

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO SQL

Structure Query Language (SQL) is a programming language used for storing and managing data in Relational Database Management System (RDBMS). SQL was the first commercial language introduced for E.F Codd's Relational model. Today almost all RDBMS (MySQL, Oracle, Infomix, Sybase, MS Access) uses SQL as the standard database language. SQL is used to perform all type of data operations in RDBMS. Most of the actions you need to perform on a database are done with SQL statements. SQL defines following data languages to manipulate data of RDBMS:

1.DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Eg: create - To create new table or database, alter - For alteration, truncate - Delete data from table, drop - To drop a table

2.DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Eg: insert - To insert a new row, update - To update existing row, delete - To delete a row, merge - merging two rows or two tables

3.TCL: Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Eg: commit - to permanently save, rollback - to undo change, save point - to save temporarily

4.DCL: Data Control Language

Data control language provides command to grant and take back authority.

Eg: grant - grant permission of right, revoke - take back permission

5.DQL: Data Query Language

DQL is used to operate on queries.

Eg: Select - retrieve records from one or more table

1.2 INTRODUCTION TO FRONT END SOFTWARE

The front end software used is PHP. PHP is an acronym for "PHP: Hypertext Preprocessor". PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. It is a widely used, open source scripting language. It is free to download and use. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP code are executed on the server, and the result is returned to the browser as plain HTML. PHP files have extension ".php".

PHP code may be embedded into HTML or HTML5 makeup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

1.3 PROJECT REPORT OUTLINE

CHAPTER 1: INTRODUCTION

The brief introduction about the backend software SQL, front end software HTML and the project report outline details are specified

CHAPTER 2: REQUIREMENT SPECIFICATION

The basic software requirements and hardware requirements to do this project are mentioned.

CHAPTER 3: OBJECTIVE OF PROJECT

The basic software requirements and hardware requirements to do this project are mentioned.

CHAPTER 4: IMPLEMENTATION

The implementation parts for developing the project are explained step wise briefly.

CHAPTER 5: FRONT END DESIGN

The front end design is explained by briefly describing about the system design and connectivity to the database. The front end codes used for main page, insertion, search, deletion are displayed.

CHAPTER 6: TESTING

The testing process, objectives and the test cases are tested and the expected results with the observed results are written with the remarks.

CHAPTER 7: RESULT

The results with the snapshots for the various operations are displayed with the snapshots.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Operating System : 64bit operating system, x64-based processor

Database : MYSQL

Tools : PHP, Xampp Server 3.2.2

2.2 HARDWARE REQUIREMENTS

Processor : Intel® Celeron® CPU N3060 @1.60GHz

RAM : 4.00 GB

Hard Disk : 1 TB

Compact Disk : CD-ROM, CD-R, CD-RW

Input device : Keyboard, mouse

Output device : Monitor screen

CHAPTER 3

OBJECTIVE OF THE PROJECT

The objectives of the project is to provide web based interface to a petshop owner to manages his petshop activities.

To provide an option for storing and managing the basic information about pets and pet products in the shop.

To provide an option for storing and managing the sales details of the shop.

To provide an option for storing and managing the basic information about the customer

To track the information about sold pets and products to a customer.

CHAPTER 4

IMPLEMENTATION

4.1 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: 4.1 is the ER diagram of “Petshop Management System” with entities pets, animals, birds, pet products, sales details, customer, sold pets

