# Deadline 5: PetPals

(Mridul Goel - 2022303, Sujal Soni - 2022513, Anushka Korlapati - 2022085, Abhinav Bagadi - 2022131)
Contribution: Equal contribution.

## Embedded SQL:

```
import mysql.connector
from mysql.connector import Error
import pandas as pd
from IPython.display import display


def create_server_connnection(hname, uname, passwd):
    connection = mysql.connector.connect(host=hname, user=uname, passwd=passwd)
    # print("Connection Successful")
    return connection


uname = "root"
hname = "localhost"
pas = "yourpass"
connection = create_server_connnection(hname, uname, pas)


def create_db_connnection(hname, uname, passwd, db_name):
    connection = mysql.connector.connect(host=hname, user=uname, passwd=passwd, database=db_name)
    # print("DB Connection Successful")
    return connection


# con = create_db_connnection(hname, uname, pas, "McM")

def execute_querries(connection, query):
    cursor = connection.cursor()
    cursor.execute(query)
    connection.commit()
    # print("Query was successful")


def read_query(connection, query):
    cursor = connection.cursor()
    cursor.execute(query)
    result = cursor.fetchall()
    return result
```

```
def bill(cart):
    bill = 0
    if len(cart) == 0:
        return 0
    else:
        for i in range(len(cart) - 3):
            bill += cart[i][2]
    return bill


def clear_cart(usr_id):
    clear_cart_q = f"""
                   delete from cart
                   where user_id = {usr_id};
                   """
    execute_querries(con, clear_cart_q)


def view_cart(usr_id, con):
    cart_view_query = f"""
                  select p.name, c.quantity, p.price
                  from cart c join product p
                  on c.product_id = p.product_id
                  where c.user_id = {usr_id};
              """

    result3 = read_query(con, cart_view_query)
    from_db3 = []
    for res in result3:
        # print(res)
        res = list(res)
        res[2] = res[2] * res[1]
        from_db3.append(res)
    if len(from_db3) == 0:
        print("Your cart is empty !! \n")
        return from_db3
    total = 0
    for item in from_db3:
        total += item[2]
    cart_total = total
    from_db3.append(["Order Cart", " ", " "])
    from_db3.append(["Clear Cart", " ", " "])
    from_db3.append(["Go Back", " ", " "])
    column2 = ["Product_Name", 'Quantity', 'Price']
```

```
    df2 = pd.DataFrame(from_db3, columns=column2)
    print("Your Cart : \n")
    print(df2)

    print(f"\n------------------------------\nTotal    :    {cart_total}\n------------------------------\n")
    return from_db3


def view_inventory(con):
    pd.set_option('display.max_columns', None)
    print("Petpals Inventory\nsorted Descending by rating")
    q_to_view_all_products = """
    select Name, Brand, Price, Quantity
    from product
    order by rating DESC;
    """
    inventory = read_query(con, q_to_view_all_products)

    inventory1 = []
    for res in inventory:
        # print(res)
        res = list(res)
        inventory1.append(res)
    inventory1.append(["Go Back", " ", " ", " "])
    column2 = ["Product_Name", 'Brand', 'Price', 'Quantity']
    df2 = pd.DataFrame(inventory1, columns=column2)
    display(df2)
    while 2 > 1:
        usr_inp = int(input("Enter the product number to view description : "))
        if usr_inp < 0 or usr_inp > len(inventory):
            print("Invalid Input !!")
        elif usr_inp == len(inventory):
            break
        else:
            q_to_view_description = f"""
                select description from product where name = '{inventory1[usr_inp][0]}';
                """
            description = read_query(con, q_to_view_description)
            print(f"{inventory1[usr_inp][0]} - {description[0][0]}")


con = create_db_connnection(hname, uname, pas, "db")

while 2 > 1:
    print("0 Login as admin\n1 Login as customer\n2 Close App")
    usr_inp = int(input(">>  "))
```

```
if usr_inp == 0:
    view_inventory(con)
elif usr_inp == 1:

    while 2 > 1:
        print("Email :- ")
        email = input()
        print("Password :- ")
        passwd = input()
        q_to_check_email = f"""
            select user_id from credentials
            where email = '{email}' and password = '{passwd}';
        """
        res_of_q = read_query(con, q_to_check_email)
        if len(res_of_q) == 0:
            print("Invalid Credentials !!")
            break

        usr_id = int(res_of_q[0][0])
        # usr_id = 1/

        q_to_getname = f"""
                select CONCAT(u.first_name, ' ', u.last_name)  from user u
                where user_id = {usr_id};
            """
        res_of_q_to_getname = read_query(con, q_to_getname)
        name_of_user = res_of_q_to_getname[0][0]
        # print("-----------  ")
        print(f"\nWelcome {name_of_user}")

        while 2 > 1:
