Mini project report on

Disaster Data Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE23CS351A – DBMS Project

Submitted by:

Shakirth Anisha	**PES2UG23CS927**
Samridhi Shreya	**PES2UG23CS907**

under the guidance of

Prof. Nivedita Kasturi

Assistant Professor

PES University

AUG - DEC 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Disasters data Management System

is a bonafide work carried out by

Shakirth Anisha	**PES2UG23CS927**
Samridhi Shreya	**PES2UG23CS907**

In partial fulfilment for the completion of fifth semester DBMS Project (UE23CS351) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2025 – DEC. 2025. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Prof. Nivedita Kasturi

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Pet Adoption Management System** has been carried out by us under the guidance of **Prof. Nivedita Kasturi, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2025.

Shakirth Anisha	**PES2UG23CS927**	**Shakirth Anisha**
Samridhi Shreya	**PES2UG23CS907**	**Samridhi Shreya**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Nivedita Kasturi, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE23CS351 - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor,ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Late Dr. M. R. Doreswamy, Founder, PES University, whose vision and dedication continue to inspire generations of learners. I would also like to express my sincere gratitude to Prof. Jawahar Doreswamy, Chancellor, PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University and Prof. Nagarjuna Sadineni, Pro Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

TABLE OF CONTENTS

1. INTRODUCTION

Animal shelters and rescue organizations play a vital role in the welfare of animals, providing them with safety, medical care, and the opportunity for a new life. However, the operational side of these organizations is often complex and resource-intensive. They must manage detailed records for hundreds of animals, track their medical history and status, process numerous adoption applications, and coordinate between staff, volunteers, and potential adopters.

In many cases, these operations are handled through manual paperwork or disparate, inefficient digital systems (like spreadsheets). This can lead to critical errors, such as:

- Allowing an animal to be adopted twice.
- Losing track of an animal's medical needs.
- A slow and frustrating application process for adopters.
- Inaccurate reporting on shelter capacity or adoption rates.

A Database Management System (DBMS) provides a robust, centralized, and scalable solution to these problems. By creating a well-structured relational database, we can ensure data integrity, automate complex workflows, and provide a single source of truth for all stakeholders.

This project, the "Pet Adoption Management System," aims to design and implement such a database using MySQL. The system is built to manage all core aspects of a pet shelter, including:

- **User Management:** Differentiating between general users, approved adopters, shelter workers, and administrators, each with specific roles.
- **Pet Inventory:** Cataloging pets with details on species, breed, age, and real-time status (e.g., 'Available', 'Adopted', 'Medical Hold').
- **Adoption Workflow:** Managing the complete lifecycle of an adoption application, from submission and payment to approval or rejection.
- **Data Automation:** Utilizing stored procedures, functions, and triggers to enforce business rules and automate repetitive tasks.

This report details the complete lifecycle of the project, from the initial problem definition and data modeling (ER and Relational) to the final implementation of DDL, DML, advanced SQL queries, and procedural logic.

2. PROBLEM DEFINITION

The primary goal of this project is to develop a comprehensive and efficient database system to manage the operations of a pet adoption agency. The existing manual or semi-automated systems are often plagued with inefficiencies and data integrity issues.

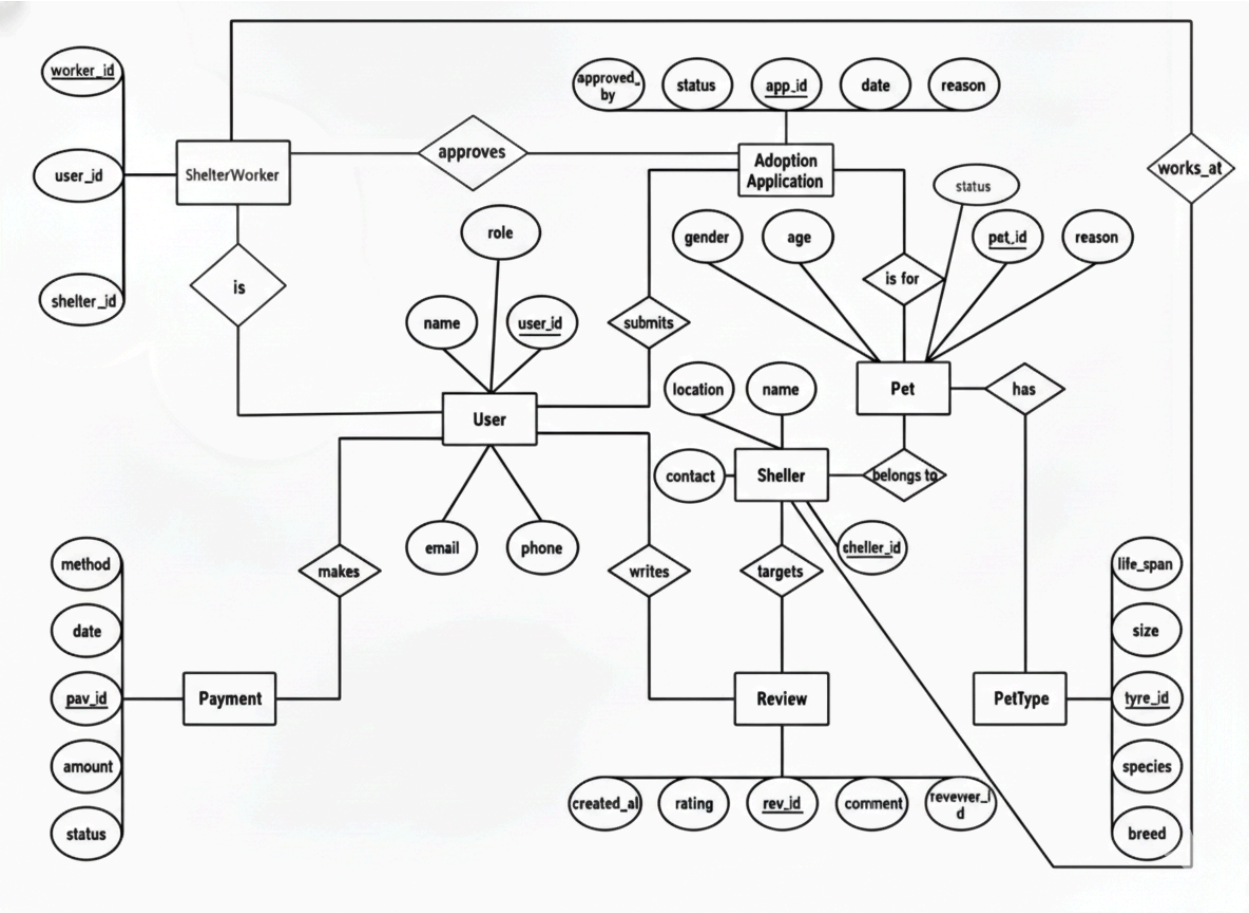
The core problems to be solved are:

1. **Inadequate Pet Status Tracking:** Shelters struggle to maintain accurate, real-time information on a pet's status. A pet might be listed as 'Available' on a website while it is actually on 'Medical Hold' or already 'Adopted'. This leads to confusion and disappointment for adopters.
2. **Complex Adoption Workflow:** The adoption process is multi-stepped (application, review, payment, approval). Handling this manually is slow. Furthermore, when one pet receives multiple applications, there is no streamlined way to approve one and automatically notify the others.
3. **Data Inconsistency and Redundancy:** Without proper constraints, the system could store conflicting information. For example, a user might submit an application for a pet that doesn't exist, or a payment record might be orphaned without a link to an application.
4. **Lack of Role-Based Access:** A public-facing user (adopter) should not have the same permissions as a shelter worker or an admin. A system is needed to manage different user roles and their capabilities within the database.
5. **Difficulty in Reporting:** Management cannot easily retrieve simple operational metrics, such as "How many pets are currently available at Shelter X?" or "Who are our most active adopters?"

Objectives: To solve these problems, the proposed "Pet Adoption Management System" will:

- **Ensure Data Integrity:** Implement primary keys, foreign keys, `CHECK` constraints, and triggers to maintain logical consistency.
- **Automate Business Logic:** Use stored procedures to handle complex operations like `ApproveApplication` and `AddAdoptionApplication`, ensuring all related tables (Pet, Payment, Application) are updated atomically.
- **Create an Audit Trail:** Employ triggers to log critical changes, such as any update to a pet's status, in a separate `PetStatusLog` table.
- **Simplify Data Access:** Create views like `AvailablePets` and `AdoptionSummary` to provide simple, pre-compiled answers to common questions.
- **Provide Utility:** Use database functions to offer quick calculations, such as `CountAvailablePets(shelter_id)` and `AvgPetAgeInShelter(shelter_id)`.

3. ER MODEL



The Emergency Room (ER) model provides a high-level, conceptual view of the database. It defines the key entities involved in the system and the relationships between them.

Key Entities:

- **User:** Represents a person interacting with the system. This entity stores details like `name`, `email`, `phone`, and `role`.
- **Shelter:** Represents the physical animal shelter. It stores the shelter's `name`, `location`, and `contact` information.
- **ShelterWorker:** A specialized type of **User** who works at a **Shelter**. It links a `user_id` to a `shelter_id`.
- **PetType:** A descriptive entity that stores information about a type of animal, such as `species`, `breed`, `life_span`, and `size`.
- **Pet:** Represents an individual animal in a shelter. It includes details like `name`, `age`, `gender`, and `status`. It is linked to a **Shelter** and a **PetType**.
- **AdoptionApplication:** An associative entity that represents a **User's** request to adopt a **Pet**. It stores the application `status`, `reason`, and the worker who `approved_by`.
- **Payment:** Represents a payment made by a **User**.

- **Review:** Allows a **User** to leave a `review` (rating and comment) for a **Shelter**.

Key Relationships:

- A **User** `is` a **ShelterWorker**.
- A **ShelterWorker** `works_at` one **Shelter**.
- A **Shelter** `belongs to` one or more **Pets**.
- A **Pet** `has` one **PetType**.
- A **User** `submits` one or more **AdoptionApplications**.
- A **Pet** `is for` one or more **AdoptionApplications**.
- A **ShelterWorker** `approves` zero or more **AdoptionApplications**.
- A **User** `makes` one or more **Payments**.
- A **User** `writes` one or more **Reviews**.
- A **Review** `targets` a **Shelter**.

4. ER TO RELATIONAL MAPPING

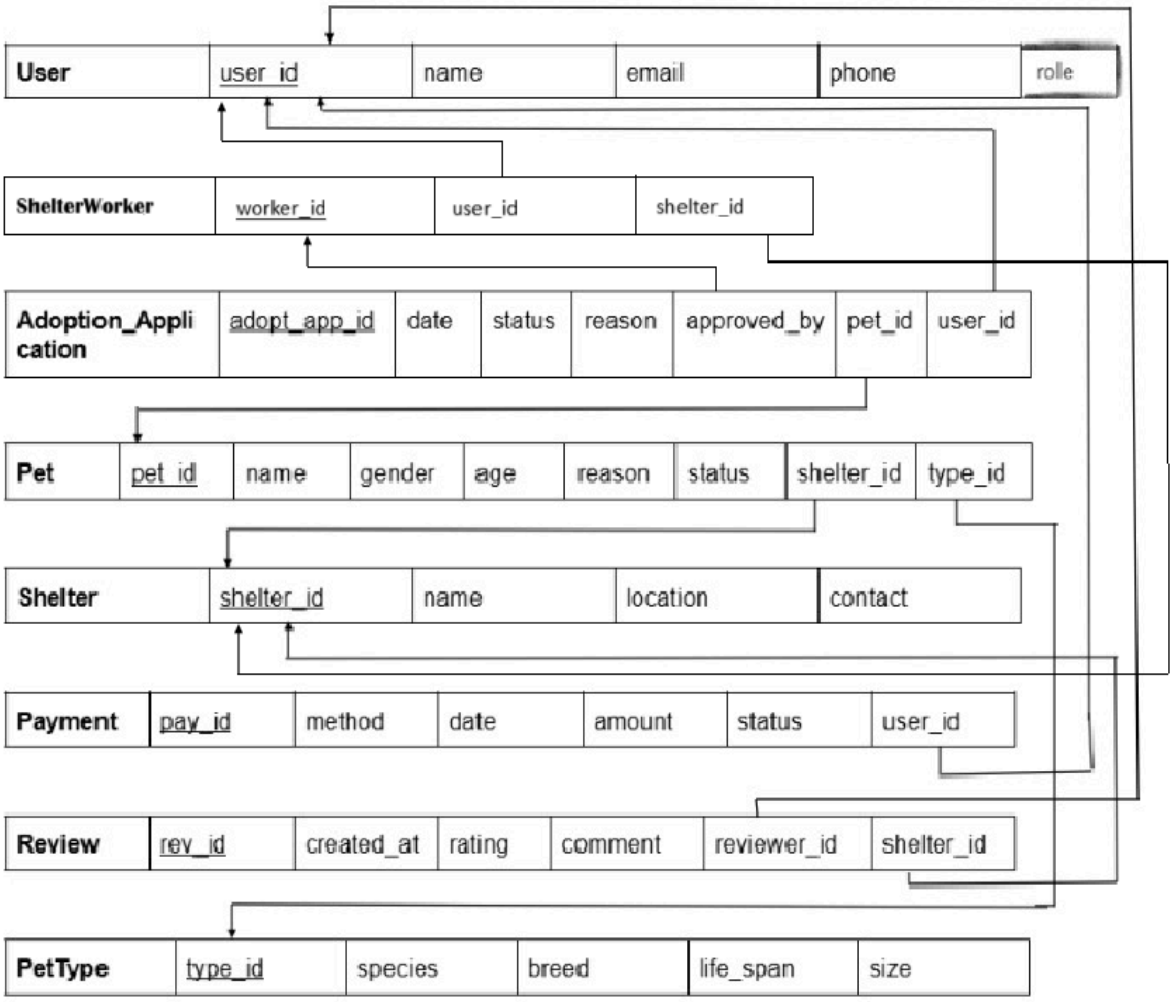
4.1 Steps of Algorithm for Chosen Problem

