interfaces. The resulting database is scalable, reliable, and capable of supporting a future web or mobile application that could offer a complete pet care service to users. This work lays the foundation for building a full-fledged application by providing a secure, organized, and easily maintainable data structure tailored to the needs of modern pet care services.

TABLE OF CONTENT

1. PROBLEM STATEMENT

In recent years, the demand for reliable pet care services has grown significantly as more individuals face situations where they must leave their pets behind due to work commitments, travel, or emergencies. However, existing solutions—such as informal arrangements with friends or unverified local sitters—often lack standardization, transparency, and accountability. These limitations raise concerns over the quality of care, safety, and health monitoring provided to pets in their owners' absence.

2. INTRODUCTION

In today's fast-paced and dynamic world, owning a pet is a source of joy, companionship, and emotional support. However, many pet owners often face difficulties when they need to travel for work, attend emergencies, or go on vacations. During such times, ensuring the safety, comfort, and well-being of their beloved pets becomes a major concern. Relying on friends, neighbors, or unverified pet sitters may not guarantee the level of care that pets truly need. To address this growing concern, we propose the development of a **Pet Care Management System** — a centralized digital platform that streamlines and modernizes the process of booking pet care services.

The primary goal of this system is to provide a reliable, secure, and structured environment for pet care, where owners can confidently leave their pets knowing they are in good hands. The platform allows pet owners to register their pets, choose a care package, and book temporary stays at certified pet care centers. These centers are equipped with trained caretakers, medical facilities, and proper accommodations to ensure the pets are well taken care of.

To meet varying customer needs and budgets, the system offers **three distinct service plans**:

- **Premium Plan**: Includes luxury boarding with air-conditioned rooms, 24/7 CCTV monitoring for remote viewing, daily visits from veterinary professionals, customized meals based on dietary needs, grooming services, and play sessions.

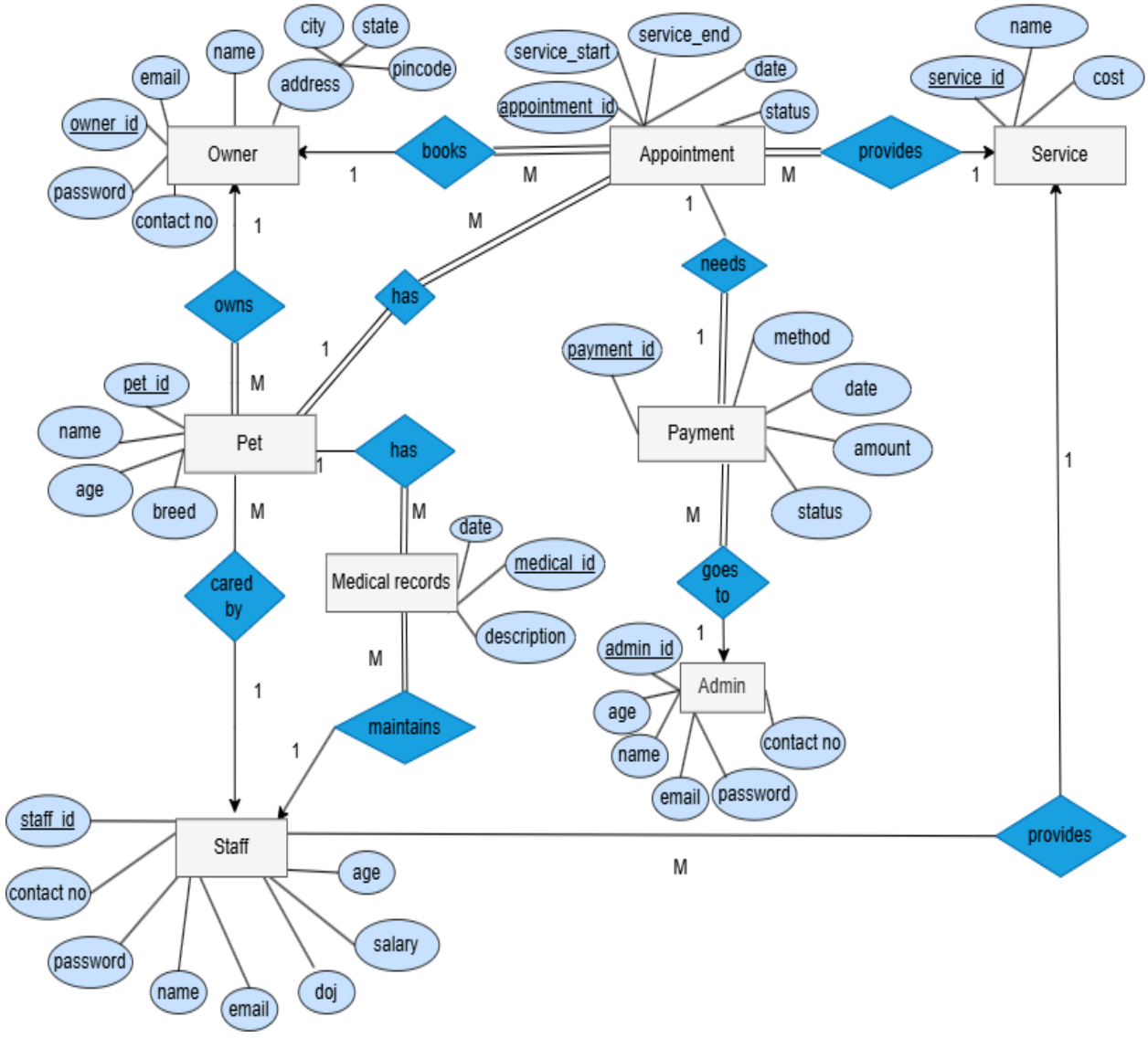
- **Standard Plan**: Offers clean and safe rooms, regular feeding, health check-ups every alternate day, and structured playtime under caretaker supervision.