            print("0 Continue Shopping\n1 Go to cart\n2 Logout")
            usr_inp = int(input(">> "))
            if usr_inp == 0:
                # print("\n==========================\n\n")
                while 2 > 1:
                    print("Select the product type you want to shop for : \n")

                    q = """
                    select distinct(Product_type) from product;
                    """

                    result = read_query(con, q)
                    # print(result)
                    # for res in result:
                    # print(res)
```

```
from_db = []
for res in result:
    res = list(res)
    from_db.append(res)
from_db.append(["Go Back"])
column = ["Product Type"]
df = pd.DataFrame(from_db, columns=column)
#
display(df)

usr_inp = int(input(">> "))
if usr_inp == len(from_db) - 1:
    break
elif usr_inp > len((from_db)) - 1:
    print("Invalid Input !!")
    continue
str_pr_type = str(from_db[usr_inp][0])

q2 = f"""
select Name, Brand, Pet_category, Price, Quantity
from product
where Product_type = '{str_pr_type}' and quantity > 0;
"""

result2 = read_query(con, q2)
from_db2 = []
for res in result2:
    # print(res)
    res = list(res)
    from_db2.append(res)
from_db2.append(["Go Back", " ", " ", " ", " "])
column2 = ["Name", 'Brand', 'Pet_Type', 'Price', 'Available_Units']
df2 = pd.DataFrame(from_db2, columns=column2)
print(df2)
# print(from_db2)
# display(df2)
length = len(from_db2)
# print(f"{length}  Go Back")
q3 = ""
usr_inp = int(input(">> "))
if usr_inp >= len(from_db2) or usr_inp < 0:
    print("Invalid Input !!")
elif usr_inp == len(from_db2) - 1:
    break
else:
```

```
            quant = int(input("Quantity to buy (max 10) >> "))
            if quant > 10 or quant < 0:
                print("Invalid Quantity")
            elif quant > int(from_db2[usr_inp][4]):
                print(f"Only {from_db2[usr_inp][4]} quantity available")
            else:
                temp_q = f"""
                select product_id from product
                where name = '{from_db2[usr_inp][0]}';"""
                # print(temp_q)
                temp_res = read_query(con, temp_q)
                temp_q2 = f"""
                select * from cart where product_id = {temp_res[0][0]} and user_id = {usr_id};
                """
                temp_res2 = read_query(con, temp_q2)
                # print(temp_res2)
                if len(temp_res2) != 0:
                    print("Item already in your cart, to edit go to Cart\n")
                else:
                    # print(temp_res[0][0])
                    q3 = f"""
                        Insert into cart (User_id, Product_ID, Quantity) VALUES
                        ({usr_id}, {temp_res[0][0]}, {quant})
                        """
                    execute_querries(con, q3)
                    print("Product added to cart !!")
                    # print(read_query(con, """select * from cart;"""))


elif usr_inp == 1:

    while 2 > 1:
        cart = view_cart(usr_id, con)
        # print(cart)
        if len(cart) > 0:
            print("Choose product you want to edit :")
            usr_inp = int(input())
            if usr_inp < 0 or usr_inp >= len(cart):
                print("Invalid input !!")
            elif 0 <= usr_inp <= len(cart) - 4:
                selected_prod = cart[usr_inp][0]
                print(selected_prod)
                quan_in_cart = cart[usr_inp][1]
                print(quan_in_cart)
                temp_q2 = f"""
```

```
                    select p.product_id, p.quantity from cart c join product p on p.product_id =
c.product_id
                    where user_id = {usr_id} and name = '{selected_prod}';
                   """
                res_temp_q2 = read_query(con, temp_q2)
                curr_prod_id = res_temp_q2[0][0]
                available_quantity = res_temp_q2[0][1]
                flag = 0
                while 2 > 1:

                  if flag == 1:
                    break
                  # flag = 0
                  print(f"\nSelected Product - {selected_prod}\nQuantity in cart - {quan_in_cart}")
                  print("\n0 Remove selected product")
                  print("1 Edit quantity")
                  print("2 Go back")
                  usr_inp = int(input(">> "))
                  if usr_inp == 0:
                    temp_q3 = f"""
                        Delete from cart
                        where user_id = {usr_id} and product_id = {curr_prod_id};
                        """
                    execute_querries(con, temp_q3)
                    print("Deletion Successful !!")

                    break
                  elif usr_inp == 1:
                    while 2 > 1:
                      temp_inp = int(input(f"Enter the updated quantity you want to order (max "