4.1 ER DIAGRAM

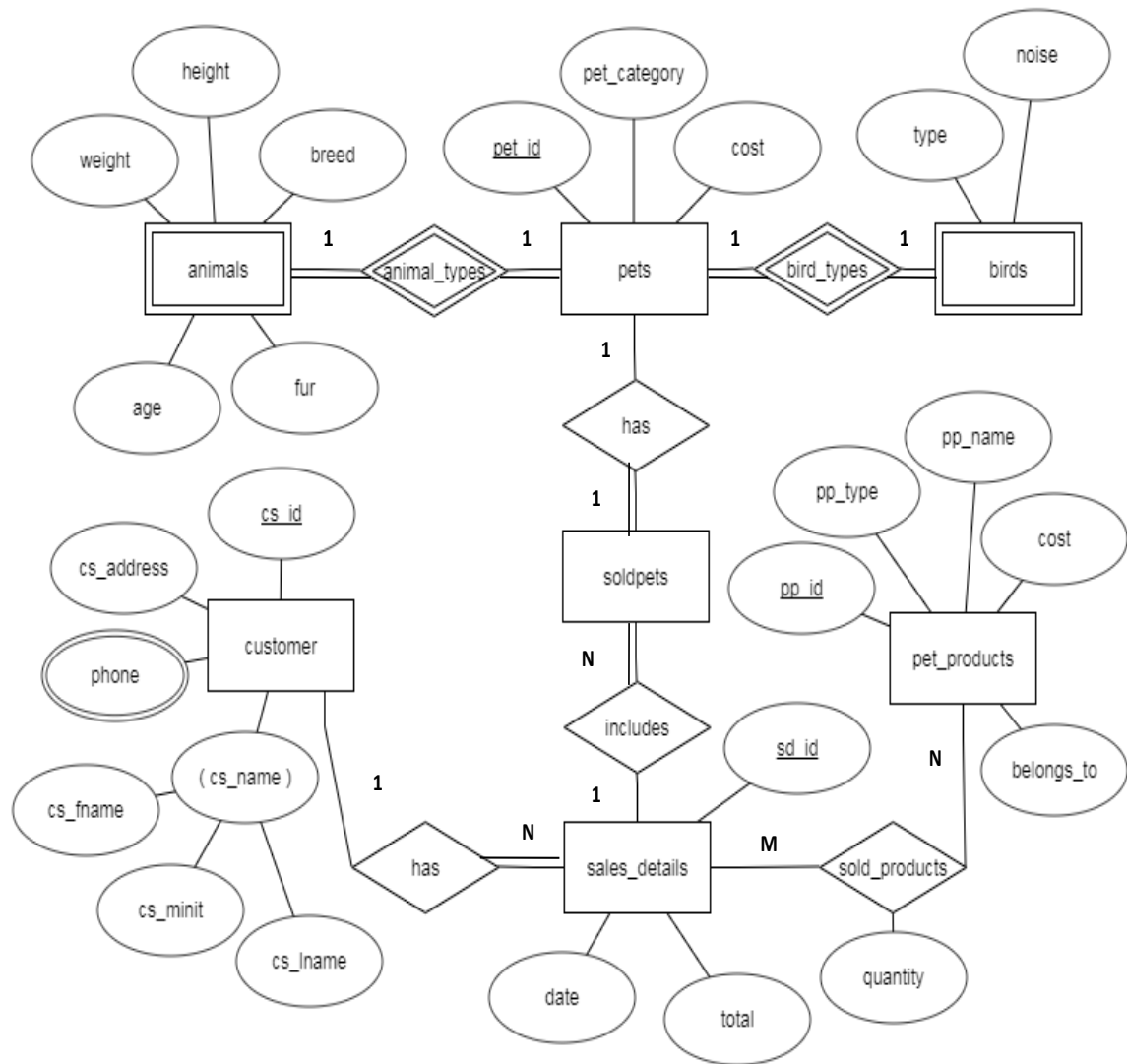


Fig.no 4.1: ER DIAGRAM OF PETSHOP MANAGEMENT SYSTEM

4.2 MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM

STEP 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

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

STEP 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
---------------	------	-------

