

Chapter 1

Introduction

A Pet Shop Management System is a software application designed to manage the day-to-day operations of a pet store. When integrated with a Database Management System (DBMS), it helps store, organize, retrieve, and manage large volumes of structured data related to pets, customers, sales, inventory, and services.

The primary goal of this system is to automate processes, reduce human error, and ensure efficient data handling. By using a DBMS, the application can maintain data integrity, security, and accessibility across various modules.

1.1 Key Features

Pet Inventory Management:

Stores information about available pets (e.g., species, breed, age, health status).

Customer Records:

Keeps track of customer details, preferences, and purchase history.

Sales and Billing:

Automates billing, invoices, and transaction records.

Pet Supplies and Products:

Manages stock levels, suppliers, and pricing for pet-related products.

Appointments and Services:

Records data on grooming, veterinary appointments, or adoption processes.

Employee Management (Optional):

Handles staff roles, salaries, and shift schedules.

1.2 Problem Statement

In many traditional pet shops, operations such as managing pet inventory, handling customer information, recording sales transactions, scheduling services (like grooming or vet appointments), and managing stock of pet supplies are done manually. This leads to several issues, including:

- Data inconsistency and lack of centralized information
- Difficulty in tracking pets, products, and sales history
- Time-consuming customer service and billing processes
- Human errors in inventory or customer management
- Inefficient scheduling of appointments and services

To overcome these challenges, there is a need for an automated Pet Shop Management System that uses a Database Management System (DBMS) to store, manage, and retrieve all necessary data efficiently and accurately.

Chapter 2

Back-end Design

2.1 Conceptual Database Design (ER-DIAGRAM)

An entity-relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database.

The main components of ER model are: entity set and relationship set.

Here are the geometric shapes and their meaning in an ER Diagram

Rectangle : Represents Entity sets.

Ellipses : Attributes.

Diamonds : Relationship set.

Lines : They link attributes to Entity Sets and this to Relationship Set.

Fig no: 2.1 is the ER diagram of “Petshop Management System” with entities pets, animals, birds, pet products, sales details, customer, sold pets

Fig:2.1 ER Diagram

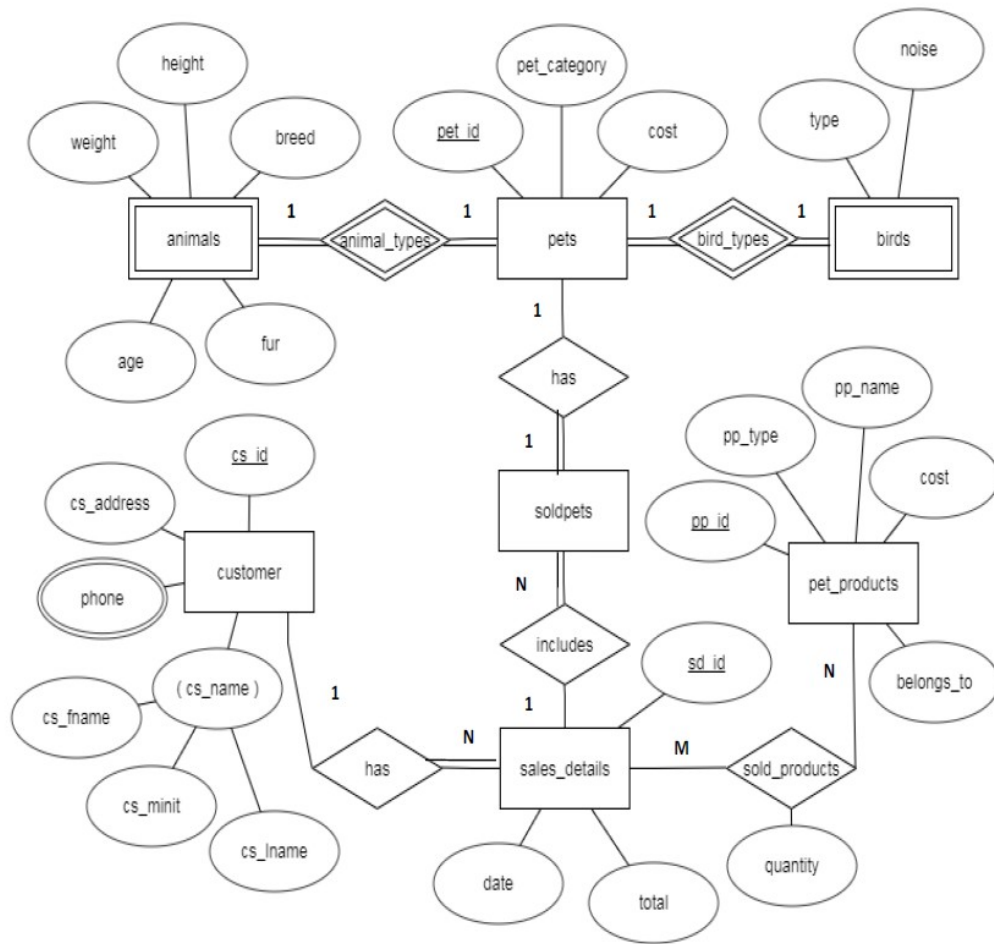


Fig.no 4.1: ER DIAGRAM OF PETSHOP MANAGEMENT SYSTEM

2.2 Logical Database Design(ER MAPPING)

2.2.1 Mapping of regular entities

For each regular (Strong) entity type E in the ER schema. Create a relation R that include all

simple attributes of E.

Regular entities of this database are pets,pet_products,customer,sales_details.

pets

| | | |
|---------------|--------------|------|
| <u>pet_id</u> | pet_category | cost |
|---------------|--------------|------|

pet_products

| | | | | |
|--------------|---------|---------|------|------------|
| <u>pp_id</u> | pp_name | pp_type | cost | belongs_to |
|--------------|---------|---------|------|------------|

customer

| | | | | |
|--------------|----------|----------|----------|------------|
| <u>cs_id</u> | cs_fname | cs-minit | cs_lname | cs-address |
|--------------|----------|----------|----------|------------|

sales_details

| | | |
|--------------|------|-------|
| <u>sd_id</u> | date | total |
|--------------|------|-------|

sold_pets

| | |
|--------------|---------------|
| <u>sd_id</u> | <u>pet_id</u> |
|--------------|---------------|

2.2.2 Mapping of weak entities

A weak entity cannot be used independently as it is dependent on a strong entity type known as its owner entity. Also, the relationship that connects the weak entity to its owner identity is called the identifying relationship.