                                   f"10)\nAvailable "
                                   f"quantities {available_quantity} : "))
                      if temp_inp <= 0 or temp_inp > 10:
                        print("Invalid input !!... Maximum 10 quantities allowed")
                      elif temp_inp > available_quantity:
                        print(f"Only {available_quantity} quantities are available")

                      else:
                        temp_q3 = f"""
                            Update cart
                            set quantity = {temp_inp} where
                            user_id = {usr_id} and product_id = {curr_prod_id};
                            """
                        execute_querries(con, temp_q3)
                        print("Cart updated !!")
                        flag = 1
```

```
                    break
                elif usr_inp == 2:
                    break
            elif usr_inp == len(cart) - 3:
                print(f"Your bill is {bill(cart)}")
                payment_modes_list = ['COD', 'Credit Card', 'Debit Card', 'UPI', 'Others']
                while 2 > 1:
                    print("Pay via : \n"
                          "0 COD\n"
                          "1 Credit Card\n"
                          "2 Debit Card\n"
                          "3 UPI\n"
                          "4 Others\n"
                          "5 Cancel Payment\n")

                    usr_inp = int(input(">> "))
                    payment_mode = ''
                    if usr_inp < 0 or usr_inp > 5:
                        print("Invalid Input !!")
                    elif usr_inp == 5:
                        print("Transaction Cancelled !!")
                        break
                    else:
                        payment_mode = payment_modes_list[usr_inp]

                        for i in range(len(cart) - 3):
                            q_to_get_prod_id = f"""select product_id from product where name =
'{cart[i][0]}'"""
                            res_of_q_to_prod_id = read_query(con, q_to_get_prod_id)
                            curr_prod_id = res_of_q_to_prod_id[0][0]
                            q_to_get_order_id = f"""
                                        Select MAX(order_id) from product_order;
                                        """
                            order_id = read_query(con, q_to_get_order_id)
                            new_order_id = order_id[0][0] + 1

                            q_to_get_payment_id = f"""
                                        Select MAX(Payment_id) from payment_and_history;
                                        """
                            payment_id = read_query(con, q_to_get_payment_id)
                            new_payment_id = payment_id[0][0] + 1

                            q_to_insert_in_product_order = f"""
                            INSERT INTO product_order (Order_id, Status, Quantity, Order_Date, User_ID,
product_id)
                            VALUES
```

```
                                ({new_payment_id},'Delivered', {cart[i][1]}, CURDATE(), {usr_id},
{curr_prod_id});
                                """
                                q_to_insert_in_paayment_and_history = f""" INSERT INTO payment_and_history
(
                                Payment_id, Amount, Payment_mode, Order_type, Payment_Date,
Product_Order_ID,
                                Service_Order_ID) VALUES ({new_payment_id}, {cart[i][2]}, '{payment_mode}',
'Product', CURDATE(),
                                {curr_prod_id}, NULL);
                                  """
                                q_to_decrease_products_in_database = f"""
                                  Update product
                                  set quantity = quantity - {cart[i][1]}
                                  where product_id = {curr_prod_id};
                                  """
                                execute_querries(con, q_to_insert_in_product_order)
                                execute_querries(con, q_to_insert_in_paayment_and_history)
                                execute_querries(con, q_to_decrease_products_in_database)

                            print("Order Placed !!")
                            clear_cart(usr_id)
                            break

                        elif usr_inp == len(cart) - 2:
                            clear_cart(usr_id)
                            print("Cart Cleared !!")
                            break
                        elif usr_inp == len(cart) - 1:
                            break
                    else:
                        break
                elif usr_inp == 2:
                    break

            break
    elif usr_inp == 2:
        print("Thank you for using Petpals.")
        break
    else:
        print("Invalid Input !!")
```