- **Basic Plan**: Provides essential services including hygienic boarding, routine feeding, and daily caretaker visits to monitor basic health and well-being.

The backend of the project is powered by a **Database Management System (DBMS)**, which plays a crucial role in managing the large volume of data generated. Information related to pet profiles, owner details, bookings, caretakers, schedules, payments, and health records is stored in relational databases. Through the use of structured queries, normalization, integrity constraints, and relationships between tables, the system ensures data accuracy, security, and efficiency.

This project demonstrates how effective database solutions can transform traditional pet care services into an organized, scalable, and customer-friendly model. By leveraging DBMS principles, the Pet Care Management System enhances the overall experience for both service providers and pet owners, ensuring that pets receive the quality care they deserve.

3. ER- DIAGRAM


Owner
owner_id
email
name
address
city
state
pincode
password
contact no
books
Appointment
appointment_id
service_start
service_end
date
status
provides
Service
service_id
name
cost
owns
has
needs
Pet
pet_id
name
age
breed
has
Payment
payment_id
method
date
amount
status
Medical records
date
medical_id
description
cared by
goes to
maintains
Admin
admin_id
age
name
email
password
contact no
Staff
staff_id
contact no
password
name
email
age
salary
doj
provides
1
M

4. ER- TABLES

```
-- Table: Owner
CREATE TABLE Owner (
    owner_id INT  PRIMARY KEY,
    oname VARCHAR(100) ,
    email VARCHAR(100) UNIQUE ,
    opassword VARCHAR(100),
    contact_no VARCHAR(15)
);

-- Table: Address
CREATE TABLE Address (
    address_id INT PRIMARY KEY,
    city VARCHAR(50),
    state VARCHAR(50) ,
    pincode VARCHAR(10) ,
    owner_id INT UNIQUE ,
    FOREIGN KEY (owner_id) REFERENCES Owner(owner_id) ON DELETE CASCADE
);

-- Table: Service
CREATE TABLE Service (
    service_id INT PRIMARY KEY,
    sname VARCHAR(100),
    cost DECIMAL(10, 2) CHECK (cost >= 0)
);


-- Table: Staff
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY,
    sname VARCHAR(100) NOT NULL,
    age INT CHECK (age >= 0),
    email VARCHAR(100) UNIQUE NOT NULL,
    spassword VARCHAR(100) NOT NULL,
    contact_no VARCHAR(15) NOT NULL,
    doj DATE NOT NULL,
    salary DECIMAL(10, 2) CHECK (salary >= 0),
    service_id INT,
    FOREIGN KEY (service_id) REFERENCES Service(service_id)}ON DELETE SET NULL;
```

```sql
-- Table: Pet
CREATE TABLE Pet (
    pet_id INT PRIMARY KEY,
    pname VARCHAR(100) ,
    age INT CHECK (age >= 0),
    breed VARCHAR(50),
    owner_id INT,
    staff_id INT,
    FOREIGN KEY (owner_id) REFERENCES Owner(owner_id) ON DELETE CASCADE,