The conceptual ER model is mapped to a set of relational tables (schema) using a standard algorithm.

1. **Map Regular Entities:** Each regular entity (`User`, `Shelter`, `PetType`) becomes a table. The entity's attributes become columns, and its identifier becomes the primary key (PK).
    - `User` -> `User(user_id PK, name, email, ...)`
    - `Shelter` -> `Shelter(shelter_id PK, name, location, ...)`
    - `PetType` -> `PetType(type_id PK, species, breed, ...)`
2. **Map Weak Entities / Specialization:** The `ShelterWorker` entity, which is a specialized role of `User` and is also dependent on `Shelter`, is mapped as its own table. It contains foreign keys (FK) referencing both `User` and `Shelter`.
    - `ShelterWorker` -> `ShelterWorker(worker_id PK, user_id FK, shelter_id FK)`
3. **Map 1:N Relationships:** The 1:N relationships are mapped by adding a foreign key to the table on the 'N' side.
    - `Shelter houses Pet` (1:N): Add `shelter_id` (FK) to the `Pet` table.
    - `PetType classifies Pet` (1:N): Add `type_id` (FK) to the `Pet` table.
    - `User makes Payment` (1:N): Add `user_id` (FK) to the `Payment` table.
    - `User writes Review` (1:N): Add `reviewer_id` (FK) to the `Review` table.
4. **Map M:N Relationships (Associative Entities):** M:N relationships are resolved by creating a new junction table (associative entity).
    - `User applies for Pet`: This is modeled by the `AdoptionApplication` table, which has foreign keys `user_id` and `pet_id`.
5. **Map Other Relationships:**
    - `ShelterWorker approves AdoptionApplication` (1:N): Add `approved_by` (FK referencing `ShelterWorker`) to the `AdoptionApplication` table.
    - `Review` for `Shelter`: The `Review` table includes foreign key `shelter_id`.

4.2 Complete Diagram of Relational Mapping

This process results in the relational schema shown in the diagram below.


User
user_id
name
email
phone
rolle
ShelterWorker
worker_id
user_id
shelter_id
Adoption_Application
adopt_app_id
date
status
reason
approved_by
pet_id
user_id
Pet
pet_id
name
gender
age
reason
status
shelter_id
type_id
Shelter
shelter_id
name
location
contact
Payment
pay_id
method
date
amount
status
user_id
Review
rev_id
created_at
rating
comment
reviewer_id
shelter_id
PetType
type_id
species
breed
life_span
size

5. DDL STATEMENTS

Data Definition Language (DDL) statements are used to create and define the database structure. The following tables were created for the pet adoption system.

1. User Table

SQL

```
mysql> CREATE TABLE User (
    ->     user_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     email VARCHAR(150) UNIQUE NOT NULL,
    ->     password_hash VARCHAR(256) NOT NULL,
    ->     phone VARCHAR(20),
    ->     role ENUM('general', 'adopter', 'shelter_worker', 'admin') DEFAULT 'general',
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: Table `User` created successfully.

2. Shelter Table

SQL

```
mysql> CREATE TABLE Shelter (
    ->     shelter_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     location VARCHAR(255) NOT NULL,
    ->     contact VARCHAR(100),
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Output: Table `Shelter` created successfully.

3. ShelterWorker Table

SQL

```
mysql> CREATE TABLE ShelterWorker (
    ->     worker_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     user_id INT NOT NULL UNIQUE,
    ->     shelter_id INT NOT NULL,
    ->     FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (shelter_id) REFERENCES Shelter(shelter_id) ON DELETE CASCADE,
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Output: Table `ShelterWorker` created successfully.

4. PetType Table

SQL

```
mysql> CREATE TABLE PetType (
    ->     type_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     species VARCHAR(50) NOT NULL,
    ->     breed VARCHAR(50),
    ->     life_span INT CHECK (life_span >= 0),
    ->     size ENUM('Small', 'Medium', 'Large', 'Giant'),
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Output: Table `PetType` created successfully.

5. Pet Table

SQL

```
mysql> CREATE TABLE Pet (
    ->     pet_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100) NOT NULL,
    ->     gender ENUM('M', 'F', 'Unknown') DEFAULT 'Unknown',
    ->     age INT CHECK (age >= 0),
    ->     reason TEXT,
    ->     status ENUM('Available', 'Adopted', 'Medical Hold') DEFAULT 'Available',
    ->     shelter_id INT NOT NULL,
    ->     type_id INT NOT NULL,
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    ->     FOREIGN KEY (shelter_id) REFERENCES Shelter(shelter_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (type_id) REFERENCES PetType(type_id),
    ->     CONSTRAINT unique_pet_per_shelter UNIQUE (name, shelter_id),
    ->     CONSTRAINT chk_pet_status CHECK (status IN ('Available', 'Adopted', 'Medical Hold'))
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: Table `Pet` created successfully.

6. AdoptionApplication Table

SQL

```
mysql> CREATE TABLE AdoptionApplication (
    ->     adopt_app_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     date DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     status ENUM('Pending', 'Approved', 'Denied', 'Withdrawn') DEFAULT 'Pending',
    ->     reason TEXT,
    ->     approved_by INT,
    ->     pet_id INT NOT NULL,
    ->     user_id INT NOT NULL,
    ->     FOREIGN KEY (approved_by) REFERENCES ShelterWorker(worker_id),
    ->     FOREIGN KEY (pet_id) REFERENCES Pet(pet_id) ON DELETE CASCADE ON UPDATE CASCADE,
    ->     FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: Table `AdoptionApplication` created successfully.

7. Payment Table

SQL

```
mysql> CREATE TABLE Payment (
    ->     pay_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     method ENUM('Credit Card', 'Debit Card', 'UPI', 'PayPal', 'Cash') NOT NULL,
    ->     date DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     amount DECIMAL(10,2) NOT NULL CHECK (amount >= 0),
    ->     status ENUM('Completed', 'Pending', 'Failed', 'Refunded') DEFAULT 'Pending',
    ->     user_id INT NOT NULL,
    ->     adoption_app_id INT,
    ->     FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,
    ->     FOREIGN KEY (adoption_app_id) REFERENCES AdoptionApplication(adopt_app_id) ON DELETE CASCADE,
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     CONSTRAINT chk_amount_positive CHECK (amount >= 0)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: Table `Payment` created successfully.

8. Review Table

SQL

```
mysql> CREATE TABLE Review (
    ->     rev_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     rating INT CHECK (rating BETWEEN 1 AND 5),
    ->     comment TEXT,
    ->     reviewer_id INT NOT NULL,
    ->     shelter_id INT,
    ->     pet_id INT,
    ->     FOREIGN KEY (reviewer_id) REFERENCES User(user_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (shelter_id) REFERENCES Shelter(shelter_id) ON DELETE CASCADE,
    ->     FOREIGN KEY (pet_id) REFERENCES Pet(pet_id) ON DELETE SET NULL
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: Table `Review` created successfully.

9. WorkerRequest Table

```
mysql> CREATE TABLE IF NOT EXISTS WorkerRequest (
    ->     request_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     user_id INT NOT NULL,
    ->     message TEXT,
    ->     requested_role ENUM('shelter_worker', 'admin') DEFAULT 'shelter_worker',
    ->     status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
    ->     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->     updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    ->     FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    ->     INDEX idx_status (status),
    ->     INDEX idx_user_id (user_id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Output: WorkerRequest created successfully.

10. PetStatusLog Table (This table is created to support the triggers)

SQL

```
mysql> CREATE TRIGGER trg_log_pet_status_change
    -> BEFORE UPDATE ON pet_adoption_db.Pet
    -> FOR EACH ROW
    -> BEGIN
    ->     IF OLD.status <> NEW.status THEN
    ->         INSERT INTO pet_adoption_db.PetStatusLog (pet_id, old_status, new_status)
    ->         VALUES (OLD.pet_id, OLD.status, NEW.status);
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

Output: Table `PetStatusLog` created successfully.

Views: Views are virtual tables based on the result-set of an SQL statement.

SQL

```
mysql> CREATE VIEW AvailablePets AS
    -> SELECT
    ->     p.pet_id, p.name, pt.species, pt.breed,
    ->     s.name AS shelter_name, p.status
    -> FROM Pet p
    -> JOIN PetType pt ON p.type_id = pt.type_id
    -> JOIN Shelter s ON p.shelter_id = s.shelter_id
    -> WHERE p.status = 'Available';
Query OK, 0 rows affected (0.01 sec)
```

Output: View `AvailablePets` created successfully.

SQL

```
mysql> CREATE VIEW AdoptionSummary AS
    -> SELECT
    ->     a.adopt_app_id,
    ->     u.name AS adopter_name,
    ->     p.name AS pet_name,
    ->     a.status AS application_status,
    ->     s.name AS shelter_name,
    ->     a.date
    -> FROM AdoptionApplication a
    -> JOIN User u ON a.user_id = u.user_id
    -> JOIN Pet p ON a.pet_id = p.pet_id
    -> JOIN Shelter s ON p.shelter_id = s.shelter_id;
Query OK, 0 rows affected (0.01 sec)
```

Output: View `AdoptionSummary` created successfully.

6. DML STATEMENTS

Data Manipulation Language (DML) statements are used to insert, update, and delete data within the tables. The following statements were used to populate the database with initial sample data.

1. Inserting into User

SQL

```
mysql> INSERT INTO User (name, email, password_hash, phone, role) VALUES
    -> ('Shakirth Anisha', 'anisha@petsystem.com', SHA2('1234', 256), '9876543210', 'admin'),
    -> ('Samridhi Shreya', 'samridhi@petsystem.com', SHA2('sam123', 256), '9876501234', 'shelter_worker'),
    -> ('Angad Bhalla', 'angad@petsystem.com', SHA2('anga205', 256), '9876003210', 'adopter'),
    -> ('Suchitra Shankar', 'suchitra@petsystem.com', SHA2('suchichi', 256), '9876123456', 'adopter'),
    -> ('Tejas R', 'tejas@petsystem.com', SHA2('maggie', 256), '9988776655', 'shelter_worker'),
    -> ('Sanjana Saxena', 'sanjana@petsystem.com', SHA2('sanJ1', 256), '9123456789', 'general'),
    -> ('Risu Kumari', 'risu@petsystem.com', SHA2('yay@123', 256), '9765432109', 'adopter'),
    -> ('Taylor Swift', 'taylor@petsystem.com', SHA2('taytay13', 256), '9000011122', 'adopter'),
    -> ('Harry Styles', 'harry@petsystem.com', SHA2('ilovelouis', 256), '7086011282', 'adopter'),
    -> ('Emma Stone', 'emma@petsystem.com', SHA2('slay', 256), '7086921122', 'shelter_worker'),
    -> ('John Wick', 'john@petsystem.com', SHA2('ilovemydog', 256), '8012346772', 'shelter_worker');
Query OK, 11 rows affected (0.01 sec)
Records: 11  Duplicates: 0  Warnings: 0
```

Output: 11 rows inserted into `User`.