A weak entity always has a total participation constraint with respect to its identifying relationship because it cannot be identified independently of its owner identity.

A weak entity may have a partial key, which is a list of attributes that identify weak entities related to the same owner entity.

Weak entities of this database are animals and birds

animals

| | | | | | |
|---------------|-------|--------|--------|-----|-----|
| <u>pet_id</u> | breed | weight | height | age | fur |
|---------------|-------|--------|--------|-----|-----|

birds

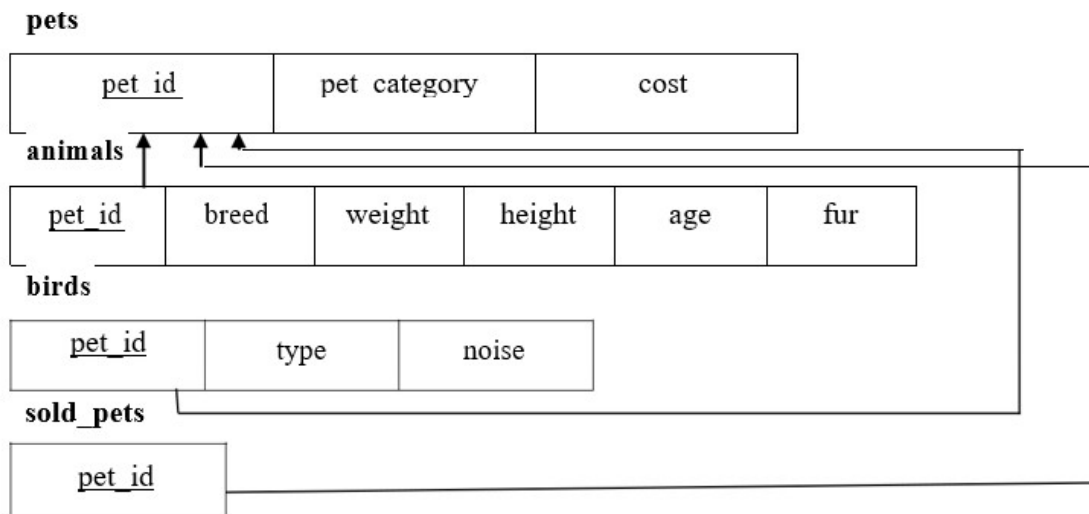
| | | |
|---------------|------|-------|
| <u>pet_id</u> | type | noise |
|---------------|------|-------|

2.2.3 Mapping of 1:1 Relationship Type

For each binary 1:1 relationship type are in the ER schema identify the relation S and T That correspond to the entity type participating in are. There are three are possible approaches

The foreign key approach, The merged relationship approach, The cross reference or relationship relation approach,

Foreign key approach: Chooses one of the relation S and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S include all the simple attribute of the 1:1 relationship type R as a attribute of S. In this database, relationship type animal_types is mapped by choosing the primary key of pets relation and included as foreign key in animals , relationship type birds_types is mapped by choosing the primary key of pets relation and included as foreign key in birds and relationship type 'has' is mapped by choosing the primary key of pets relation and included as foreign key in sold_pets.



2.2.4 Mapping of 1:N Relationship Type

For each regular binary 1: N relationship type R.

Identify the relation S that represents the participating entity type at the N-side of the relationship type.

Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

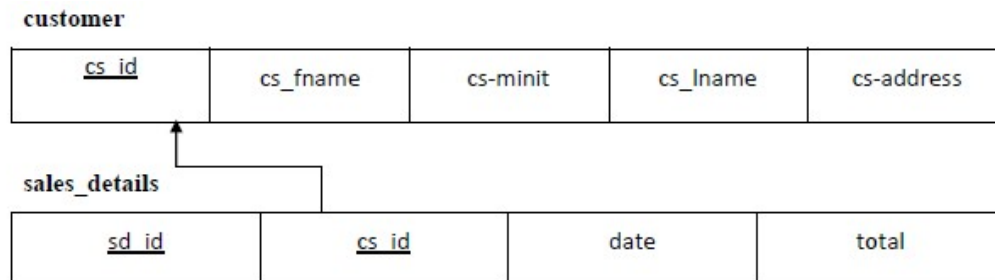
Include any simple attributes of the 1:N relationship type as attribute of S.

In this database ,the relation type ‘includes’ is mapped by choosing the primary key of sales_details relation and included as foreign key in sold_pets.



2.2.5 Mapping of N:1 Relationship Type

In this database ,the relation type ‘has’ is mapped by choosing the primary key of customer relation and included as foreign key in sales_details.



2.2.7 Mapping of M:N Relationship Type

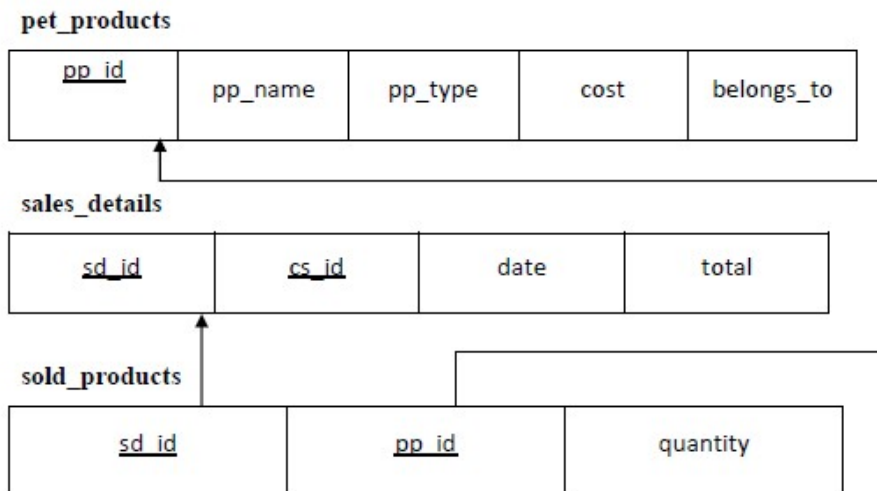
For each binary M:N type R

Create new relation S to represent R

Include a foreign key attributes in S the primary keys of the relations that represent the participating entity types their combinations will form the primary key of S