    FOREIGN KEY (staff_id) REFERENCES Staff(staff_id)ON DELETE SET NULL,
);
```

```sql
-- Table: Medical_Record
CREATE TABLE Medical_Record (
    medical_id INT PRIMARY KEY,
    mdate DATE,
    mdescription VARCHAR(500) ,
    pet_id INT ,
    staff_id INT,
    FOREIGN KEY (pet_id) REFERENCES Pet(pet_id)ON DELETE CASCADE,
    FOREIGN KEY (staff_id) REFERENCES Staff(staff_id)ON DELETE SET NULL
);
```

```sql
-- Table: Appointment
CREATE TABLE Appointment (
    appointment_id INT PRIMARY KEY,
    service_start DATE ,
    service_end DATE,
    adate DATE ,
    astatus VARCHAR(50) DEFAULT 'Pending',
    pet_id INT ,
    owner_id INT,
    service_id INT ,
    FOREIGN KEY (pet_id) REFERENCES Pet(pet_id) ON DELETE CASCADE,
    FOREIGN KEY (owner_id) REFERENCES Owner(owner_id) ON DELETE CASCADE,
    FOREIGN KEY (service_id) REFERENCES Service(service_id) ON DELETE SET NULL,
    CHECK (service_end >= service_start)
);
```

```sql
-- Table: Admin
CREATE TABLE Admin (
    admin_id INT PRIMARY KEY,
    aname VARCHAR(100) NOT NULL,
    age INT CHECK (age >= 0),
    email VARCHAR(100) UNIQUE NOT NULL,
    apassword VARCHAR(100) NOT NULL,
    contact_no VARCHAR(15) NOT NULL
);

-- Table: Payment
CREATE TABLE Payment (
    payment_id INT  PRIMARY KEY,
    method VARCHAR(50) ,
    pdate DATE ,
    amount DECIMAL(10, 2) CHECK (amount >= 0),
    pstatus VARCHAR(50) DEFAULT 'Unpaid',
    appointment_id INT,
    admin_id INT ,
    FOREIGN KEY (appointment_id) REFERENCES Appointment(appointment_id)ON DELETE
CASCADE,
    FOREIGN KEY (admin_id) REFERENCES Admin(admin_id)ON DELETE CASCADE,
    CHECK ((pstatus = 'Paid' AND pdate IS NOT NULL) OR (pstatus = 'Pending' AND pdate
IS NULL))

);
```

5. FUNCTIONS AND TRIGGERS

1.Function to calculate no of days service provided

```
CREATE OR REPLACE FUNCTION fn_days_between(dt1 DATE, dt2 DATE)
RETURN INT
IS
  days_diff INT;
BEGIN
  days_diff := dt2 - dt1+1;
  RETURN  days_diff;
END;
```

2.Function to calculate total cost

```
2.CREATE OR REPLACE FUNCTION fn_total_cost (
  svc IN NUMBER,
  days IN NUMBER
) RETURN NUMBER IS
  unit_cost NUMBER(10, 2);
BEGIN
  SELECT cost INTO unit_cost FROM Service WHERE service_id = svc;
  RETURN unit_cost * days;
END;
```

3. Trigger for updating total payment

```
CREATE OR REPLACE TRIGGER trg_calc_payment_amount
BEFORE INSERT OR UPDATE ON Payment
FOR EACH ROW
DECLARE
  v_service_id NUMBER;
  v_start DATE;
  v_end DATE;
  v_days NUMBER;
BEGIN
  SELECT service_start, service_end, service_id
  INTO v_start, v_end, v_service_id
  FROM Appointment
```

```
  WHERE appointment_id = :NEW.appointment_id;


  v_days := fn_days_between(v_start, v_end);
  :NEW.amount := fn_total_cost(v_service_id, v_days);
END;
```