2. Inserting into Shelter

SQL

```
mysql> INSERT INTO Shelter (name, location, contact)
    -> VALUES
    -> ('PES EC Campus, Bangalore', 'Electronic City, Bangalore', 'pesec@petshelter.org'),
    -> ('Paws Shelter', 'Electronic City, Bangalore', 'paws@petshelter.org'),
    -> ('Happy Tails Home', 'Chennai, India', 'contact@happytails.in'),
    -> ('Pawfect Care Foundation', 'Bangalore', 'contact@pawfectcare.in'),
    -> ('Animal Ark Trust', 'Hyderabad', 'support@animalarktrust.org'),
    -> ('StreetPaws Rehabilitation', 'Delhi', 'help@streetpaws.org');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Output: 6 rows inserted into `Shelter`.

3. Inserting into ShelterWorker

SQL

```
mysql> -- SHELTER WORKERS
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO ShelterWorker (user_id, shelter_id)
    -> VALUES
    -> (1, 5),   -- Anisha at Animal Ark Trust
    -> (2, 2),  -- Samridhi at Paws Shelter
    -> (5, 1),  -- Tejas at PES EC Campus
    -> (10, 3),  -- Emma at Happy Tails Home
    -> (11, 4);   -- John at Pawfect Care Foundation
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Output: 5 rows inserted into `ShelterWorker`.

4. Inserting into PetType

SQL

```
mysql> INSERT INTO PetType (species, breed, life_span, size)
    -> VALUES
    -> ('Dog', 'Shih Tzu', 12, 'Small'),
    -> ('Dog', 'Beagle', 14, 'Medium'),
    -> ('Cat', 'Persian', 15, 'Small'),
    -> ('Parrot', 'African Grey', 50, 'Small'),
    -> ('Rabbit', 'Holland Lop', 10, 'Small'),
    -> ('Dog', 'German Shepherd', 10, 'Large'),
    -> ('Cat', 'Calico', 14, 'Small');
Query OK, 7 rows affected (0.00 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

Output: 7 rows inserted into `PetType`.

5. Inserting into Pet

SQL

```
mysql> INSERT INTO Pet (name, gender, age, reason, status, shelter_id, type_id)
    -> VALUES
    -> ('Yuki', 'M', 3, 'Owner relocation', 'Adopted', 1, 1),
    -> ('Milo', 'M', 2, 'Found stray', 'Adopted', 2, 6),
    -> ('Luna', 'F', 1, 'Abandoned kitten', 'Available', 1, 7),
    -> ('Snowball', 'F', 4, 'Health issues', 'Medical Hold', 3, 5),
    -> ('Zoomie', 'F', 2, 'Rescued stray cat', 'Available', 2, 3),
    -> ('Coco', 'M', 1, 'Owner could not care for it', 'Available', 3, 4),
    -> ('Coffee', 'F', 2, 'Rescued stray cat', 'Adopted', 2, 7);
Query OK, 7 rows affected (0.00 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

Output: 7 rows inserted into `Pet`.

6. Inserting into AdoptionApplication

SQL

```
mysql> INSERT INTO AdoptionApplication (status, reason, approved_by, pet_id, user_id)
    -> VALUES
    -> ('Pending', 'Want to adopt a cat for my family', NULL, 5, 4), -- Suchitra applying for Zoomie
    -> ('Approved', 'Looking for a friendly dog', 1, 1, 3), -- Angad approved for Yuki
    -> ('Pending', 'Not enough space at home', NULL, 2, 7), -- Risu denied for Milo
    -> ('Pending', 'Want a small pet for my apartment', NULL, 3, 8), -- Taylor applying for Luna
    -> ('Approved', 'Need a companinion', 2, 2, 9), -- Harry applying for Milo
    -> ('Approved', 'Looking for a companion for my dog', 1, 7, 3); -- Angad approved for Coffee
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Output: 6 rows inserted into `AdoptionApplication`.

7. Inserting into Payment

SQL

```
mysql>
mysql> INSERT INTO Payment (method, amount, status, user_id, adoption_app_id)
    -> VALUES
    -> ('Credit Card', 1500.00, 'Completed', 3, 2),   -- Angad pays for adopting Yuki
    -> ('UPI', 200.00, 'Pending', 4, 1),             -- Suchitra's payment is pending for Zoomie
    -> ('UPI', 500.00, 'Pending', 8, 4),         -- Taylor's payment completed for Coco
    -> ('PayPal', 800.00, 'Completed', 8, 3),         -- Taylor's payment completed for Coco
    -> ('Debit Card', 950.00, 'Failed', 9, 5),         -- Taylor's payment completed for Coco
    -> ('Credit Card', 950.00, 'Completed', 9, 5);         -- Taylor's payment completed for Coco
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Output: 6 rows inserted into `Payment`.

After running the DML statements, the following SELECT queries show the populated tables:

Query 1: Pet Details

SQL

```
mysql> SELECT
    ->     p.pet_id,
    ->     p.name AS Pet_Name,
    ->     pt.species AS Species,
    ->     pt.breed AS Breed,
    ->     p.age AS Age,
    ->     p.gender AS Gender,
    ->     p.status AS Status,
    ->     s.name AS Shelter_Name
    -> FROM Pet p
    -> JOIN PetType pt ON p.type_id = pt.type_id
    -> JOIN Shelter s ON p.shelter_id = s.shelter_id
    -> ORDER BY pt.species, p.name;
```

Result:

```
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
| pet_id | Pet_Name | Species | Breed           | Age  | Gender | Status       | Shelter_Name             |
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
|      7 | Coffee   | Cat     | Calico          |    2 | F      | Adopted      | Paws Shelter             |
|      3 | Luna     | Cat     | Calico          |    1 | F      | Available    | PES EC Campus, Bangalore |
|      5 | Zoomie   | Cat     | Persian         |    2 | M      | Available    | Paws Shelter             |
|      2 | Milo     | Dog     | German Shepherd |    2 | M      | Adopted      | Paws Shelter             |
|      1 | Yuki     | Dog     | Shih Tzu        |    3 | M      | Adopted      | PES EC Campus, Bangalore |
|      6 | Coco     | Parrot  | African Grey    |    1 | M      | Available    | Happy Tails Home         |
|      4 | Snowball | Rabbit  | Holland Lop     |    4 | F      | Medical Hold | Happy Tails Home         |
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
7 rows in set (0.00 sec)
```

Query 2: Payment Details

SQL

```
mysql> SELECT
    ->     u.user_id,
    ->     u.name AS User_Name,
    ->     u.email AS Email,
    ->     p.method AS Payment_Method,
    ->     p.amount AS Amount,
    ->     p.status AS Payment_Status,
    ->     p.date AS Payment_Date
    -> FROM Payment p
    -> JOIN User u ON p.user_id = u.user_id
    -> ORDER BY p.date DESC;
```

Result:

```
+---------+-----------------+-------------------------+----------------+---------+----------------+---------------------+
| user_id | User_Name       | Email                   | Payment_Method | Amount  | Payment_Status | Payment_Date        |
+---------+-----------------+-------------------------+----------------+---------+----------------+---------------------+
|       3 | Angad Bhalla    | angad@petsystem.com     | Credit Card    | 1500.00 | Completed      | 2025-11-14 13:32:17 |
|       4 | Suchitra Shankar | suchitra@petsystem.com | UPI            |  200.00 | Pending        | 2025-11-14 13:32:17 |
|       8 | Taylor Swift    | taylor@petsystem.com    | UPI            |  500.00 | Pending        | 2025-11-14 13:32:17 |
|       8 | Taylor Swift    | taylor@petsystem.com    | PayPal         |  800.00 | Completed      | 2025-11-14 13:32:17 |
|       9 | Harry Styles    | harry@petsystem.com     | Credit Card    |  950.00 | Completed      | 2025-11-14 13:32:17 |
+---------+-----------------+-------------------------+----------------+---------+----------------+---------------------+
5 rows in set (0.00 sec)
```

7. QUERIES

7.1 SIMPLE QUERY WITH GROUP BY, AGGREGATE

Query 1: List all pets with their species, breed, and shelter name. This query joins three tables (Pet, PetType, Shelter) to provide a comprehensive list of all pets.

```
mysql> SELECT
    ->     p.pet_id,
    ->     p.name AS Pet_Name,
    ->     pt.species AS Species,
    ->     pt.breed AS Breed,
    ->     p.age AS Age,
    ->     p.gender AS Gender,
    ->     p.status AS Status,
    ->     s.name AS Shelter_Name
    -> FROM Pet p
    -> JOIN PetType pt ON p.type_id = pt.type_id
    -> JOIN Shelter s ON p.shelter_id = s.shelter_id
    -> ORDER BY pt.species, p.name;
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
| pet_id | Pet_Name | Species | Breed           | Age  | Gender | Status       | Shelter_Name             |
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
|      7 | Coffee   | Cat     | Calico          |    2 | F      | Adopted      | Paws Shelter             |
|      3 | Luna     | Cat     | Calico          |    1 | F      | Available    | PES EC Campus, Bangalore |
|      5 | Zoomie   | Cat     | Persian         |    2 | M      | Available    | Paws Shelter             |
|      2 | Milo     | Dog     | German Shepherd |    2 | M      | Adopted      | Paws Shelter             |
|      1 | Yuki     | Dog     | Shih Tzu        |    3 | M      | Adopted      | PES EC Campus, Bangalore |
|      6 | Coco     | Parrot  | African Grey    |    1 | M      | Available    | Happy Tails Home         |
|      4 | Snowball | Rabbit  | Holland Lop     |    4 | F      | Medical Hold | Happy Tails Home         |
+--------+----------+---------+-----------------+------+--------+--------------+--------------------------+
7 rows in set (0.00 sec)
```

Query 2: List all payments with user details, ordered by date. This query joins Payment and User to show who made which payment and when.

```
mysql> SELECT
    ->     u.user_id,
    ->     u.name AS User_Name,
    ->     u.email AS Email,
    ->     p.method AS Payment_Method,
    ->     p.amount AS Amount,
    ->     p.status AS Payment_Status,
    ->     p.date AS Payment_Date
    -> FROM Payment p
    -> JOIN User u ON p.user_id = u.user_id
    -> ORDER BY p.date DESC;
+---------+-----------------+-----------------------+----------------+---------+----------------+---------------------+
| user_id | User_Name       | Email                 | Payment_Method | Amount  | Payment_Status | Payment_Date        |
+---------+-----------------+-----------------------+----------------+---------+----------------+---------------------+
|       3 | Angad Bhalla    | angad@petsystem.com   | Credit Card    | 1500.00 | Completed      | 2025-11-17 14:47:24 |
|       4 | Suchitra Shankar | suchitra@petsystem.com | UPI            |  200.00 | Pending        | 2025-11-17 14:47:24 |
|       8 | Taylor Swift    | taylor@petsystem.com  | UPI            |  500.00 | Pending        | 2025-11-17 14:47:24 |
|       8 | Taylor Swift    | taylor@petsystem.com  | PayPal         |  800.00 | Completed      | 2025-11-17 14:47:24 |
|       9 | Harry Styles    | harry@petsystem.com   | Credit Card    |  950.00 | Completed      | 2025-11-17 14:47:24 |
+---------+-----------------+-----------------------+----------------+---------+----------------+---------------------+
5 rows in set (0.00 sec)
```

Query 3: Count the number of pets of each species in each shelter. This query uses JOIN, GROUP BY, and the COUNT() aggregate function to generate a summary report.

```
mysql> SELECT
    ->     s.name AS Shelter_Name,
    ->     pt.species AS Species,
    ->     COUNT(p.pet_id) AS Number_of_Pets
    -> FROM Pet p
    -> JOIN Shelter s ON p.shelter_id = s.shelter_id
    -> JOIN PetType pt ON p.type_id = pt.type_id
    -> GROUP BY s.name, pt.species
    -> ORDER BY Shelter_Name, Species;
+-------------------------+---------+----------------+
| Shelter_Name            | Species | Number_of_Pets |
+-------------------------+---------+----------------+
| Happy Tails Home        | Parrot  |              1 |
| Happy Tails Home        | Rabbit  |              1 |
| Paws Shelter            | Cat     |              2 |
| Paws Shelter            | Dog     |              1 |
| PES EC Campus, Bangalore | Cat     |              1 |
| PES EC Campus, Bangalore | Dog     |              1 |
+-------------------------+---------+----------------+
6 rows in set (0.01 sec)
```

7.2 UPDATE OPERATION

Query: Update the gender of the pet named 'Zoomie' to 'M'. This DML command modifies existing data in the Pet table.

```
mysql> UPDATE Pet
    -> SET gender = 'M'
    -> WHERE name = 'Zoomie';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

7.3 DELETE OPERATION

Query 1: Delete all payment records that have a status of 'Failed'. This command removes rows from the Payment table based on a condition.