STEP 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

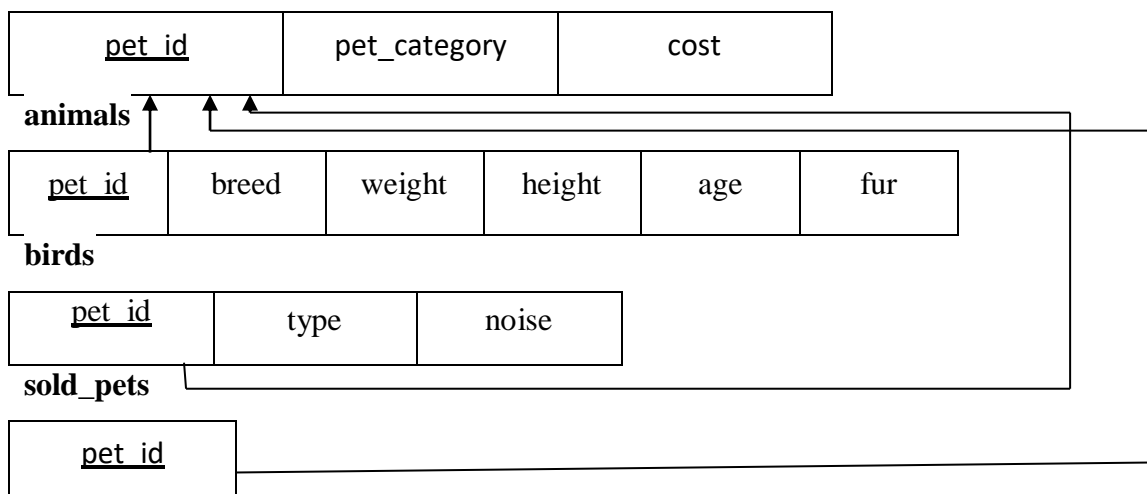
That correspond to the entity type participating in are. There are three are possible approaches

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.

pets



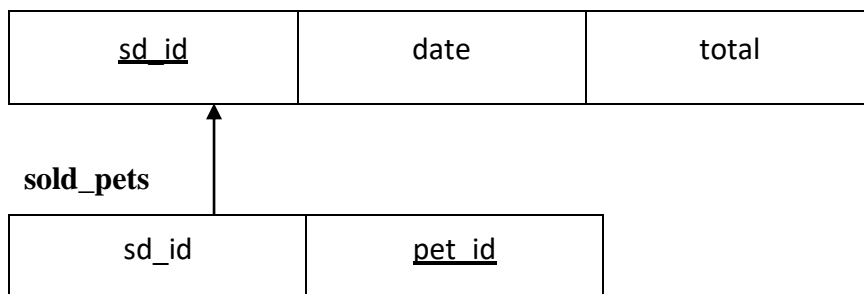
STEP 4: MAPPING OF 1:N RELATIONSHIP TYPE

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.

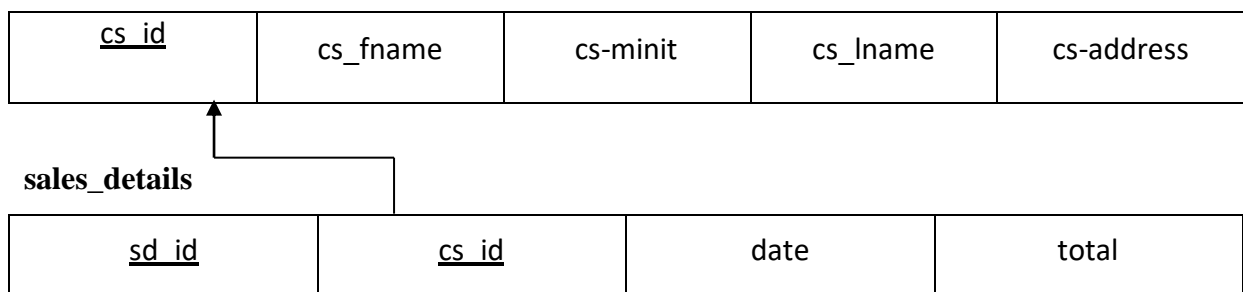
sales_details



STEP 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.

customer



STEP 6: MAPPING OF M:N RELATIONSHIP TYPE

For each binary M:N type R

- A) Create new relation S to represent R
- B) 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
- C) 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.