4. Trigger for updating appointment status

```
CREATE OR REPLACE TRIGGER trg_update_astatus
BEFORE INSERT OR UPDATE ON Appointment
FOR EACH ROW
BEGIN
  IF :NEW.service_end >= SYSDATE THEN
    :NEW.astatus := 'Scheduled';
  ELSE
    :NEW.astatus := 'Completed';
  END IF;
END;
```

6. QUERIES

6.1 Display All Pets Along With Owner And Staff name

```
SELECT p.pet_id, p.pname, o.oname AS owner_name, s.sname AS staff_name

FROM Pet p

JOIN Owner o ON p.owner_id = o.owner_id

JOIN Staff s ON p.staff_id = s.staff_id order by p.pet_id;
```

	PET_ID	PNAME	OWNER_NAME	STAFF_NAME
1	1	Buddy	Fiona Green	Alice
2	2	Milo	George Hall	Bob
3	3	Luna	Hannah Scott	Diana
4	4	Max	Fiona Green	Neeraj
5	5	Bella	Ian Moore	Ravi
6	6	Tiger	Rahul Verma	Charlie
7	7	Daisy	Sneha Kapoor	Ethan
8	8	Oscar	Amit Jain	Anjali
9	9	Tommy	Divya Rana	Tina
10	10	Lily	Divya Jyoti	Aditya Menon

6.2 Count of pets per owner

```
SELECT o.owner_id, o.oname, COUNT(p.pet_id) AS num_pets

FROM Owner o

LEFT JOIN Pet p ON o.owner_id = p.owner_id

GROUP BY o.owner_id, o.oname;
```

	OWNER_ID	ONAME	NUM_PETS
1	1	Fiona Green	2
2	2	George Hall	1
3	3	Hannah Scott	1
4	4	Ian Moore	1
5	5	Julia Adams	0
6	6	Rahul Verma	1
7	7	Sneha Kapoor	1
8	8	Amit Jain	1
9	9	Divya Rana	1
10	10	Divya Jyoti	1

6.3 List all pending payments

```
SELECT *

FROM Payment

WHERE pstatus = 'Pending';
```

	PAYMENT_ID	METHOD	PDATE	AMOUNT	PSTATUS	APPOINTMENT_ID	ADMIN_ID
1	3	Cash	*(null)*	1200	Pending	3	1
2	5	Cash	*(null)*	2100	Pending	5	1
3	7	Cash	*(null)*	300	Pending	7	1
4	9	Cash	*(null)*	2000	Pending	9	1

6.4 Average appointment length per service

```
SELECT s.sname,

       AVG(((a.service_end)-(a.service_start))) AS avg_days

FROM Service s

JOIN Appointment a ON s.service_id = a.service_id

GROUP BY s.service_id, s.sname;
```

	SNAME	AVG_DAYS
1	Premium_care	3.3333333333333335
2	Standard_care	2
3	Basic_care	2.6666666666666665

6.5 Total payment received vs pending

```
SELECT pstatus, SUM(amount) AS total_amount

FROM Payment

GROUP BY pstatus;
```

	PSTATUS	TOTAL_AMOUNT
1	Paid	8050
2	Pending	4650

6.6 View all staff details along with their service name

```
SELECT s.staff_id, s.sname, s.age, s.email, s.contact_no, s.salary, sv.sname
AS service_name

FROM Staff s

LEFT JOIN Service sv ON s.service_id = sv.service_id;
```

	STAFF_ID	SNAME	AGE	EMAIL	CONTACT_NO	SALARY	SERVICE_NAME
1	1	Alice	28	alice@example.com	9876543210	25000	Premium_care
2	2	Bob	32	bob@example.com	9876543211	26000	Premium_care
3	3	Charlie	29	charlie@example.cor	9876543212	25500	Premium_care
4	4	Diana	35	diana@example.com	9876543213	27000	Standard_care
5	5	Ethan	30	ethan@example.com	9876543214	26500	Standard_care
6	6	Ravi	28	ravi.k@petcare.com	9012345678	30000	Standard_care
7	7	Anjali	31	anjali.v@petcare.com	8899776655	28000	Standard_care
8	8	Neeraj	26	neeraj.d@petcare.co	7788990011	25000	Basic_care
9	9	Tina	35	tina.d@petcare.com	9911223344	31000	Basic_care
10	10	Aditya Menon	30	aditya.m@petcare.cc	9090909090	27000	Basic_care