## Triggers:

1. **Inventory Management Trigger: Automatically reorder products when inventory levels fall below a certain threshold.**

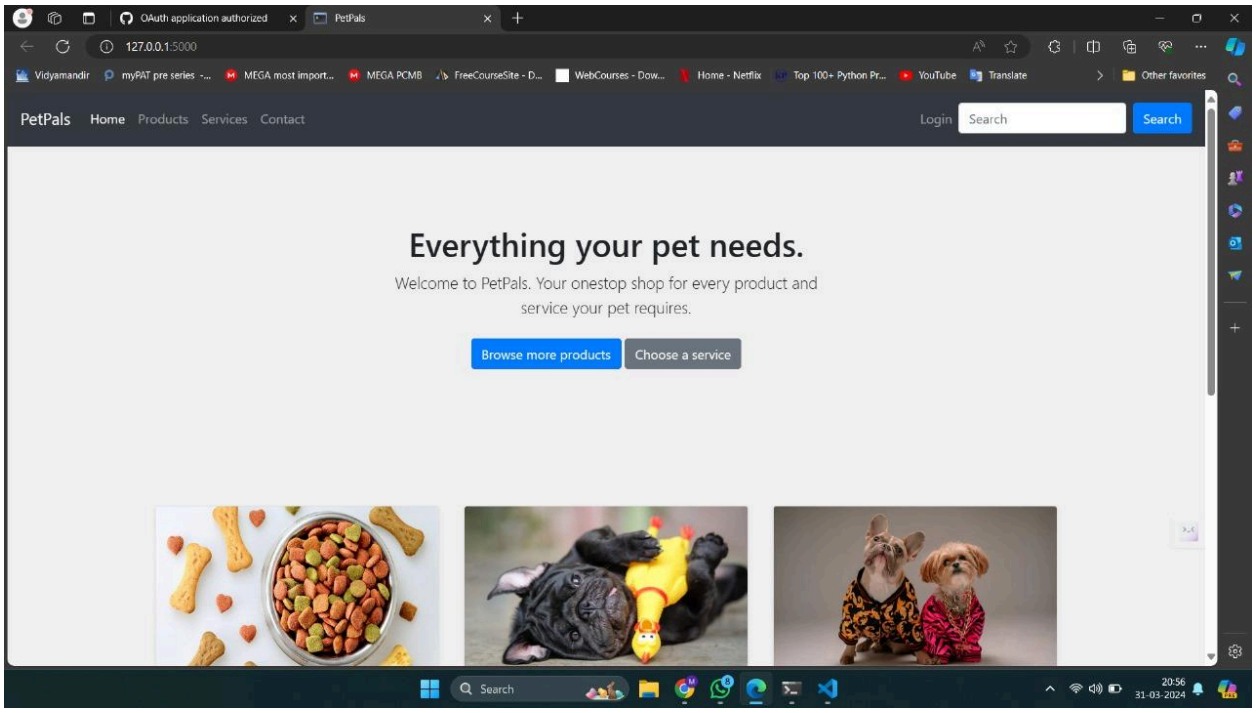
```
CREATE TABLE reorder_history (
   reorder_id INT AUTO_INCREMENT PRIMARY KEY,
   product_id INT,
   reorder_quantity INT,
   reorder_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (product_id) REFERENCES product(Product_ID)
);

DROP TRIGGER IF EXISTS reorder_trigger;

DELIMITER //

CREATE TRIGGER reorder_trigger
BEFORE UPDATE ON product
FOR EACH ROW
BEGIN
   IF NEW.Quantity < 5 THEN
      -- Reorder 20 quantities if the current quantity falls below 5
      SET NEW.Quantity = NEW.Quantity + 20;

      -- Insert a record into reorder_history
      INSERT INTO reorder_history (product_id, reorder_quantity)
      VALUES (NEW.Product_ID, 20);
   END IF;
END;
//

DELIMITER ;
```

**2. Product order Trail Trigger : Stores audit for each product sold**

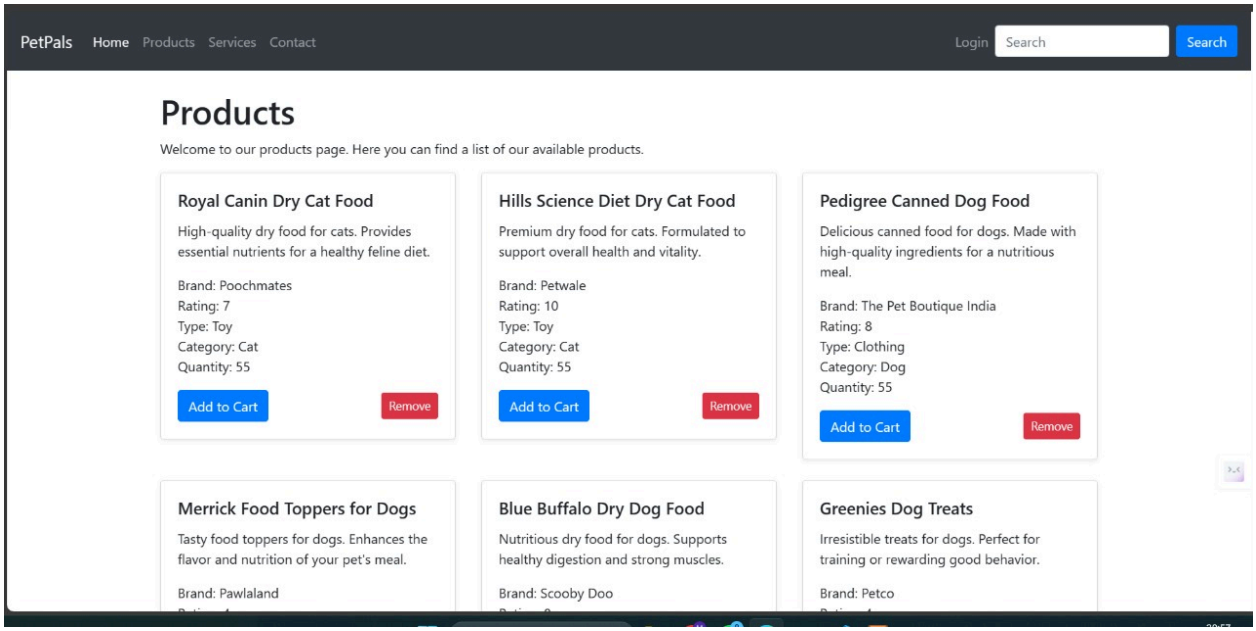
```
CREATE TABLE product_order_audit (
   audit_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT,
   operation_type VARCHAR(10),
   operation_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id INT,
   details TEXT
);


DELIMITER //
CREATE TRIGGER product_order_audit_trigger
AFTER INSERT ON product_order
FOR EACH ROW
BEGIN
   INSERT INTO product_order_audit (order_id, operation_type, user_id, details)
   VALUES (NEW.order_id, 'INSERT', NULL, CONCAT('New order created with ID ',
NEW.order_id));
END;
//

CREATE TRIGGER product_order_audit_trigger_update
AFTER UPDATE ON product_order
FOR EACH ROW
BEGIN
   INSERT INTO product_order_audit (order_id, operation_type, user_id, details)
   VALUES (NEW.order_id, 'UPDATE', NULL, CONCAT('Order with ID ', NEW.order_id, '
updated'));
END;
//

CREATE TRIGGER product_order_audit_trigger_delete
AFTER DELETE ON product_order
FOR EACH ROW
BEGIN
   INSERT INTO product_order_audit (order_id, operation_type, user_id, details)
   VALUES (OLD.order_id, 'DELETE', NULL, CONCAT('Order with ID ', OLD.order_id, ' deleted'));
END;
//
DELIMITER ;
```