Also, include any simple attributes of the M:N relationship type as attributes of S

In this database relation ‘sold_products’ is mapped by choosing primary key of sales_details and pet_products and included as foreign key in sold_products entity.



2.2.8 Mapping Multivalued Attribute

An attributes that may have multiple values for the same entity. In this database cs_phone of customer entity can have multiple value.



2.2.9 Mapping ER Diagram to Relational Schema

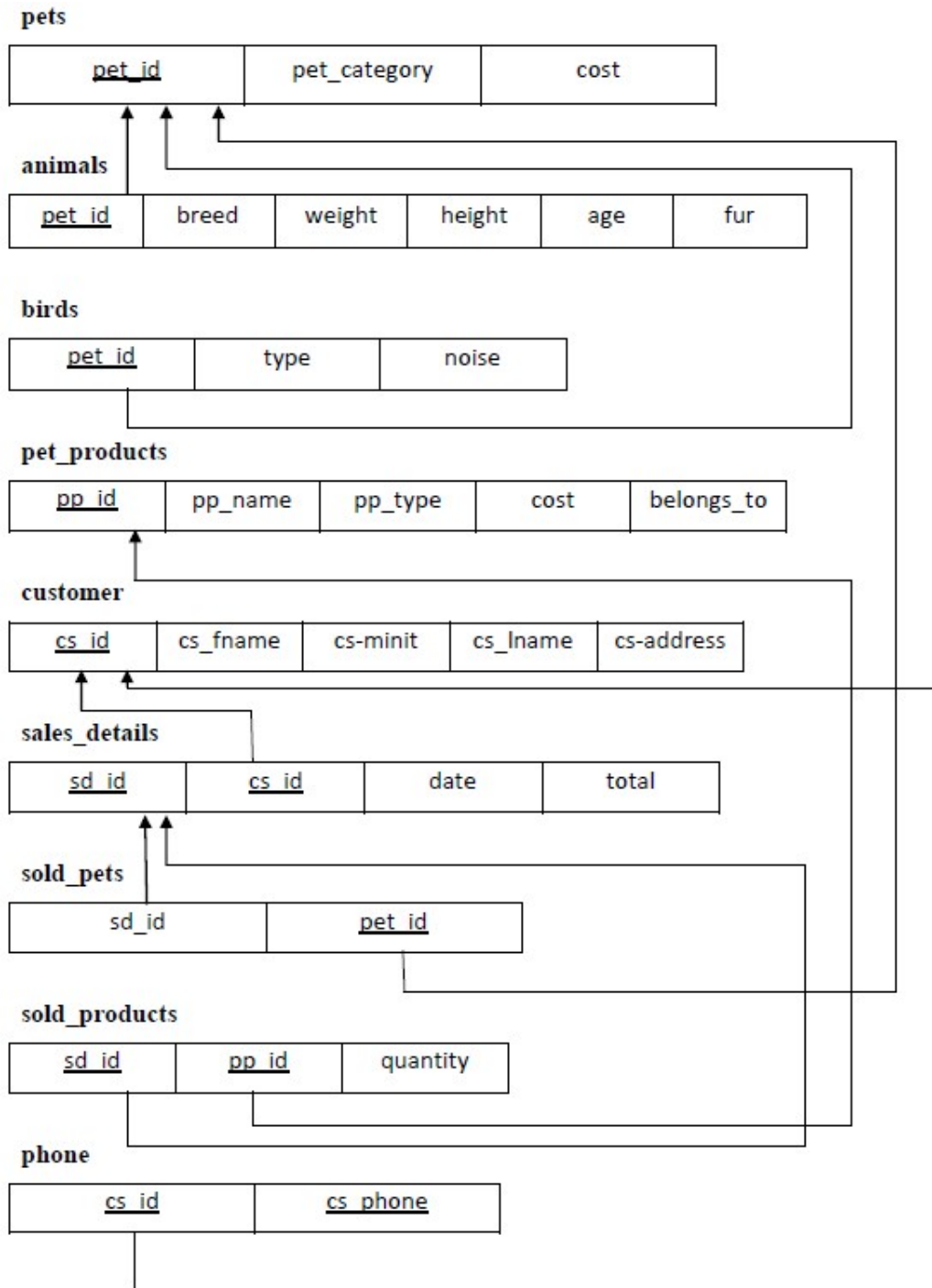


FIG 2.2: Schema diagram of pet shop management system.

2.3 Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

There are three main types of normal forms:

- a) First normal form(1NF)
- b) Second normal form(2NF)
- c) Third normal form(3NF)

1. First normal form (1NF)

a) As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values.

b) It should hold only atomic values.

This table holds only the atomic values company id and the company name and no multiple values are stored in this table so it can be considered as the 1NF.

2. Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- a) Table is in 1NF (First normal form)
- b) No non-prime attribute is dependent on the proper subset of any candidate key of table.
- c) An attribute that is not part of any candidate key is known as non-prime attribute

3. Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- a) Table must be in 2NF
- b) Transitive functional dependency of non-prime attribute on any super key should be removed.
- c) An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:

X is a super key of table

Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

The relations are already in the normalized form in the schema diagram without any redundancy.