```
mysql> DELETE FROM Payment
    -> WHERE status = 'Failed';
Query OK, 0 rows affected (0.00 sec)
```

Query 2: Delete a specific pet type (e.g., 'Beagle', type_id = 2). This command removes a row from the PetType table. (Note: This might fail if a pet still references this type and ON DELETE behavior is not set to SET NULL or CASCADE).

```
mysql> DELETE FROM PetType
    -> WHERE type_id = 2;
Query OK, 0 rows affected (0.00 sec)
```

7.4 CORRELATED QUERY

Query: Find all pets that are older than the average age of pets *of the same species*. A correlated subquery is used here, where the inner query (calculating average age) is re-executed for every row processed by the outer query, using the type_id from the outer query.

```
mysql> SELECT
    ->     p1.name AS Pet_Name,
    ->     pt.species,
    ->     p1.age AS Pet_Age,
    ->     (SELECT AVG(p2.age) FROM Pet p2 WHERE p2.type_id = p1.type_id) AS Avg_Species_Age
    -> FROM Pet p1
    -> JOIN PetType pt ON p1.type_id = pt.type_id
    -> WHERE p1.age > (SELECT AVG(p2.age)
    ->                 FROM Pet p2
    ->                 WHERE p2.type_id = p1.type_id);
+----------+---------+---------+-----------------+
| Pet_Name | species | Pet_Age | Avg_Species_Age |
+----------+---------+---------+-----------------+
| Coffee   | Cat     |       2 |          1.5000 |
+----------+---------+---------+-----------------+
1 row in set (0.01 sec)
```

7.5 NESTED QUERY

Query: Find the names and emails of all users who have successfully adopted a 'Dog'. This query uses multiple levels of nesting (subqueries in WHERE clauses) to filter data.

- The innermost query finds type_id for 'Dog'.
- The next query finds pet_ids that match those types.
- The next query finds user_ids who have 'Approved' applications for those pets.
- The outermost query retrieves the User details.

```
mysql> SELECT name, email
    -> FROM User
    -> WHERE user_id IN (
    ->     SELECT user_id
    ->     FROM AdoptionApplication
    ->     WHERE status = 'Approved'
    ->     AND pet_id IN (
    ->         SELECT pet_id
    ->         FROM Pet
    ->         WHERE type_id IN (
    ->             SELECT type_id
    ->             FROM PetType
    ->             WHERE species = 'Dog'
    ->         )
    ->     )
    -> );
+--------------+---------------------+
| name         | email               |
+--------------+---------------------+
| Angad Bhalla | angad@petsystem.com |
| Harry Styles | harry@petsystem.com |
+--------------+---------------------+
2 rows in set (0.01 sec)
```

8. STORED PROCEDURES, FUNCTIONS AND TRIGGERS

Advanced database logic is implemented using stored procedures, functions, and triggers to enforce business rules and automate tasks.

8.1 Stored Procedures

Stored Procedure: `ApproveApplication` This is the most complex procedure. When a shelter worker approves an application, it:

1. Validates the application and pet.
2. Updates the application status to 'Approved'.
3. Updates the associated payment to 'Completed'.
4. Updates the pet's status to 'Adopted'.
5. Calls *another* procedure (`AutoRejectOtherApplications`) to deny all other pending applications for that same pet.

SQL

```
mysql> CREATE PROCEDURE ApproveApplication (
    ->     IN p_app_id INT,
    ->     IN p_worker_id INT
    -> )
    -> BEGIN
    ->     DECLARE v_pet_id INT;
    ->     DECLARE v_pet_status VARCHAR(50);
    ->
    ->     SELECT a.pet_id INTO v_pet_id
    ->     FROM pet_adoption_db.AdoptionApplication a
    ->     WHERE a.adopt_app_id = p_app_id
    ->     LIMIT 1;
    ->
    ->     IF v_pet_id IS NULL THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Adoption application not found.';
    ->     END IF;
    ->
    ->     SELECT p.status INTO v_pet_status
    ->     FROM pet_adoption_db.Pet p
    ->     WHERE p.pet_id = v_pet_id
    ->     LIMIT 1;
    ->
    ->     IF v_pet_status = 'Adopted' THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'This pet has already been adopted.';
    ->     END IF;
    ->
    ->     UPDATE pet_adoption_db.AdoptionApplication a
    ->     SET a.status = 'Approved',
    ->         a.approved_by = p_worker_id
    ->     WHERE a.adopt_app_id = p_app_id;
    ->
    ->     UPDATE pet_adoption_db.Payment pay
    ->     SET pay.status = 'Completed'
    ->     WHERE pay.adoption_app_id = p_app_id AND pay.status <> 'Completed';
    ->
    ->     UPDATE pet_adoption_db.Pet
    ->     SET status = 'Adopted'
    ->     WHERE pet_id = v_pet_id;
    ->
    ->     CALL pet_adoption_db.AutoRejectOtherApplications(v_pet_id, p_app_id);
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

Stored Procedure: `AddAdoptionApplication` This procedure ensures a user cannot apply for a pet that is already adopted. It checks the pet's status *before* inserting the new application.

SQL

```
mysql> CREATE PROCEDURE AddAdoptionApplication (
    ->     IN p_user_id INT,
    ->     IN p_pet_id INT,
    ->     IN p_reason TEXT
    -> )
    -> BEGIN
    ->     DECLARE v_status VARCHAR(50);
    ->     DECLARE v_app_id INT;
    ->
    ->     SELECT p.status INTO v_status
    ->     FROM pet_adoption_db.Pet p
    ->     WHERE p.pet_id = p_pet_id
    ->     LIMIT 1;
    ->
    ->     IF v_status = 'Adopted' THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot apply: pet already adopted.';
    ->     END IF;
    ->
    ->     -- 1. Insert the new Adoption Application
    ->     INSERT INTO pet_adoption_db.AdoptionApplication (status, reason, approved_by, pet_id, user_id)
    ->     VALUES ('Pending', p_reason, NULL, p_pet_id, p_user_id);
    ->
    ->     SET v_app_id = LAST_INSERT_ID();
    ->
    ->     -- 2. Insert the initial Payment record
    ->     INSERT INTO pet_adoption_db.Payment (method, amount, status, user_id, adoption_app_id)
    ->     VALUES (
    ->         'Cash',
    ->         500.00,
    ->         'Pending',
    ->         p_user_id,
    ->         v_app_id
    ->     );
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

Stored Procedure: `RemoveApplication`

SQL

```
mysql> CREATE PROCEDURE RejectApplication (
    ->     IN p_app_id INT,
    ->     IN p_reason TEXT
    -> )
    -> BEGIN
    ->     UPDATE pet_adoption_db.AdoptionApplication
    ->     SET status = 'Denied',
    ->         reason = p_reason
    ->     WHERE adopt_app_id = p_app_id;
    ->
    ->     UPDATE pet_adoption_db.Payment
    ->     SET status = 'Refunded'
    ->     WHERE adoption_app_id = p_app_id AND status <> 'Refunded';
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

Stored Procedure: `AutoRejectOtherApplications`

SQL

```
mysql> CREATE PROCEDURE AutoRejectOtherApplications (
    ->     IN p_pet_id INT,
    ->     IN p_exclude_app_id INT
    -> )
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE v_app_id INT;
    ->
    ->     DECLARE app_cursor CURSOR FOR
    ->         SELECT adopt_app_id
    ->         FROM pet_adoption_db.AdoptionApplication
    ->         WHERE pet_id = p_pet_id
    ->           AND status = 'Pending'
    ->           AND adopt_app_id <> p_exclude_app_id;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     OPEN app_cursor;
    ->     app_loop: LOOP
    ->         FETCH app_cursor INTO v_app_id;
    ->         IF done THEN
    ->             LEAVE app_loop;
    ->         END IF;
    ->         CALL pet_adoption_db.RejectApplication(
    ->             v_app_id,
    ->             'Automatically rejected since the pet has been adopted.'
    ->         );
    ->     END LOOP;
    ->     CLOSE app_cursor;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

8.2 Triggers

Trigger: `trg_log_pet_status_change` This trigger automatically fires *before* any update on the `Pet` table. If the `status` column is being changed, it inserts a record into the `PetStatusLog` table, creating an audit trail.

SQL

```
mysql> CREATE TRIGGER trg_log_pet_status_change
    -> BEFORE UPDATE ON pet_adoption_db.Pet
    -> FOR EACH ROW
    -> BEGIN
    ->     IF OLD.status <> NEW.status THEN
    ->         INSERT INTO pet_adoption_db.PetStatusLog (pet_id, old_status, new_status)
    ->         VALUES (OLD.pet_id, OLD.status, NEW.status);
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

Trigger: `trg_prevent_duplicate_adoption` This trigger fires *before* an `INSERT` on the `AdoptionApplication` table. It checks the status of the pet being applied for. If the pet is already 'Adopted', the trigger raises an error and cancels the `INSERT` operation.

SQL

```
mysql> CREATE TRIGGER trg_prevent_duplicate_adoption
    -> BEFORE INSERT ON pet_adoption_db.AdoptionApplication
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE pet_stat VARCHAR(50);
    ->     SELECT p.status INTO pet_stat
    ->     FROM pet_adoption_db.Pet p
    ->     WHERE p.pet_id = NEW.pet_id
    ->     LIMIT 1;
    ->
    ->     IF pet_stat = 'Adopted' THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'This pet has already been adopted.';
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

Trigger: `trg_update_pet_status_on_approval` This trigger fires after an UPDATE on the AdoptionApplication table. When an application's status changes to 'Approved', it automatically updates the related pet's status to 'Adopted' and marks the corresponding payment as 'Completed'. This keeps the data consistent even during manual updates.

SQL

```
mysql> CREATE TRIGGER trg_update_pet_status_on_approval
    -> AFTER UPDATE ON pet_adoption_db.AdoptionApplication
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.status = 'Approved' AND OLD.status <> 'Approved' THEN
    ->         UPDATE pet_adoption_db.Pet
    ->         SET status = 'Adopted'
    ->         WHERE pet_id = NEW.pet_id;
    ->
    ->         UPDATE pet_adoption_db.Payment
    ->         SET status = 'Completed'
    ->         WHERE adoption_app_id = NEW.adopt_app_id AND status <> 'Completed';
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

8.3 Functions

Function: CountAvailablePets

This function takes a shelter's ID as input and returns the total number of pets in that shelter whose status is `'Available'`. It helps track real-time availability across shelters.

SQL

```
mysql> CREATE FUNCTION CountAvailablePets (p_shelter_id INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE pet_count INT;
    ->     SELECT COUNT(*) INTO pet_count
    ->     FROM pet_adoption_db.Pet
    ->     WHERE shelter_id = p_shelter_id AND status = 'Available';
    ->     RETURN pet_count;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

Function: TotalAdoptionsByUser

This function takes a user's ID and returns how many pets that user has successfully adopted (i.e., applications marked as `'Approved'`).

SQL

```
mysql> CREATE FUNCTION TotalAdoptionsByUser (p_user_id INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE total_adopted INT;
    ->     SELECT COUNT(*) INTO total_adopted
    ->     FROM pet_adoption_db.AdoptionApplication
    ->     WHERE user_id = p_user_id AND status = 'Approved';
    ->     RETURN total_adopted;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

Function: AvgPetAgeInShelter

This function calculates the average age of all pets (in any status) in a given shelter. It is useful for analytics and reports.