## Front-End:

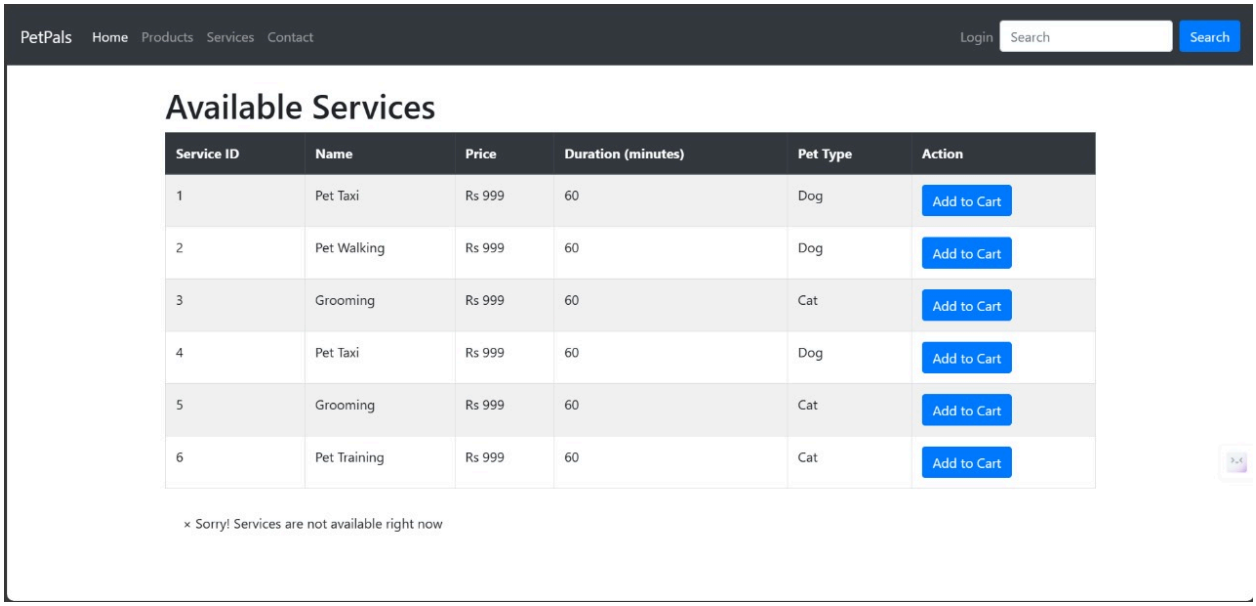
1. Home Page



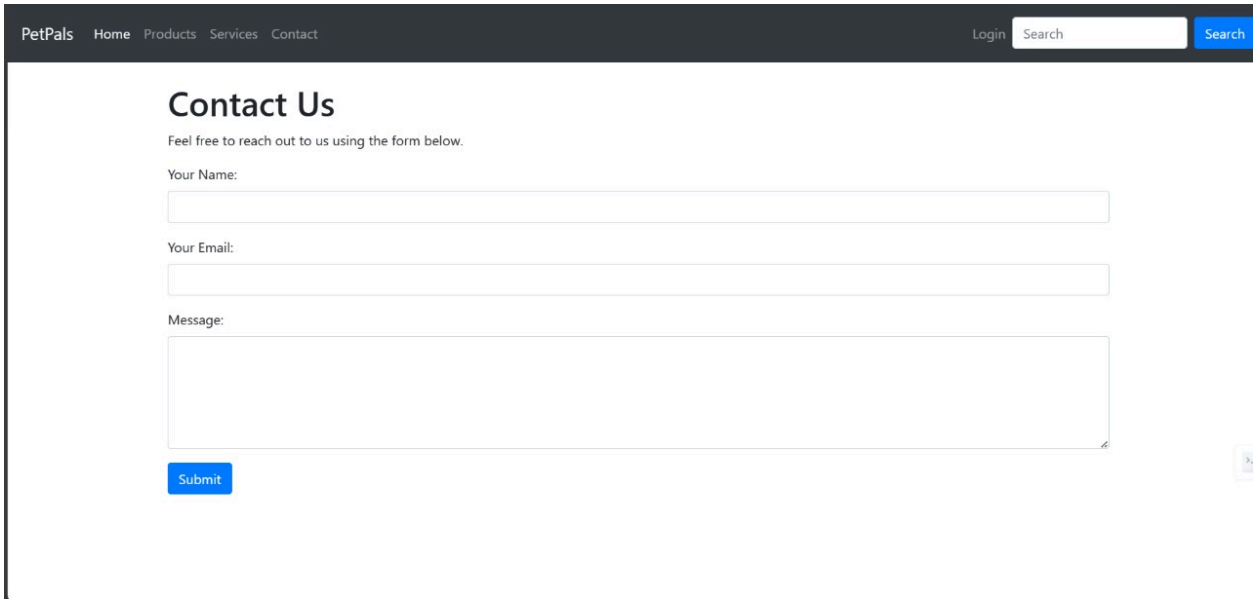

2. Products Page



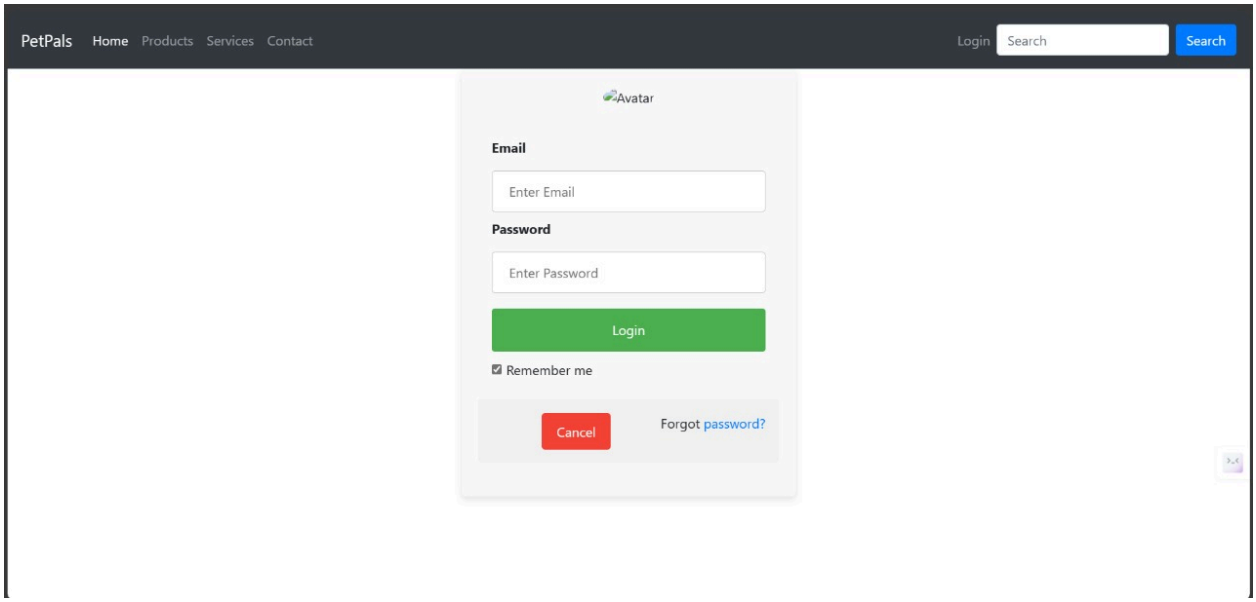