Chapter 3

Front-end Design

3.1 Screen Layout Design

The screen design for the Pet Shop Management System has been carefully crafted using HTML and CSS with a goal to create an interactive, responsive, and visually intuitive web interface for managing various components of a pet shop. This interface serves as the home page of the system and provides access to essential modules like Animals, Birds, Products, Sales Details, and Customer information.

1. Overall Layout & Structure

The web page begins with a standard `<!DOCTYPE html>` declaration, defining the document as an HTML5 page. The `<head>` section includes a `<title>`, which appears on the browser tab, and a `<meta name="viewport">` tag, crucial for making the page mobile-responsive by adjusting the width and scale depending on the user's device screen.

The styling is written within a `<style>` tag directly in the head section, which is suitable for small-scale or educational projects. This CSS provides both the aesthetics and layout of the page, including the background, fonts, color schemes, and button styles.

The overall page structure is divided into three major sections:

- Top Navigation Bar
- Background (Screen) with Form Buttons
- Styling and Interactivity for Buttons

2. Top Navigation Bar (.topnav)

At the top of the interface is a navigation bar styled using the `.topnav` class. The navigation bar uses a dark maroon-brown background color (`rgb(73, 25, 21)`) with a solid black border, which gives it a bold and formal appearance. The height is fixed at 70 pixels to ensure consistency across devices.

The navigation bar contains three key elements:

- Logo: Implemented using an image tag (``). The logo visually brands the system and is linked to the home page.
- Title Link: A clickable label "Pets Shop" that also navigates to `home.html`. It is styled with a large font size (35px) and bold text to make it stand out.
- Logout Button: Positioned to the far right using the `.topnav-right` class and linked to a placeholder `#about`. This element allows users to securely exit the system.

The navigation bar is made sticky in layout by ensuring its overflow is hidden and the items are floated properly. This ensures a clean and organized header that separates the page logically from the interactive elements.

3. Main Display Section (.screen)

The primary visual section of the page is implemented with the `.screen` class. This is a `div` element that acts as the central background for the page's main content.

The `.screen`:

- Uses a background image (`aaron.jpg`) that is set to cover the entire container, ensuring no tiling and full coverage.

- Has a fixed height of 600 pixels and takes up the full width (100%) of the viewport.
- Contains a form with all the navigation buttons that serve as gateways to different parts of the system.

The use of a visually appealing background image improves user engagement and makes the application feel more personalized and immersive. The image likely relates to pets, giving the user an immediate sense of the system's purpose.

4. Form and Button Structure

Inside the .screen div, there's a <form> that contains five buttons. Each of these buttons navigates to a separate PHP backend script which presumably manages CRUD (Create, Read, Update, Delete) operations related to that entity.

Buttons include:

1. Animals (animals.php)
2. Birds (birds.php)
3. Products (petproducts.php)
4. Sales Details (sales.php)
5. Customer (customer.php)

Each button uses a common class .button for shared styling (such as padding, font size, alignment, and transitions), and also a specific class (button1, button2, ..., button5) to differentiate in color, hover effects, and borders.

For example:

- button1 uses a green theme with #4CAF50
- button3 uses a red theme with #f44336
- button5 has a purple/pinkish theme with #b40a70

This color differentiation helps users quickly distinguish between different functions. The borders are thick and vibrant, giving the interface a dynamic and modern feel. The hover effects provide interactivity, confirming to the user that the button is clickable.

The form uses formaction attributes in each button to individually redirect to the required PHP file. This is a smart use of HTML5 features where each button inside the same form can lead to different destinations without needing multiple forms.

5. Responsiveness & Accessibility

The screen is designed with a responsive intent in mind. The inclusion of the meta viewport tag allows the layout to adapt to different screen sizes, from desktops to tablets and mobile phones.

However, some enhancements could be made to further improve responsiveness, such as using media queries, grid or flexbox layout for form buttons, and adjusting font sizes and padding for smaller devices. For basic usage, though, the current layout is quite effective.

Accessibility could also be enhanced by:

- Adding alt text to the logo image.
- Including aria-labels for buttons.
- Using semantic HTML tags like <nav>, <header>, and <main> for better screen reader support.

6. Aesthetic Choices and Color Psychology

The choice of dark header colors mixed with colorful buttons provides both a professional and engaging look. Here's the psychology behind some of the colors used:

- Green (#4CAF50): Represents growth and health — fitting for a pet-related business.
- Blue (rgba(31, 58, 147, 1)): Indicates trust and reliability — good for bird records.
- Red (#f44336): Signals urgency or action — appropriate for sales.
- Orange (rgba(249, 105, 14, 1)): Reflects energy — perfect for dynamic operations like products.
- Purple (#b40a70): Adds a creative and luxurious touch — works well for customer interaction.

The background image provides emotional engagement while the white font and transparent button backgrounds help maintain clarity without making the interface overwhelming.

7. Code Quality and Best Practices

The HTML and CSS are relatively clean and easy to follow. Still, a few minor syntax errors were fixed:

- Misspelled `buttonclass` changed to `class="button button3"` (etc.)
- Added missing spaces and proper formatting
- Ensured HTML tags were properly closed

For real-world or production-grade applications, it's best to:

- Move CSS to an external stylesheet
- Use semantic tags and follow accessibility standards
- Validate HTML and CSS using W3C tools

8. Functional Perspective

From a user flow perspective, the screen is highly functional:

- A user logs in and lands on this homepage
- They can immediately choose an action — whether to view animals, birds, inventory (products), recent sales, or customer data
- Each button directs to a backend PHP handler, ensuring database integration and dynamic content

This interface can be easily extended with additional modules like “Appointments,” “Medical Records,” or “Feedback” as the business expands.

3.2 Connectivity

1.mysql connect ()

To connect to MySQL using the MySQL Improved extension, follow these steps:

Use the following PHP code to connect to MySQL and select a database. Replace username with your username, password with your password, and dbname with the database name:

```
<?php
$mysqli = new mysqli("localhost", "username", "password", "dbname");
?>
```

b) After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations.