pet_products

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

sales_details

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

sold_products

<u>sd_id</u>	<u>pp_id</u>	quantity
--------------	--------------	----------

STEP 6:MAPPING MULTIVALUED ATTRIBUTES

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

phone

<u>cs_id</u>	<u>cs_phone</u>
--------------	-----------------

4.3 MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM

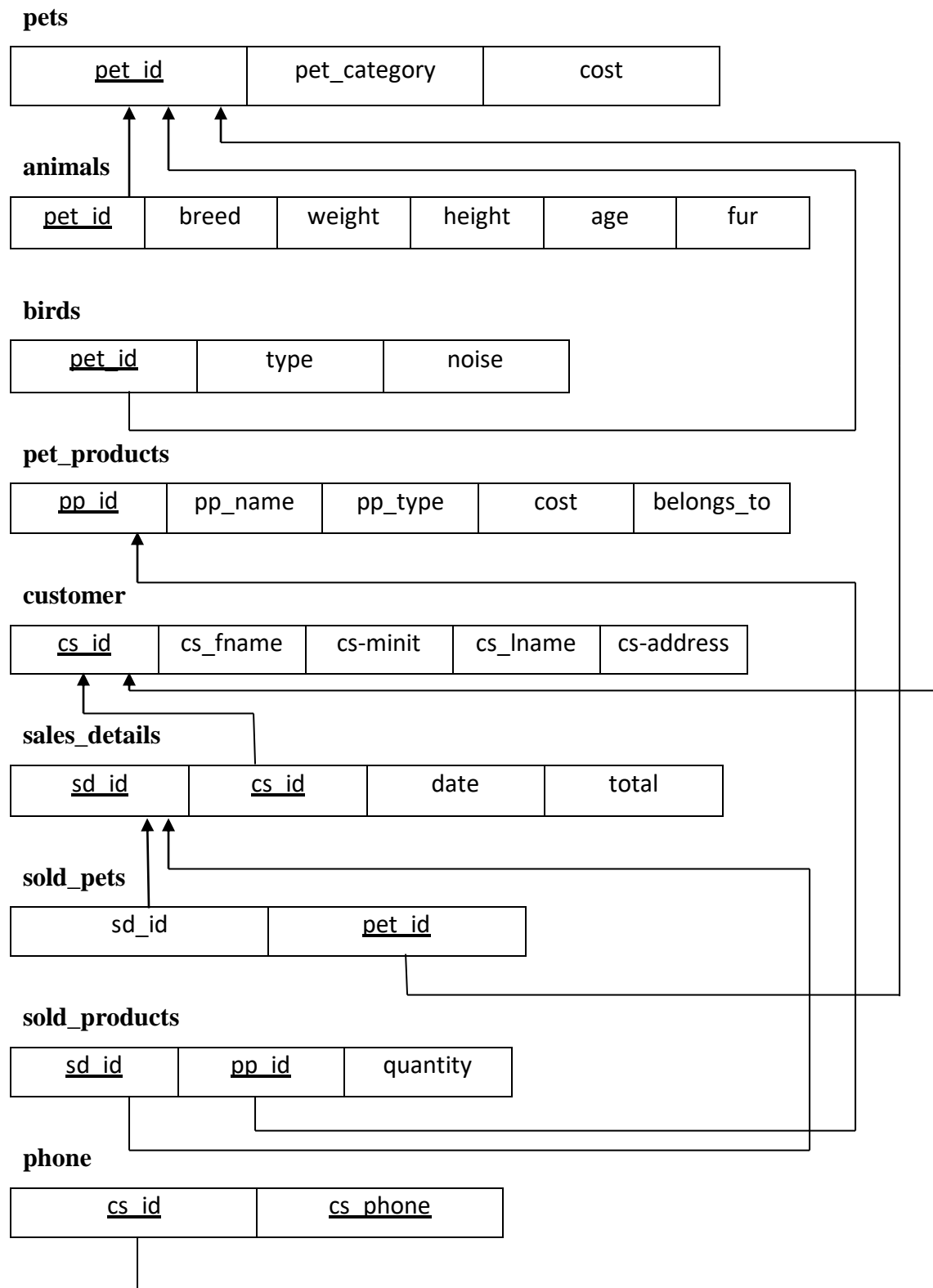


Fig. no 4.3 Schema diagram of pet shop management system.

4.4 NORMALIZE THE RELATIONS

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.