6.7 List all upcoming appointments with owner and pet details

```
SELECT a.appointment_id, TO_CHAR(a.service_start,'DD-MON-YYYY')as service_start,
TO_CHAR(a.service_end,'DD-MON-YYYY') as service_end,TO_CHAR (a.adate,'DD-MON-YYYY')as
app_date, a.astatus,
       o.oname AS owner_name, p.pname AS pet_name, s.sname AS service_name
FROM Appointment a
JOIN Owner o ON a.owner_id = o.owner_id
JOIN Pet p ON a.pet_id = p.pet_id
JOIN Service s ON a.service_id = s.service_id
WHERE a.status = 'Scheduled'
ORDER BY a.service_start;
```

	APPOINTMENT_ID	SERVICE_START	SERVICE_END	APP_DATE	ASTATUS	OWNER_NAME
1	8	04-MAY-2025	12-MAY-2025	01-MAY-2025	Scheduled	Amit Jain
2	2	05-MAY-2025	09-MAY-2025	03-MAY-2025	Scheduled	George Hall
3	3	07-MAY-2025	10-MAY-2025	05-MAY-2025	Scheduled	Fiona Green

6.8 Get medical records for a specific pet (e.g., pet_id = 1)

```
SELECT mr.medical_id, TO_CHAR(mr.mdate,'DD-MON-YYYY')as medical_date, mr.mdescription,
st.sname AS staff_name

,p.pname as pet_name

FROM Medical_Record mr

JOIN Staff st  ON mr.staff_id = st.staff_id JOIN Pet p on p.staff_id=st.staff_id

WHERE mr.pet_id = 1;
```

	MEDICAL_ID	MEDICAL_DATE	MDESCRIPTION	STAFF_NAME	PET_NAME
1	1	10-APR-2025	Routine health monitoring during boarding. Vital signs normal, pet active and eating well.	Alice	Buddy

6.9 To calculate for how many days the appointment is provided and total cost

```
SELECT

  a.appointment_id,

  o.oname,

  TO_CHAR(a.service_start, 'DD-MON-YYYY') AS service_start,

  TO_CHAR(a.service_end, 'DD-MON-YYYY') AS service_end,

  fn_days_between(a.service_start, a.service_end) AS service_duration_days,

  fn_total_cost(a.service_id, fn_days_between(a.service_start, a.service_end)) AS
total_cost

FROM
```

```
Appointment a

JOIN Owner o ON a.owner_id = o.owner_id

JOIN Service s ON a.service_id = s.service_id;
```

	APPOINTMENT_ID	ONAME	SERVICE_START	SERVICE_END	SERVICE_DURATION_DAYS	TOTAL_COST
1	1	Fiona Green	10-APR-2025	14-APR-2025	5	2500
2	3	Fiona Green	07-MAY-2025	10-MAY-2025	4	1200
3	2	George Hall	05-MAY-2025	09-MAY-2025	5	2500
4	4	Hannah Scott	14-APR-2025	18-APR-2025	5	2000
5	5	Ian Moore	14-APR-2025	20-APR-2025	7	2100
6	6	Rahul Verma	18-APR-2025	20-APR-2025	3	1500
7	7	Sneha Kapoor	19-APR-2025	19-APR-2025	1	300
8	8	Amit Jain	04-MAY-2025	12-MAY-2025	9	2700
9	9	Divya Rana	21-APR-2025	25-APR-2025	5	2000
10	10	Divya Jyoti	22-APR-2025	26-APR-2025	5	2000

7. Conclusion

In this project, we successfully designed and implemented a robust relational database for the Pet Care management system. The database effectively handles core entities such as owners, staff, pets, services, and appointments with clearly defined relationships and constraints to maintain data integrity. Through practical SQL operations, including table creation, data insertion, and enforcement of referential integrity, we ensured that the system can reliably store and manage comprehensive information required for efficient pet care services. This implementation not only strengthens data organization but also lays a solid foundation for future scalability, enabling advanced features like analytics and customer relationship management to be integrated seamlessly.

8.References

https://www.w3schools.com/sql/default.asp

https://www.geeksforgeeks.org/pl-sql-tutorial/#plsql-triggers

https://www.prepbytes.com/blog/dbms/trigger-in-dbms/