The connectivity code used in this database is as follows:

```
<?php
$servername="localhost";
$username = "root";
$password = ""
$dbname="petshop_ management"
$conn= new mysqli( $servername,$username,$password,$dbname);
if ($conn -> connect_error)
{ die ("connection failed:".$conn->connect_error); }
```

2. close() - Closing a Database Connection

It is not always necessary to close a connection when you are finished, but it is advised. It is, however, necessary to close the connection to the database if you want to open up a new connection to a different database.

To close a connection to a database, we use the `mysql_close()` function, as follows:

```
mysql_close();
```

3. Error Handling

It is useful when debugging, and even when you just want to make sure that a database does not behave unexpectedly. Once a query has been created via the `mysql_query()` function, any error messages generated will be stored in the `mysql_error()` function. Here is a sample code snippet to display a error message. However, when there is no error messages, a blank string is returned.

```
print mysql_error();
```

Chapter 4

Implementation

4.1 Front-end Code

4.1.1 To link all tables

```
<!doctype html>
<html>
<head>
<title>Petshop</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
    margin: 0; background-size: cover; font-family: Arial, Helvetica, sans-serif;
}
.topnav {
    overflow: hidden; background-color:rgb(73, 25, 21); height: 70px; border: 2px solid
    black;
}
.topnav a { float: left; color: #f2f2f2; text-align: center; padding: 14px 16px; text-
decoration:
none; font-size: 35px;font-weight: bold;
}
.topnav-right {float: right;
}
.button {
    background-color: #4CAF50; border: none; color: white; padding: 16px 32px; text-
align:
center; text-decoration: none; display: inline-block; font-size: 16px; margin: 180px 8px;
-webkit-transition-duration: 0.2s; transition-duration: 0.2s; cursor: pointer;
}
.screen{
    background-image:url('aaron.jpg');background-size: cover;width:100%;height:600px;
}
```

```
.button1 {
    background-color: transparent;color:white; border: 3px solid #4CAF50;border-radius:
    5px;
}
.button1:hover {
    background-color: #4CAF50;color: white;
}
.button2 {
    background-color: transparent;color: white; border-radius: 5px;
border: 3px solid rgba(31, 58, 147, 1);
}
.button2:hover {
    background-color:rgba(31, 58, 147, 1); color: white;
}
.button3 {
    background-color:transparent; color: white; border-radius: 5px;border: 3px solid
    #f44336;
}
.button3:hover {
    background-color: #f44336;color: white;
}
.button4 {
    background-color: transparent; color: white; border-radius: 5px;
border: 3px solid rgba(249, 105, 14, 1);
}
.button4:hover {background-color:rgba(249, 105, 14, 1);color:white;
}
.button5 {
    background-color: transparent;color: white;border-radius: 5px;border: 3px solid
    #b40a70;
}
.button5:hover {
```

```

        background-color:#8d2663; color: white;
    }
</style>
</head>
<body>
<div class="topnav">
    <a class="active" href="home.html"></a> <a href="home.html">pets shop</a>
    <div class="topnav-right">
        <a href="#about">logout</a></div>
    </div>
<div class="screen">
    <form>
        <button class="button button1" type="submit"
        formaction="animals.php">animals</button>
        <button class="button button2" type="submit" formaction="birds.php">Birds</button>
        <button class="button button3" type="submit" formaction="petproducts.php">products</button>
        <button class="button button4" type="submit"
        formaction="sales.php">salesdetails</button>
        <button class="button button5" type="submit" formaction="customer.php">customer</button>
    </form>
</div>
</body>
</html>

```

4.1.2 DISPLAY CODE FOR PET_PRODUCTS TABLE

```

<?php
$con = mysqli_connect("localhost","root","","Petshop_management");
if(!$con){ die("could not connect".mysql_error());}
$var=mysqli_query($con,"select * from pet_products ");
echo "<table border size=10>";
echo "<tr><th>pp_ID</th><th>pp_name</th><th>pp_type</th><th>cost</th>";
echo "<th>belongs_to</th></tr>";

```



```

if(mysqli_num_rows($var)>0){
while($arr=mysqli_fetch_row($var))
{ echo "<tr><td>$arr[0]</td><td>$arr[1]</td><td>$arr[2]</td> <td>$arr[3]</td>
<td>$arr[4]</td> </tr>";
}
echo "</table>";
mysqli_free_result($var);
}
mysqli_close($con);
?>

```

4.1.3 INSERTION CODE FOR PET_PRODUCTS TABLE:

```

<form>
<form method="post" action="productsadd.php">
<fieldset>
<input type="text" name="id" placeholder=" Enter product_id"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;" >
<br><br>
<input type="text" name="name" placeholder=" Enter product name"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;">
<br><br>
<input type="text" name="type" placeholder=" Enter product type"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;">
<br><br>
<input type="number" name="cost" placeholder=" Enter cost"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;">
<br><br>
<input type="text" name="belong" placeholder=" which pet category it belongs to"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;">
<br><br>
<input type="submit" name="submit" value="save" placeholder=" which pet category it
belongs to" style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; cursor:pointer;background-
color:#f44336">&ensp;
</fieldset>
</form>
<?php
if(isset($_POST["submit"]))
{
$servername = "localhost";

```

```

$username = "root";
$password = "";
$dbname = "Petshop_management";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {die("Connection failed: " . $conn->connect_error);}
$id = $_POST["id"];
$name = $_POST["name"];
$type = $_POST["type"];
$belongs = $_POST["belong"];
$cost = $_POST["cost"];
$sql = "INSERT INTO pet_products( pp_id,pp_name,pp_type,cost,belongs_to)
VALUES ('$id','$name','$type','$cost','$belongs')";
if ($conn->query($sql) == TRUE) {echo "New record of id=$id created successfully";}
else { echo "Error: " . $sql . "<br>" . $conn->error;}
$conn->close();
}
?>