4.5 CREATION OF TABLES

1. CREATION OF PETS TABLE

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

2. CREATION OF ANIMALS TABLE

```
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);
```

3. CREATION OF BIRDS TABLE

```
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);
```

4. CREATION OF PET_PRODUCTS TABLE

```
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));
```

5.CREATION OF CUSTOMER TABLE

```
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));
```

6.CREATION OF PHONE TABLE

```
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);
```

7.CREATION OF SALES_DETAILS TABLE

```
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);
```

8.CREATION OF SOLD_PETS TABLE

```
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);
```

9. CREATION OF SOLD_PRODUCTS TABLE

```
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 );
```

4.6 INSERTION OF TUPLE

1. INSERTION OF PETS TABLE

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

2. INSERTION OF ANIMALS TABLE

```
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')
```


3. INSERTION OF BIRDS TABLE

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

4. INSERTION OF PET_PRODUCTS TABLE

```
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')
```

5. INSERTION OF CUSTOMER TABLE

```
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')
```

6. INSERTION OF PHONE TABLE

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

7. INSERTION OF SALES_DETAILS TABLE

```
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')
```

8. INSERTION OF SALES_DETAILS TABLE

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

9. INSERTION OF SALES_DETAILS TABLE

```
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')
```

4.7 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
    
```

4.8 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.

1. calculations_for_pets

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

2. calculations_for_product

**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

FRONT END DESIGN

5.1 CONNECTIVITY TO DATABASE

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.[citation needed] The UML has become the standard language in object-oriented analysis and design.[citation needed] It is widely used for modelling software systems and is increasingly used for high designing non-software systems and organizations.[citation needed]

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words the first step in the solution to the problem is the design of the project.

5.1.1 CONNECTIVITY FROM FRONT END TO BACK END PHP CODE

1.mysql connect ()

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

a) 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();`

5.2 FRONT END CODE

FRONT END PAGE CODE 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>
```

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>
<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);
?>
```

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();
}
?>
```

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();
?>
```

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();
}
?>

```

CHAPTER 6

TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 TESTING OBJECTIVES

The main objectives of testing process are as follows.

- a. Testing is a process of executing a program with the intent of finding an error.
- b. A good test case is one that has high probability of finding undiscovered error.
- c. A successful test is one that uncovers the undiscovered error.

6.3 TEST CASES

The test cases provided here test the most important features of the project

Table 6.3.1: TEST CASES FOR THE PETSHOP MANAGEMENT SYSTEM

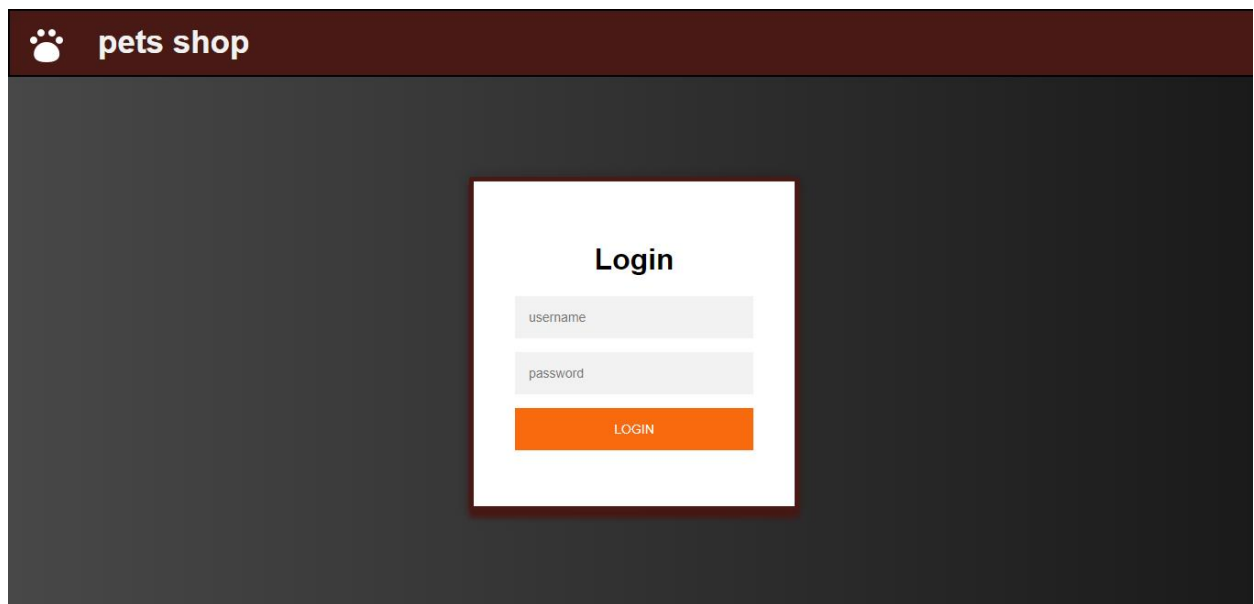
SI No	Test Input	Expected Results	Observed Results	Remarks
1	Insert a record	Insertion of a new tuple	Query OK 1 row affected or inserted	TRUE
2	Insert a record	Insertion of a new tuple	Error(Same Primary Key)	FAIL
3	update a record	Updation of a existing tuple	Required record is updated	TRUE
4	Update a record	Updation of a existing tuple	Required record not updated	FAIL
5	Delete a record	Deletion of a record	Record deleted successfully	TRUE
6	Display a record	Display the record	Record not deleted successfully	FAIL
7	Creation of trigger	Trigger created	Query OK.1 row effected.	TRUE
8	Creation of stored procedure	Stored procedure created	Your SQL query has been executed successfully.	TRUE