SQL

```
mysql> CREATE FUNCTION AvgPetAgeInShelter (p_shelter_id INT)
    -> RETURNS DECIMAL(5,2)
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE avg_age DECIMAL(5,2);
    ->     SELECT AVG(age) INTO avg_age
    ->     FROM pet_adoption_db.Pet
    ->     WHERE shelter_id = p_shelter_id;
    ->     RETURN avg_age;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)
```

9. FRONT END DEVELOPMENT

While the core of this project is the backend database, a front-end application would be the user-facing component that interacts with the database to provide a functional system. No front-end was implemented for this project, but this section describes a potential design.

Proposed Technology Stack:

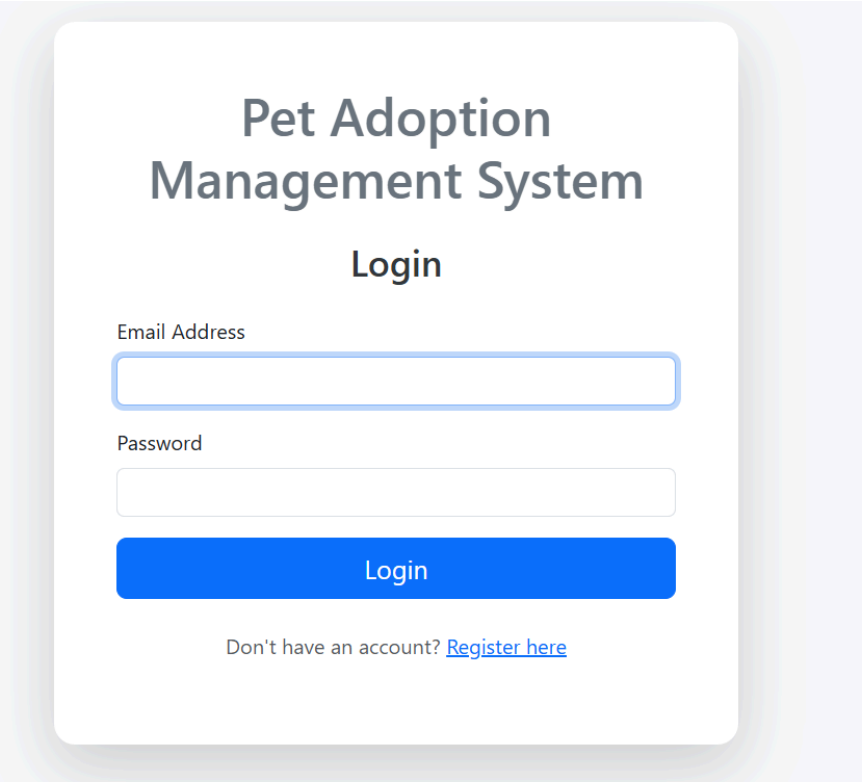
- **Backend:** A Python framework like **Flask** or **Django**, or a **Node.js** backend with **Express**. This layer would host the API.
- **Frontend:** A modern JavaScript framework like **React** or **Vue.js** to build a dynamic and responsive single-page application (SPA).
- **Connector:** A library like `mysql-connector-python` (for Flask/Django) or `mysql2` (for Node.js) to communicate between the API and the MySQL database.

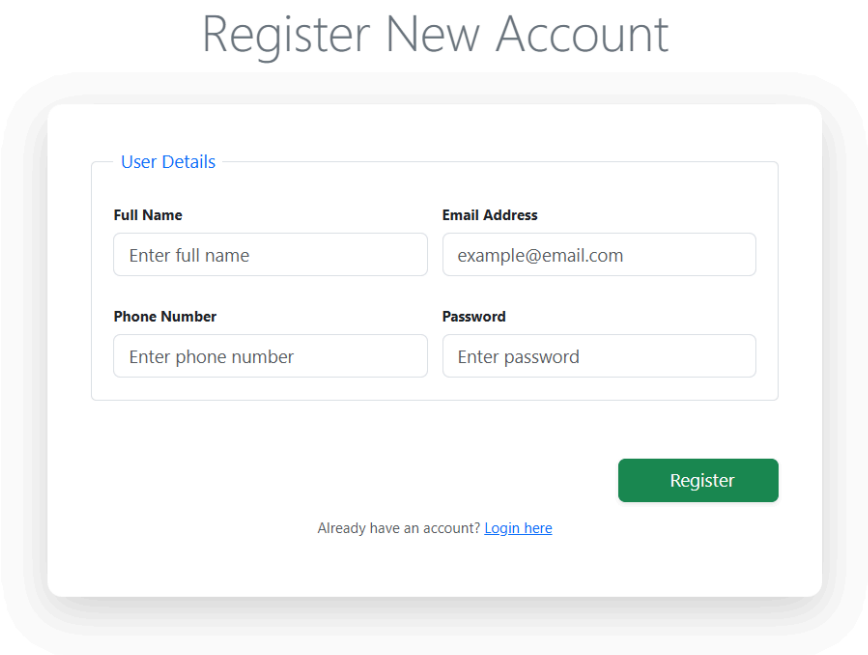
Key Functional Components:

1. **User Authentication and Dashboards:**
    - Users would log in. The backend would check their `role` from the `User` table.
    - **Adopter Dashboard:** Shows the user's past adoptions (from `TotalAdoptionsByUser` function) and pending applications.
    - **Shelter Worker Dashboard:** Shows all pending applications for their shelter. This page would allow them to `CALL ApproveApplication` or `CALL RejectApplication` by clicking buttons.
    - **Admin Dashboard:** Allows `CALL AddUser` to create new worker accounts or `CALL RegisterPet` to add new animals to the system.
2. **Pet Listing Page:**
    - This public page would display all available pets.
    - It would be populated by querying the `AvailablePets` view, providing a simple and secure way to show pets without exposing underlying table structures.
    - Filters could be added to query based on species, breed, or shelter (using the `CountAvailablePets` function to show counts).

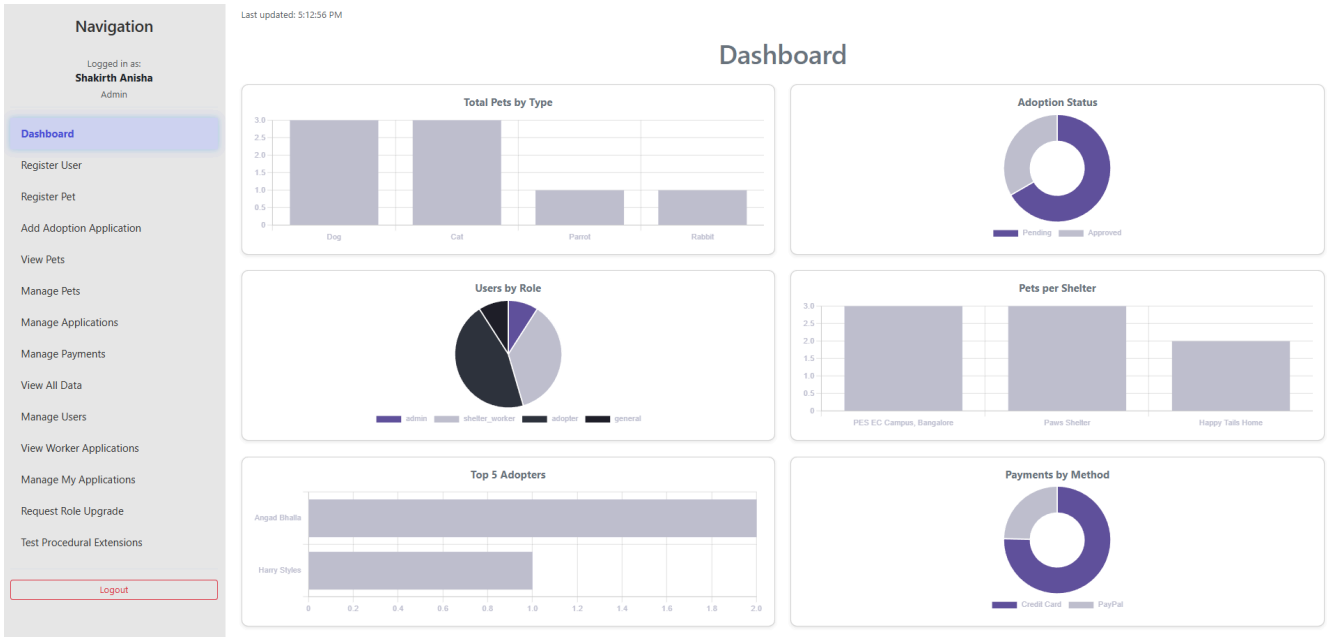
3. **Adoption Application Form:**
    - When a user clicks "Adopt" on a pet, a form would appear.
    - Submitting this form would `CALL AddAdoptionApplication` with the `user_id`, `pet_id`, and the reason text.
    - The trigger `trg_prevent_duplicate_adoption` would provide immediate feedback (an error message) if the user tries to apply for an already-adopted pet.