3. Services Page



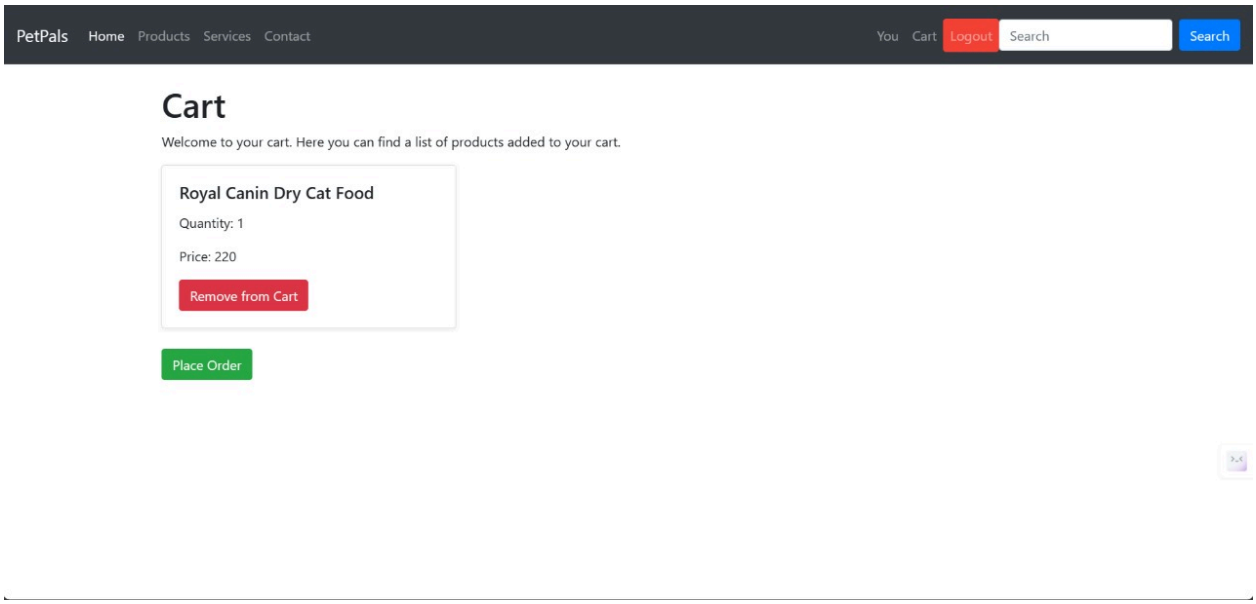

4. Contacts Page



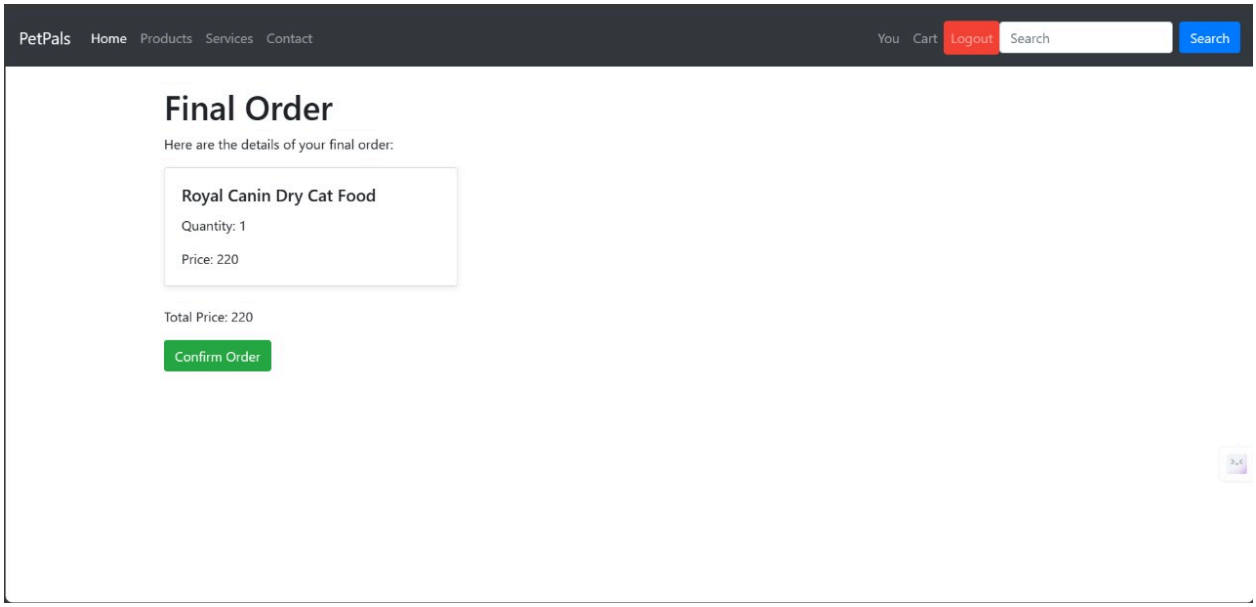

5. Login Page


PetPals
Home
Products
Services
Contact
Login
Search
Search
Avatar
Email
Enter Email
Password
Enter Password
Login
Remember me
Cancel
Forgot password?