CHAPTER 7

RESULTS

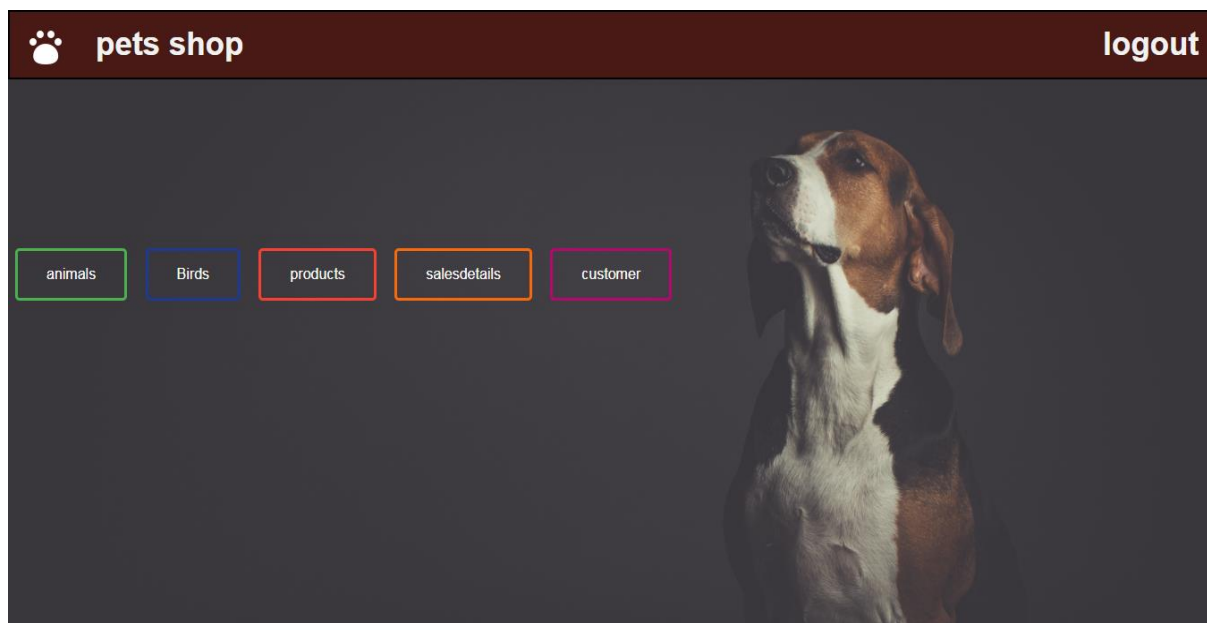
This section describes the screens of the “PETSHOP MANAGEMENT SYSTEM”. The snapshots are shown below for each module.