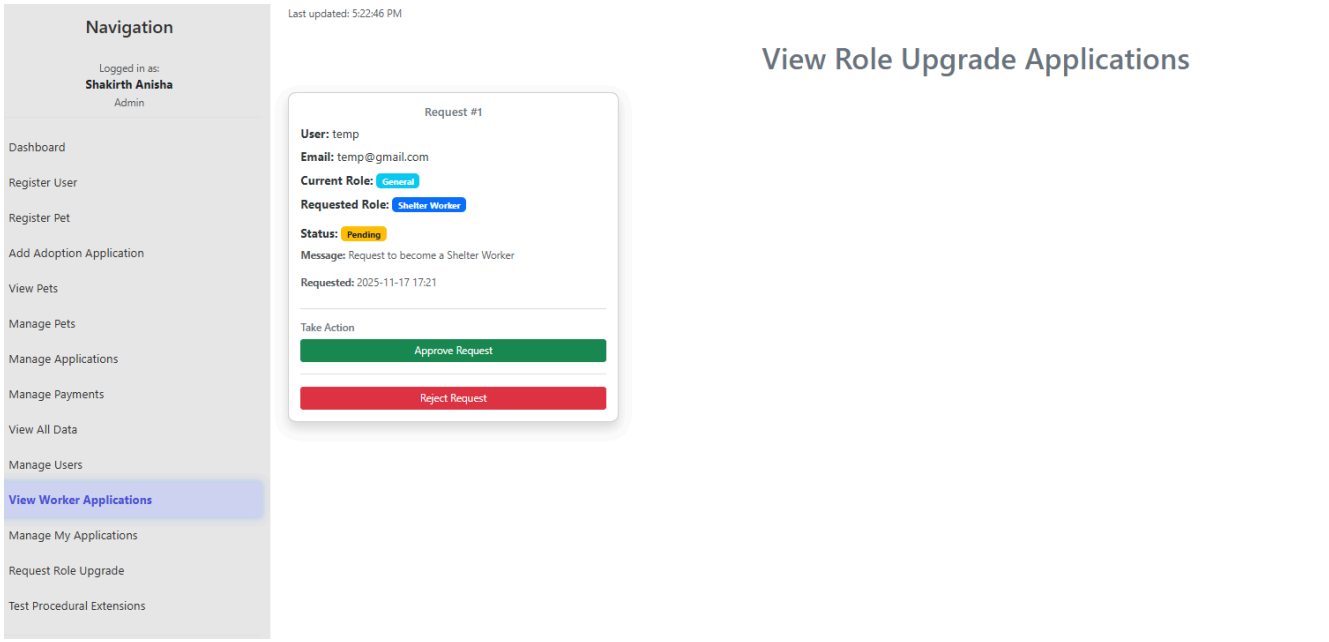
Login

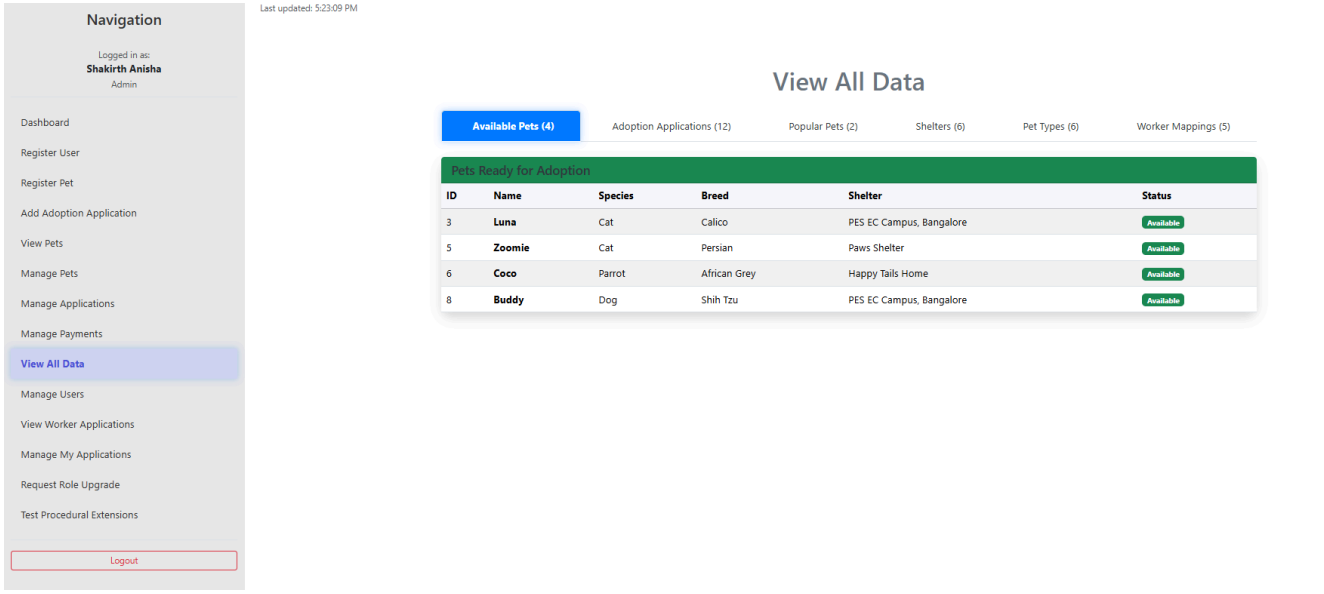

Pet Adoption Management System
Login
Email Address
Password
Login
Don't have an account? Register here

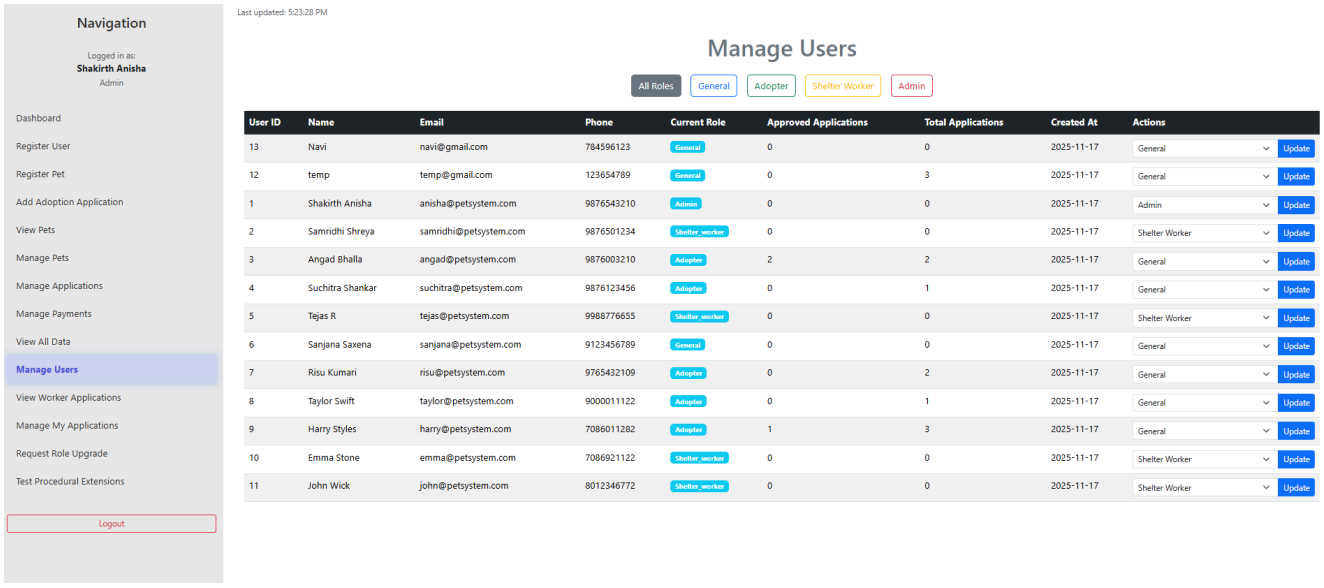

Register New Account
User Details
Full Name
Enter full name
Email Address
example@email.com
Phone Number
Enter phone number
Password
Enter password
Register
Already have an account? Login here

Admin:

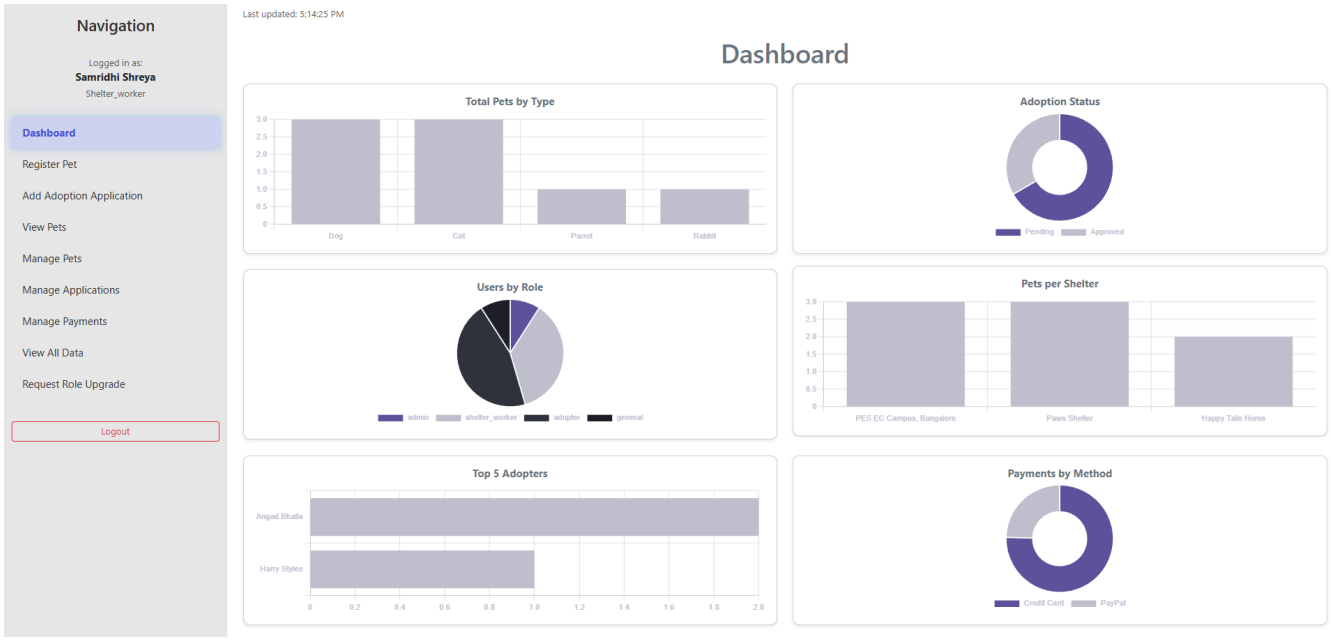

Navigation
Logged in as:
Shakirth Anisha
Admin
Dashboard
Register User
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Manage Users
View Worker Applications
Manage My Applications
Request Role Upgrade
Test Procedural Extensions
Logout
Last updated: 5:12:56 PM
Dashboard
Total Pets by Type
Dog
Cat
Parrot
Rabbit
Adoption Status
Pending
Approved
Users by Role
admin
shelter_worker
adopter
general
Pets per Shelter
PES EC Campus, Bangalore
Paws Shelter
Happy Tails Home
Top 5 Adopters
Angad Bhalla
Harry Styles
Payments by Method
Credit Card
PayPal

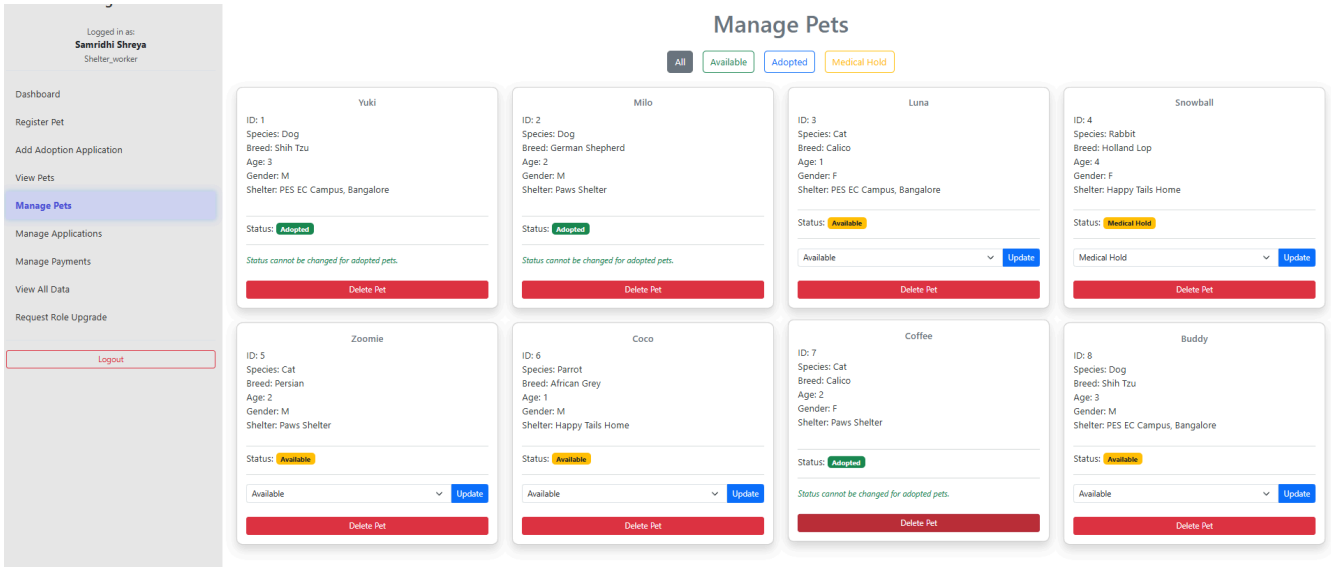

Navigation
Logged in as:
Shakirth Anisha
Admin
Dashboard
Register User
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Manage Users
View Worker Applications
Manage My Applications
Request Role Upgrade
Test Procedural Extensions
Last updated: 5:22:46 PM
View Role Upgrade Applications
Request #1
User: temp
Email: temp@gmail.com
Current Role: General
Requested Role: Shelter Worker
Status: Pending
Message: Request to become a Shelter Worker
Requested: 2025-11-17 17:21
Take Action
Approve Request
Reject Request