```

4.1.4 DELETION CODE FOR PET_PRODUCTS TABLE:

```

<form action="deleteproducts.php" method="post">
<input type="text" name="t1" placeholder="Enter the id to delete" required >
<input style="width:75px;height:44px;cursor:pointer;border-radius:15px;
border: 3px solid #ff0000;background-color:#f44336;color:#f2f2f2;font-size:17px;"
type="submit" value="delete">
</form>
<?php $servername = "localhost";
$username = "root";
$password = "";
$dbname = "Petshop_management";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);}
$pp_id=$_POST["t1"];
$sql = "DELETE FROM pet_products WHERE pp_id='$pp_id'";
if ($conn->query($sql) == TRUE) {echo "<div><h1 style='color:#f2f2f2;font-size:50px;
font-family: \"Roboto\", sans-serif;margin:auto;'>Data deleted successfully</h1> </div>";}
else {echo "Error: " . $sql . "<br>" . $conn->error;}
$conn->close();
?>

```

4.1.5 UPDATION CODE FOR PET_PRODUCTS TABLE:

```

</form>
<form method="post" action="productupdate.php">
<fieldset>

```

```

<input type="text" name="id" placeholder="Enter product_id"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;" ><br><br>
<input type="text" name="name" placeholder=" Enter product name"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;"><br><br>
<input type="text" name="type" placeholder=" Enter product type"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;"><br><br>
<input type="number" name="cost" placeholder=" Enter cost"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;"><br><br>
<input type="text" name="belong" placeholder=" which pet category it belongs to"
style="width:100%;height:30px;
border: 2px solid #f44336; border-radius:3px; background:transparent;"><br><br>
<input type="submit" name="submit" value="update" placeholder=" which pet category
it belongs to" style="width:100%;height:30px;border: 2px solid #f44336; border-
radius:3px; cursor:pointer;background-color:#f44336">&nbsp; </fieldset></form>
<?php
if(isset($_POST["submit"]))
{
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "Petshop_management";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {die("Connection failed: " . $conn->connect_error);}
    $id = $_POST["id"];
    $name = $_POST["name"];
    $type = $_POST["type"];
    $belongs = $_POST["belong"];
    $cost = $_POST["cost"];
    $sql = "UPDATE pet_products SET pp_name='$name',pp_type='$type',cost='$cost'
,belongs_to='$belongs' WHERE pp_id='$id'";
    if ($conn->query($sql) == TRUE) { echo "id=$id updated successfully";}
    else { echo "Error: " . $sql . "<br>" . $conn->error;}
    $conn->close();
}
?>

```

4.2 Back-End Code

4.2.1 Creation of Tables

```

create table pets( pet_id varchar(9) not null,
pet_category varchar(15) not null,
cost int(11) not null,

```

```
primary key(pet_id));
```

```
create table animals(pet_id varchar(9) not null,  
breed varchar(30) not null,  
weight float not null,  
height float not null,  
age int(11) not null,  
fur varchar(15) not null,  
primary key(pet_id),  
foreign key(pet_id) references pets(pet_id) on delete cascade);
```

```
create table birds(pet_id varchar(9) not null,  
type varchar(25) not null,  
noise varchar(10) not null,  
primary key(pet_id),  
foreign key(pet_id) references pets(pet_id) on delete cascade);
```

```
create table pet_products(pp_id varchar(9) not null,  
pp_name varchar(30) not null,  
pp_type varchar(20) not null,  
cost int(11) not null,  
belongs_to varchar(20) not null,  
primary key(pp_id));
```

```
create table customer(cs_id varchar(9) not null,  
cs_fname varchar(10) not null,  
cs_minit varchar(10) not null,  
cs_lname varchar(10) not null,  
cs_address varchar(30) not null,  
primary key(cs_id));
```

```
create table phone (cs_id varchar(9) not null,  
cs_phone bigint(10) not null,  
primary key(cs_id,cs_phone),  
foreign key(cs_id) references customer(cs_id)on delete cascade);
```

```
create table sales_details(sd_id varchar(9) not null,  
cs_id varchar(9) not null,  
date date not null,  
total int(11) not null,  
primary key(sd_id,cs_id),  
foreign key(cs_id)references customer(cs_id)on delete cascade);
```

```
create table sold_pets(sd_id varchar(9) not null,
pet_id varchar(9) not null,
primary key(pet_id),
foreign key(sd_id)references sales_details(sd_id)on delete cascade,
foreign key(pet_id)references pets(pet_id)on delete cascade);
```

```
create table sold_products(sd_id varchar(9) not null,
pp_id varchar(9) not null,
quantity int(11) not null,
primary key(pet_id,pp_id),
foreign key(sd_id)references sales_details(sd_id)on delete cascade,
foreign key(pp_id)references pet_products(pp_id)on delete cascade );
```