7.1 SNAPSHOTS



Snapshots 7.1.1: Login page

This page asks admin username and password for authentication ,if the authentication is successfull then it loads home page.



Snapshots 7.1.2: Home page

This page provides links to animals page, birds page, products page ,salesdetails page and customer page.

Animals								logout
Add new animal				update animal				
pet_ID	petcategory	breed	weight(kg)	height(cm)	age(m)	fur	cost(Rs)	
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	
pa08	dog	german shepard	12	42	4	black	10000	
<input type="text" value="Enter the id to delete"/> <input type="button" value="Delete"/>								

Snapshots 7.1.3: Animals page

This page displays the animals data and also provides link to access insertion and updation page of animals and also at left bottom of the page it gives an option for deletion.

The screenshot shows the 'Animals' page with a green header bar containing a paw icon, the title 'Animals', and a 'logout' link. Below the header is a 'Back' button. The main content area is a dark grey rectangle containing a white form with green borders. The form has the following fields: 'Enter pet_id', 'Enter pet_category', 'Enter breed', 'Enter weight' and 'Enter height' (side-by-side), 'Enter age' and 'Enter fur' (side-by-side), and 'Enter cost'. At the bottom of the form is a green 'save' button.

Snapshots 7.1.4: Animal insertion page

This page accept the data to save in animals entity and pet entity.

The screenshot shows the 'Birds' page with a blue header bar containing a paw icon, the title 'Birds', and a 'logout' link. Below the header are two buttons: 'Add new bird' and 'update bird'. The main content area is a dark grey rectangle containing a table with the following data:

pet_ID	petcategory	type	noise	cost
pb01	parrot	grey parrot	moderate	2000
pb02	lovebirds	black cheeked	low	800
pb03	lovebirds	grey headed	moderate	600
pb04	lovebirds	lilian	moderate	800
pb05	cockatoo	white cockatoo	moderate	10000

Below the table is a dark grey rectangle containing a text input field with the placeholder 'Enter the id to delete' and a blue 'delete' button.

Snapshots 7.1.5: Birds page

This page displays the birds data and also provides link to access insertion and updation page of Birds and also at left bottom of the page it gives an option for deletion.

pets products					logout
Add new product			update product		
pp_ID	pp_name	pp_type	cost	belongs_to	
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	
pp06	bird feeds	food	300	birds	
Enter the id to delete		delete			

Snapshots 7.1.6: pet products page

This page displays pet products data and also provides link to access insertion and updation page of pet products and also at left bottom of the page it gives an option for deletion.



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	250
sd05	cs02	2018-11-17	9350
sd06	cs05	2018-11-20	1600


Enter the id to delete

Delete

Snapshots 7.1.7: sales details page

This page displays sales details data and also provides link to access insertion and updation page of sales details and also at left bottom of the page it gives an option for deletion.

It also provides link to access sold pet and sold products page.


Customers
logout


Add new customer
update customer
phone nos.

cs_ID	cs_fname	cs_minit	cs_lname	cs_address
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

Delete

Snapshots 7.1.8: customers page

This page displays customers data and also provides link to access insertion and updation page of customers. and also at left bottom of the page it gives an option for deletion.


Animals
logout

Back

Snapshots 7.1.9: Animal insertion page

This page accepts the data to update in animals entity and pet entity.

CONCLUSION

The development of this Petshop Management System is great improvement over the manual system which uses lots of manual work and paper. The computerization of the system speeds up the process.

The Petshop Management System is fast, efficient and reliable, Avoids data redundancy and inconsistency. It contains all the functional features described in objective of the project.

REFERENCES

[1] Elmasri, Ramez, Fundamentals of database systems. Pearson education in india 2008.

[2] <https://programmerblog.net/createmysal-trigger-php/>

[3]<http://www.tutorialspoint.com>

[4] <http://www.w3schools.com>