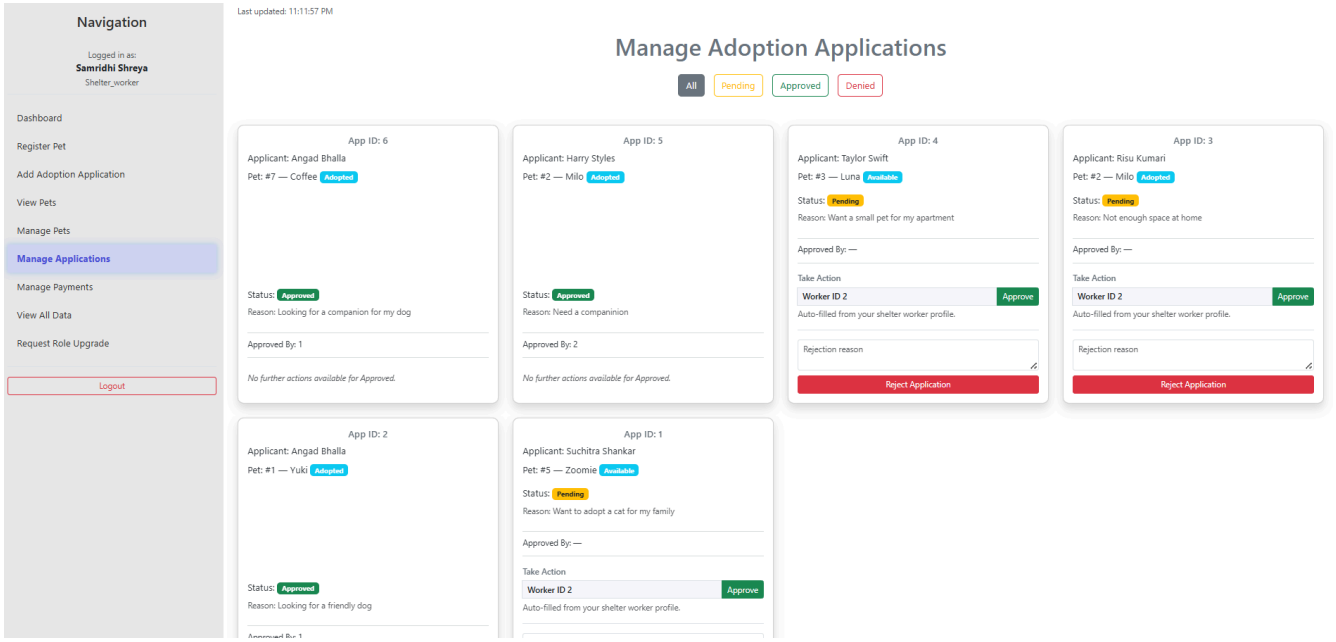

Navigation
Logged in as:
Shakirth Anisha
Admin
Dashboard
Register User
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Manage Users
View Worker Applications
Manage My Applications
Request Role Upgrade
Test Procedural Extensions
Logout
Last updated: 5:23:09 PM
View All Data
Available Pets (4)
Adoption Applications (12)
Popular Pets (2)
Shelters (6)
Pet Types (6)
Worker Mappings (5)
Pets Ready for Adoption
ID Name Species Breed Shelter Status
3 Luna Cat Calico PES EC Campus, Bangalore Available
5 Zoomie Cat Persian Paws Shelter Available
6 Coco Parrot African Grey Happy Tails Home Available
8 Buddy Dog Shih Tzu PES EC Campus, Bangalore Available

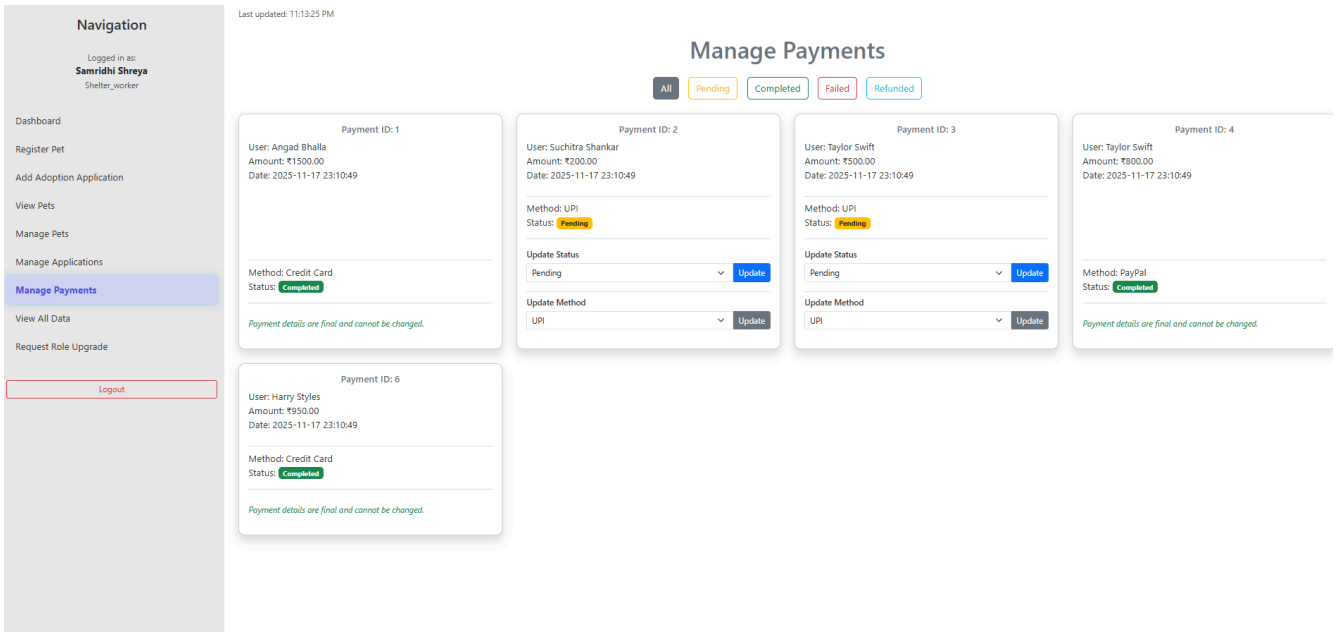

Navigation
Logged in as:
Shakirth Anisha
Admin
Dashboard
Register User
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Manage Users
View Worker Applications
Manage My Applications
Request Role Upgrade
Test Procedural Extensions
Logout
Last updated: 5:23:28 PM
Manage Users
All Roles
General
Adopter
Shelter Worker
Admin
User ID Name Email Phone Current Role Approved Applications Total Applications Created At Actions
13 Navi navi@gmail.com 784596123 General 0 0 2025-11-17 General Update
12 temp temp@gmail.com 123654789 General 0 3 2025-11-17 General Update
1 Shakirth Anisha anisha@petsystem.com 9876543210 Admin 0 0 2025-11-17 Admin Update
2 Samridhi Shreya samridhi@petsystem.com 9876501234 Shelter_worker 0 0 2025-11-17 Shelter Worker Update
3 Angad Bhalla angad@petsystem.com 9876003210 Adopter 2 2 2025-11-17 General Update
4 Suchitra Shankar suchitra@petsystem.com 9876123456 Adopter 0 1 2025-11-17 General Update
5 Tejas R tejas@petsystem.com 9988776655 Shelter_worker 0 0 2025-11-17 Shelter Worker Update
6 Sanjana Saxena sanjana@petsystem.com 9123456789 General 0 0 2025-11-17 General Update
7 Risu Kumari risu@petsystem.com 9765432109 Adopter 0 2 2025-11-17 General Update
8 Taylor Swift taylor@petsystem.com 9000011122 Adopter 0 1 2025-11-17 General Update
9 Harry Styles harry@petsystem.com 7086011282 Adopter 1 3 2025-11-17 General Update
10 Emma Stone emma@petsystem.com 7086921122 Shelter_worker 0 0 2025-11-17 Shelter Worker Update
11 John Wick john@petsystem.com 8012346772 Shelter_worker 0 0 2025-11-17 Shelter Worker Update

Shelter Worker:

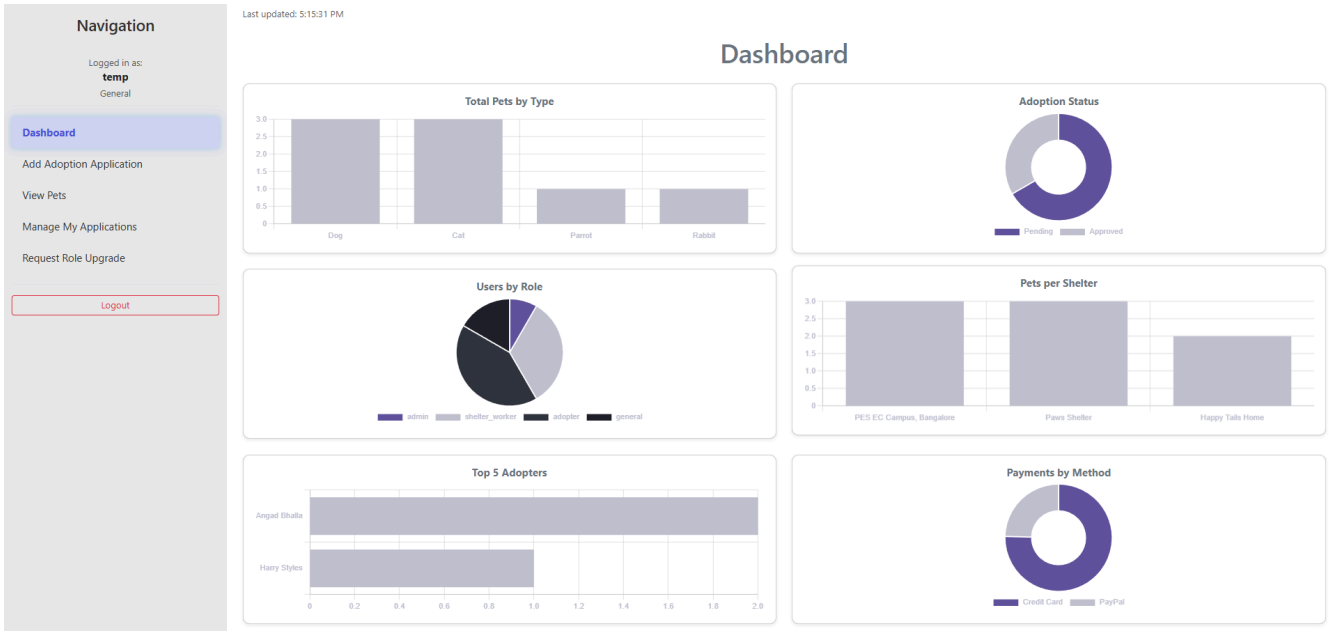


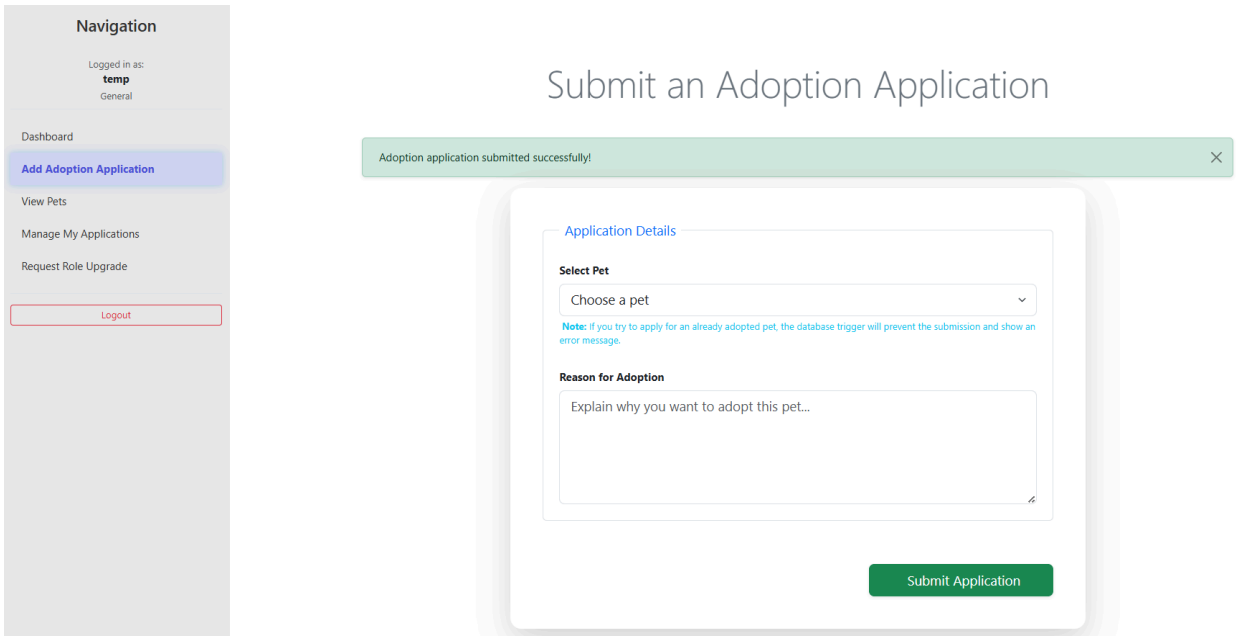


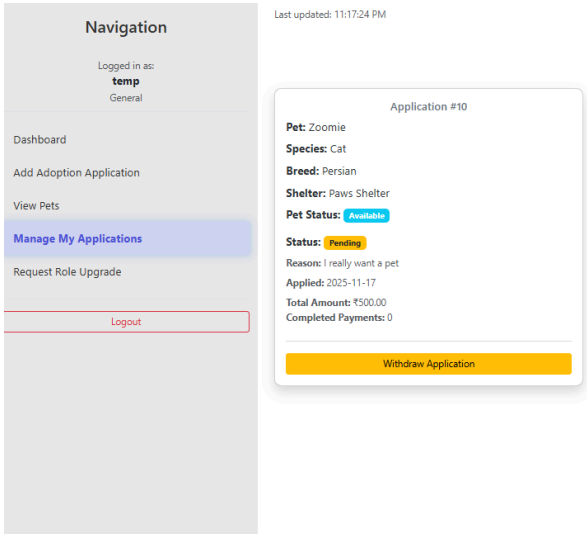

Navigation
Logged in as:
Samridhi Shreya
Shelter_worker
Dashboard
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Request Role Upgrade
Logout
Last updated: 11:11:57 PM
Manage Adoption Applications
All
Pending
Approved
Denied
App ID: 6
Applicant: Angad Bhalla
Pet: #7 — Coffee Adopted
Status: Approved
Reason: Looking for a companion for my dog
Approved By: 1
No further actions available for Approved.
App ID: 5
Applicant: Harry Styles
Pet: #2 — Milo Adopted
Status: Approved
Reason: Need a companinion
Approved By: 2
No further actions available for Approved.
App ID: 4
Applicant: Taylor Swift
Pet: #3 — Luna Available
Status: Pending
Reason: Want a small pet for my apartment
Approved By: —
Take Action
Worker ID 2
Approve
Auto-filled from your shelter worker profile.
Rejection reason
Reject Application
App ID: 3
Applicant: Risu Kumari
Pet: #2 — Milo Adopted
Status: Pending
Reason: Not enough space at home
Approved By: —
Take Action
Worker ID 2
Approve
Auto-filled from your shelter worker profile.
Rejection reason
Reject Application
App ID: 2
Applicant: Angad Bhalla
Pet: #1 — Yuki Adopted
Status: Approved
Reason: Looking for a friendly dog
App ID: 1
Applicant: Suchitra Shankar
Pet: #5 — Zoomie Available
Status: Pending
Reason: Want to adopt a cat for my family
Approved By: —
Take Action
Worker ID 2
Approve
Auto-filled from your shelter worker profile.

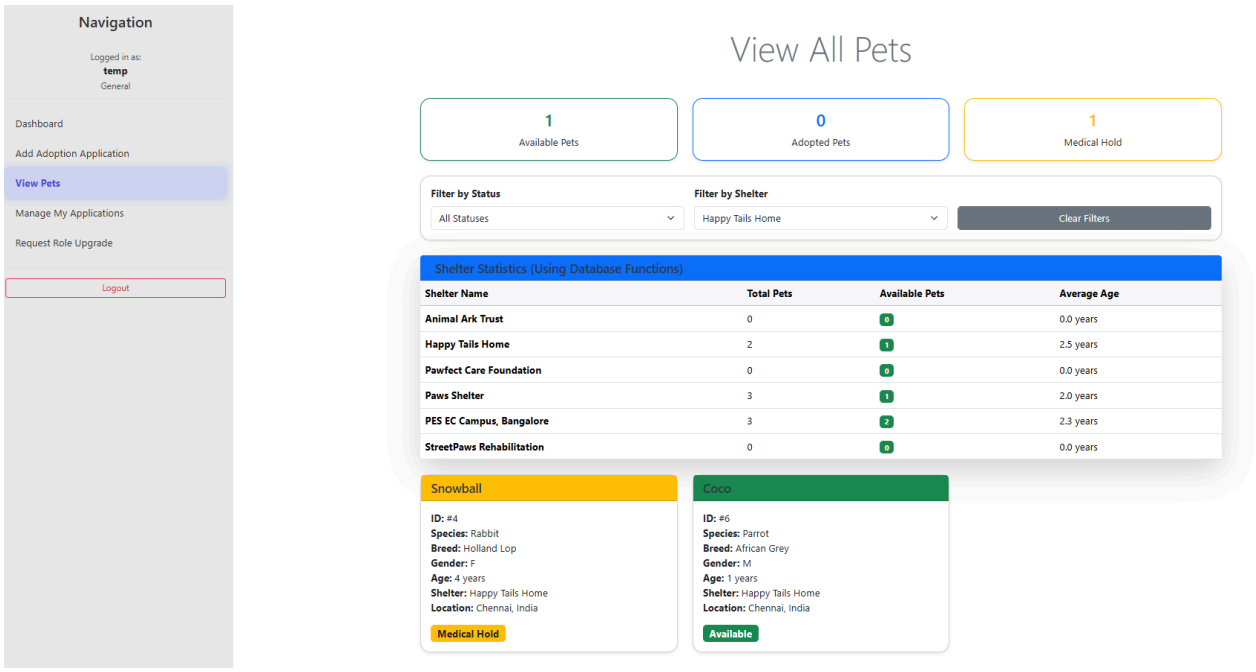

Navigation
Logged in as:
Samridhi Shreya
Shelter_worker
Dashboard
Register Pet
Add Adoption Application
View Pets
Manage Pets
Manage Applications
Manage Payments
View All Data
Request Role Upgrade
Logout
Last updated: 11:13:25 PM
Manage Payments
All
Pending
Completed
Failed
Refunded
Payment ID: 1
User: Angad Bhalla
Amount: ₹1500.00
Date: 2025-11-17 23:10:49
Method: Credit Card
Status: Completed
Payment details are final and cannot be changed.
Payment ID: 2
User: Suchitra Shankar
Amount: ₹200.00
Date: 2025-11-17 23:10:49
Method: UPI
Status: Pending
Update Status
Pending
Update
Update Method
UPI
Update
Payment ID: 3
User: Taylor Swift
Amount: ₹500.00
Date: 2025-11-17 23:10:49
Method: UPI
Status: Pending
Update Status
Pending
Update
Update Method
UPI
Update
Payment ID: 4
User: Taylor Swift
Amount: ₹800.00
Date: 2025-11-17 23:10:49
Method: PayPal
Status: Completed
Payment details are final and cannot be changed.
Payment ID: 6
User: Harry Styles
Amount: ₹950.00
Date: 2025-11-17 23:10:49
Method: Credit Card
Status: Completed
Payment details are final and cannot be changed.

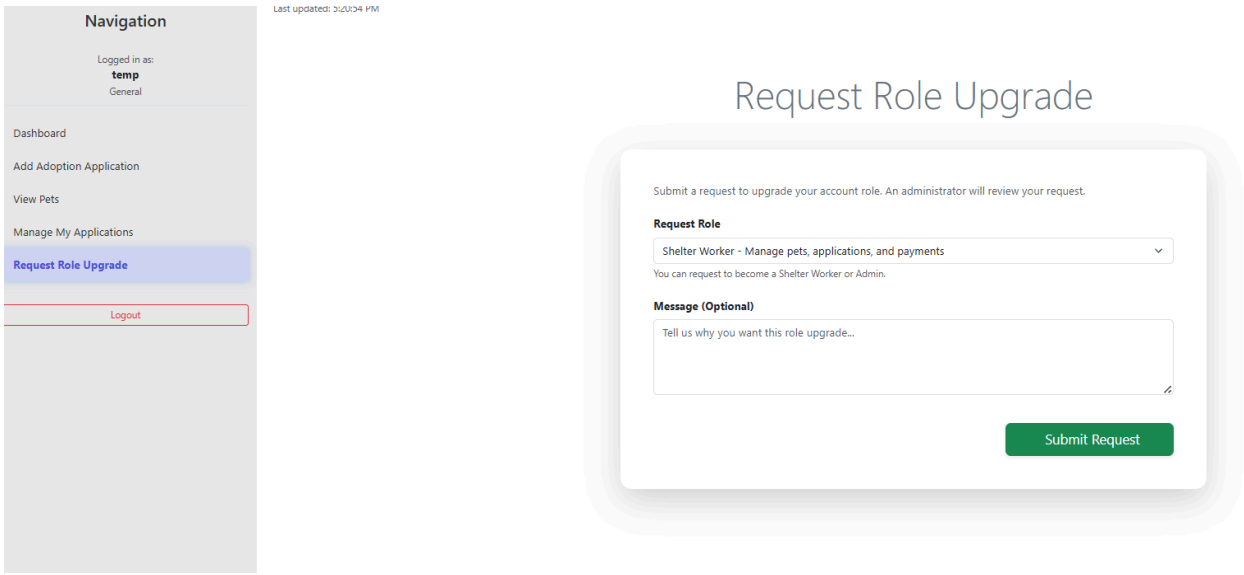
General:


Navigation
Logged in as:
temp
General
Dashboard
Add Adoption Application
View Pets
Manage My Applications
Request Role Upgrade
Logout
Last updated: 5:15:31 PM
Dashboard
Total Pets by Type
Dog
Cat
Parrot
Rabbit
Adoption Status
Pending
Approved
Users by Role
admin
shelter_worker
adopter
general
Pets per Shelter
PES EC Campus, Bangalore
Paws Shelter
Happy Tails Home
Top 5 Adopters
Angad Bhalla
Harry Styles
Payments by Method
Credit Card
PayPal


Navigation
Logged in as:
temp
General
Dashboard
Add Adoption Application
View Pets
Manage My Applications
Request Role Upgrade
Logout
Submit an Adoption Application
Adoption application submitted successfully!
Application Details
Select Pet
Choose a pet
Note: If you try to apply for an already adopted pet, the database trigger will prevent the submission and show an error message.
Reason for Adoption
Explain why you want to adopt this pet...
Submit Application


Navigation
Logged in as:
temp
General
Dashboard
Add Adoption Application
View Pets
Manage My Applications
Request Role Upgrade
Logout
Last updated: 11:17:24 PM
Manage My Applications
Application #10
Pet: Zoomie
Species: Cat
Breed: Persian
Shelter: Paws Shelter
Pet Status: Available
Status: Pending
Reason: I really want a pet
Applied: 2025-11-17
Total Amount: ₹500.00
Completed Payments: 0
Withdraw Application


Navigation
Logged in as:
temp
General
Dashboard
Add Adoption Application
View Pets
Manage My Applications
Request Role Upgrade
Logout
View All Pets
1
Available Pets
0
Adopted Pets
1
Medical Hold
Filter by Status
All Statuses
Filter by Shelter
Happy Tails Home
Clear Filters
Shelter Statistics (Using Database Functions)
Shelter Name
Total Pets
Available Pets
Average Age
Animal Ark Trust
0
0
0.0 years
Happy Tails Home
2
1
2.5 years
Pawfect Care Foundation
0
0
0.0 years
Paws Shelter
3
1
2.0 years
PES EC Campus, Bangalore
3
2
2.3 years
StreetPaws Rehabilitation
0
0
0.0 years
Snowball
ID: #4
Species: Rabbit
Breed: Holland Lop
Gender: F
Age: 4 years
Shelter: Happy Tails Home
Location: Chennai, India
Medical Hold
Coco
ID: #6
Species: Parrot
Breed: African Grey
Gender: M
Age: 1 years
Shelter: Happy Tails Home
Location: Chennai, India
Available


Navigation
Logged in as:
temp
General
Dashboard
Add Adoption Application
View Pets
Manage My Applications
Request Role Upgrade
Logout
Request Role Upgrade
Submit a request to upgrade your account role. An administrator will review your request.
Request Role
Shelter Worker - Manage pets, applications, and payments
You can request to become a Shelter Worker or Admin.
Message (Optional)
Tell us why you want this role upgrade...
Submit Request