4.2.2 Insertion into tables

```
INSERT INTO 'pets' (`pet_id`, `pet_category`, `cost`) VALUES
('pa01', 'dog', '8000'),
('pa02', 'cat', '3000'),
('pa03', 'dog', '8500'),
('pa04', 'dog', '15000'),
('pa05', 'cat', '3500')
```

```
INSERT INTO `animals`(`pet_id`, `breed`, `weight`, `height`, `age`, `fur`)VALUES
('pa01', 'labrador', '11.3', '30', '2', 'white'),
('pa02', 'parsian', '3.6', '20', '2', 'white'),
('pa03', 'goldenretriever', '12.5', '40', '2', 'gloden'),
('pa04', 'boxer', '11.5', '45', '3', 'black'),
('pa05', 'rag doll', '2.6', '20', '5', 'white')
```

```
INSERT INTO `birds` (`pet_id`, `type`, `noise`) VALUES
('pb01', 'grey parrot', 'moderate'),
('pb02', 'black cheeked', 'low'),
('pb03', 'grey headed', 'moderate'),
('pb04', 'lilian', 'moderate'),
('pb05', 'white cockatoo', 'moderate')
```

```
INSERT INTO `pet_products`(`pp_id`, `pp_name`, `pp_type`, `cost`, `belongs_to`)
VALUES ('pp01', 'dog collar', 'accessories', '500', 'dog'),
('pp02', 'chain', 'accessories', '100', 'cat'),
('pp03', 'pedigree', 'food', '1500', 'dog'), ('pp04', 'mouth mask', 'accessories', '250', 'dog'),
('pp05', 'food bowl', 'accessories', '250', 'dog ')
```

```
INSERT INTO `customer`(`cs_id`,`cs_fname`,`cs_minit`,`cs_lname`,`cs_address`)
VALUES ('cs01','Naveen','kumar','k','Mandya'),
('cs02','manjunath','kumar','h v','BENGALURU'),
('cs03','pavan','chikkanna','gowda','BENGALURU'), ('cs04','kushal','kumar','k',
'BENGALURU'),
('cs05','ravi','shankar','c','BENGALURU')
```

```
INSERT INTO `phone`(`cs_id`,`cs_phone`) VALUES
('cs01','8867762336'),
('cs01','9902587276'),
('cs03','9845034784'), ('cs04','6361261639'),
('cs05','86660873855')
```

```
INSERT INTO `sales_details`(`sd_id`,`cs_id`,`date`,`total`) VALUES
('sd01','cs03','2018-10-26','9500'),
('sd02','cs01','2018-11-01','3000'),
('sd03','cs03','2018-11-08','500'),
('sd04','cs04','2018-11-15','250'),
('sd05','cs02','2018-11-17','9350')
```

```
INSERT INTO `sold_pets`(`sd_id`,`pet_id`) VALUES
('sd01','pa01'),
('sd02','pa02'),
('sd05','pa03'),
('sd06','pb02'), ('sd06','pb04')
```

```
INSERT INTO `sold_products`(`sd_id`,`pp_id`,`quantity`) VALUES
('sd01','pp03','1'),
('sd03','pp01','1'),
('sd04','pp04','1'),
('sd05','pp05','1'),
('sd05','pp06','2')
```

4.2.3 Creation of Triggers

A trigger is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data.

Here in this database, trigger avoids the updation of sold pet values in pet entity .

```

create or replace trigger check_sold
before update on pets
for each row
BEGIN
DECLARE
checking int;
set checking=(select count(*) from sold_pets where pet_id=old.pet_id);
if (checking > 0) then
signal sqlstate '45000' set message_text = 'cannot update sold pet';
end if;
END

```

4.2.4 Creation of Stored Procedures

A stored procedure is a set of structured query language(SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs. Stored procedures can access or modify data in a database

Here in this database , there are two stored procedures

1. calculations_for_pets : it calculates the cost of pet sold to a particular sale and updates that in sales_details entity by adding the cost with the old total value of that sale.
2. calculations_for_product: it calculates the cost of product sold to a particular sale and updates that in sales_details entity by adding the cost with the old total value of that sale.

```

create procedure calculations_for_pets(in pid varchar(9),in sid varchar(9))
BEGIN
DECLARE
cpid,csid int DEFAULT 0;
set cpid=(select cost from pets where pet_id=pid);
set csid=(select total from sales_details where sd_id=sid);
set csid=csid+cpid;
update sales_details set total=csid where sd_id=sid;
end

```

```

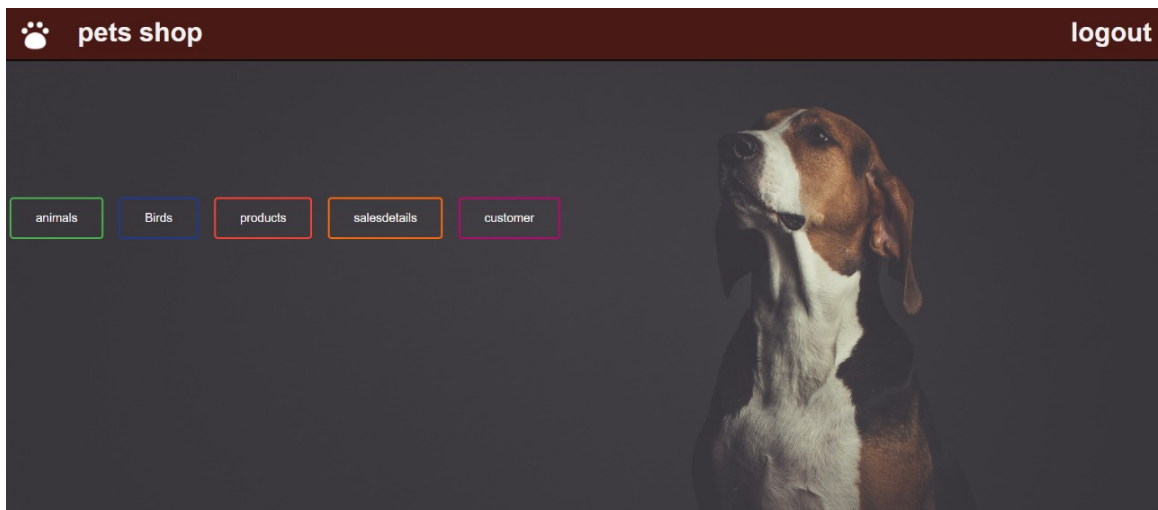
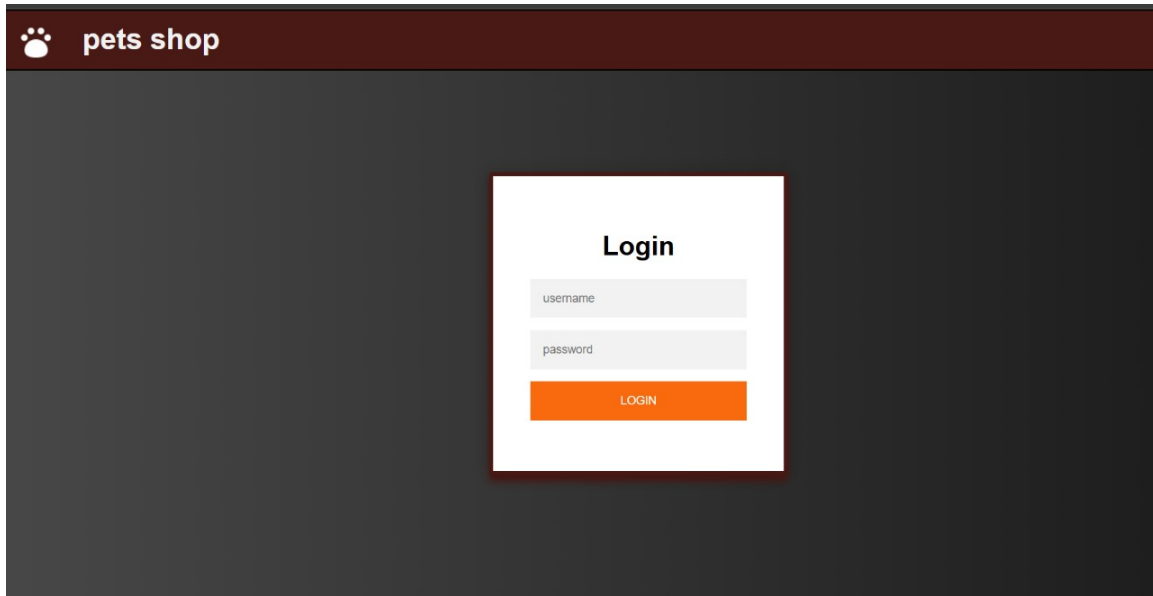
create procedure calculations_for_product(in ppid varchar(9),in sid varchar(9),in qnty
int(11))
BEGIN
DECLARE
cppid,csid int DEFAULT 0;
set cppid=(select cost from pet_products where pp_id=ppid);
set csid=(select total from sales_details where sd_id=sid);
set csid=csid+qnty*cppid;
update sales_details set total=csid where sd_id=sid;


```

end

Chapter 5

Snapshots





Animals
logout

Add new animal
update animal

| pet_ID | petcategory | breed | weight(kg) | height(cm) | age(m) | fur | cost(Rs) |
|--------|-------------|------------------|------------|------------|--------|-----------------|----------|
| 23 | 222 | 32112 | 32 | 42142 | 421 | 342424 | 422 |
| pa01 | dog | labrador | 11.3 | 30 | 2 | white | 8000 |
| pa02 | cat | parsian | 3.6 | 20 | 2 | white | 3000 |
| pa03 | dog | golden retriever | 12.5 | 40 | 2 | gloden | 8500 |
| pa04 | dog | boxer | 11.5 | 45 | 3 | black | 15000 |
| pa05 | cat | rag doll | 2.6 | 20 | 5 | white | 3500 |
| pa06 | dog | st bernard | 10.8 | 35 | 3 | brownish yellow | 10500 |
| pa07 | dog | bulldog | 8 | 25 | 3 | white | 12000 |


Delete


Birds
logout

Add new bird
update bird

| pet_ID | petcategory | type | noise | cost |
|--------|-------------|----------------|----------|-------|
| pb02 | lovebirds | black cheeked | low | 800 |
| pb03 | lovebirds | grey headed | moderate | 600 |
| pb04 | lovebirds | lilian | moderate | 800 |
| pb05 | cockatoo | white cockatoo | moderate | 10000 |


delete


pets products
logout

Add new product
update product

| pp_ID | pp_name | pp_type | cost | belongs_to |
|-------|------------|-----------|------|------------|
| pp01 | dog collar | acesories | 500 | dog |
| pp02 | chain | acesories | 100 | cat |
| pp03 | pedigree | food | 1500 | dog |
| pp04 | mouth mask | acesories | 250 | dog |
| pp05 | food bowl | acesories | 250 | dog |
| pp06 | bird feeds | food | 300 | birds |


delete


Customers
logout

Add new customer
update customer
phone nos.

| cs_ID | cs_fname | cs_minit | cs_lname | cs_address |
|-------|-----------|-----------|----------|------------|
| aa | 123 | 12 | 22 | 22 |
| cs01 | Naveen | kumar | k | Mandya |
| cs02 | manjunath | kumar | h v | BENGALURU |
| cs03 | pavan | chikkanna | gowda | BENGALURU |
| cs04 | kushal | kumar | k | BENGALURU |
| cs05 | ravi | shankar | c | BENGALURU |
| cs06 | Ankit | kumar | jha | Bhagalpur |
| cs07 | Ayush | Raj | singh | patna |
| cs10 | Aryan | kumar | Jaiswal | Ranchi |

Delete


Sales details
logout

Add new details
update details
sold products
sold pets

| sd_ID | cs_id | date | total |
|-------|-------|------------|-------|
| sd01 | cs03 | 2018-10-26 | 9500 |
| sd02 | cs01 | 2018-11-01 | 3000 |
| sd03 | cs03 | 2018-11-08 | 500 |
| sd04 | cs04 | 2018-11-15 | 12250 |
| sd05 | cs02 | 2018-11-17 | 9350 |
| sd06 | cs05 | 2018-11-20 | 1900 |
| sd07 | cs03 | 2018-12-08 | 10000 |

Delete

Chapter 6

Conclusion

The Pet Shop Management System effectively demonstrates the application of database concepts in solving real-world problems associated with managing pet-related businesses. By leveraging a structured database approach, the system enhances the efficiency and accuracy of daily operations such as inventory tracking, customer management, sales processing, and service scheduling.

The back-end database has been designed using robust normalization principles and entity-relationship modeling, ensuring minimal redundancy and optimal data integrity. The logical schema and relational mappings provide a strong foundation for reliable data storage and retrieval. In addition, the implementation of triggers and stored procedures strengthens data consistency and enforces business rules.

On the front-end, the web-based user interface offers a responsive and intuitive platform for users to interact with the system. It integrates seamlessly with the backend through PHP and MySQL connectivity, enabling real-time data manipulation and reporting.

Overall, this project not only automates and simplifies the various operations of a pet shop but also serves as a practical example of applying DBMS concepts in a software development environment. It provides a scalable and maintainable solution that can be expanded with additional modules such as appointment scheduling, health records, and analytics in future enhancements.