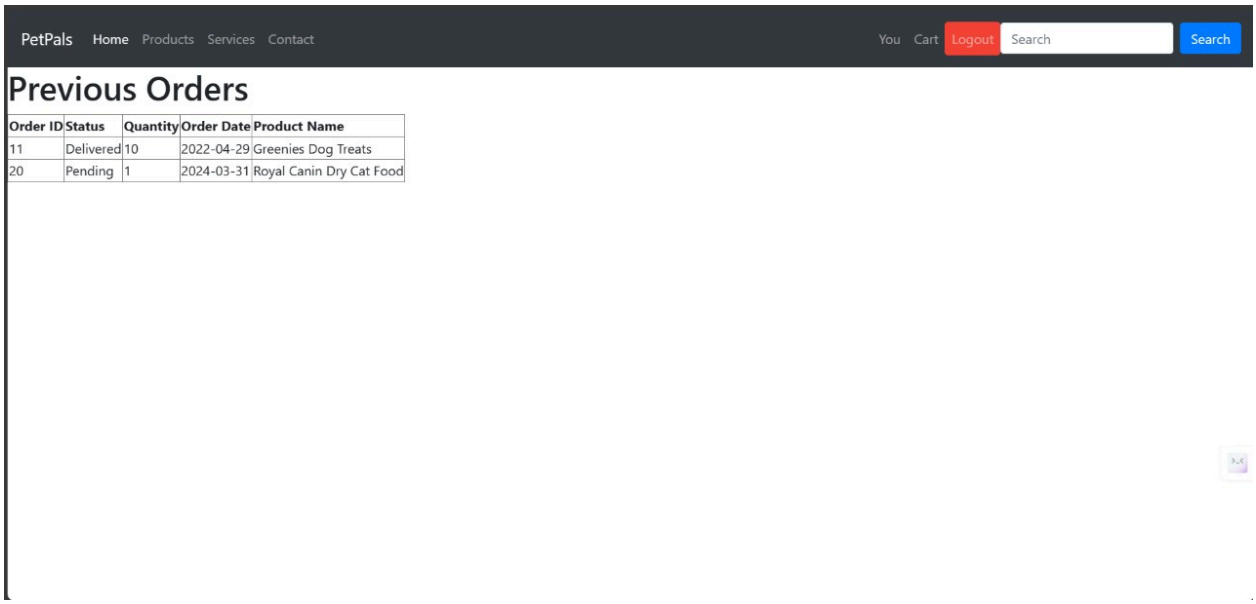
6. Cart Page


PetPals
Home
Products
Services
Contact
You
Cart
Logout
Search
Search
Cart
Welcome to your cart. Here you can find a list of products added to your cart.
Royal Canin Dry Cat Food
Quantity: 1
Price: 220
Remove from Cart
Place Order

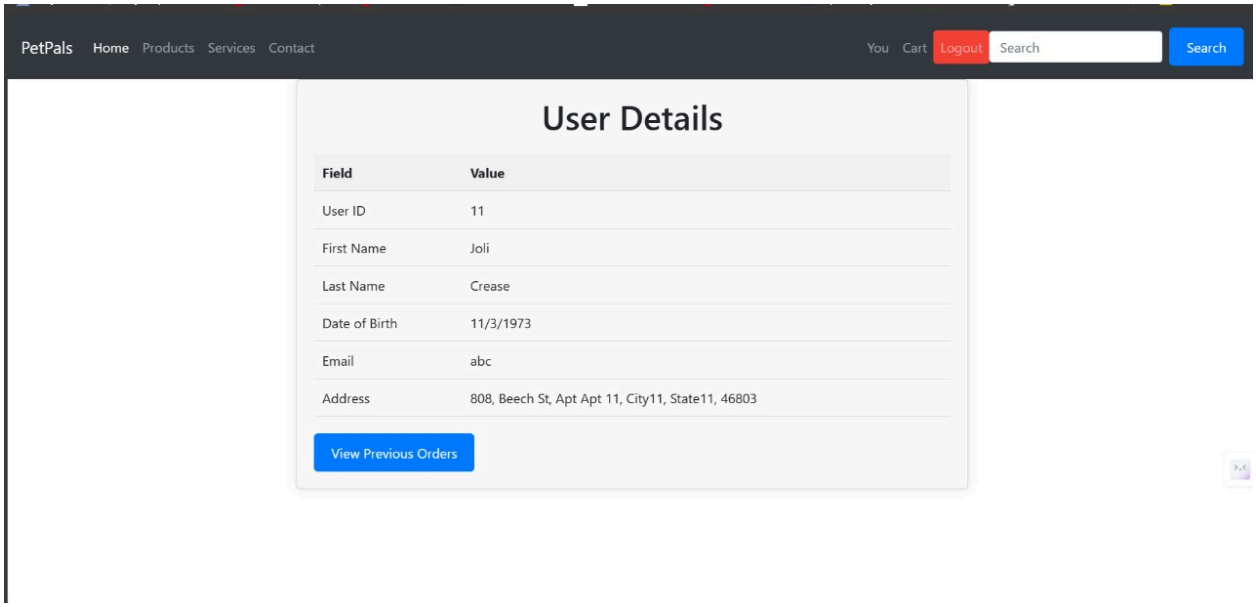
7. Confirmation Page



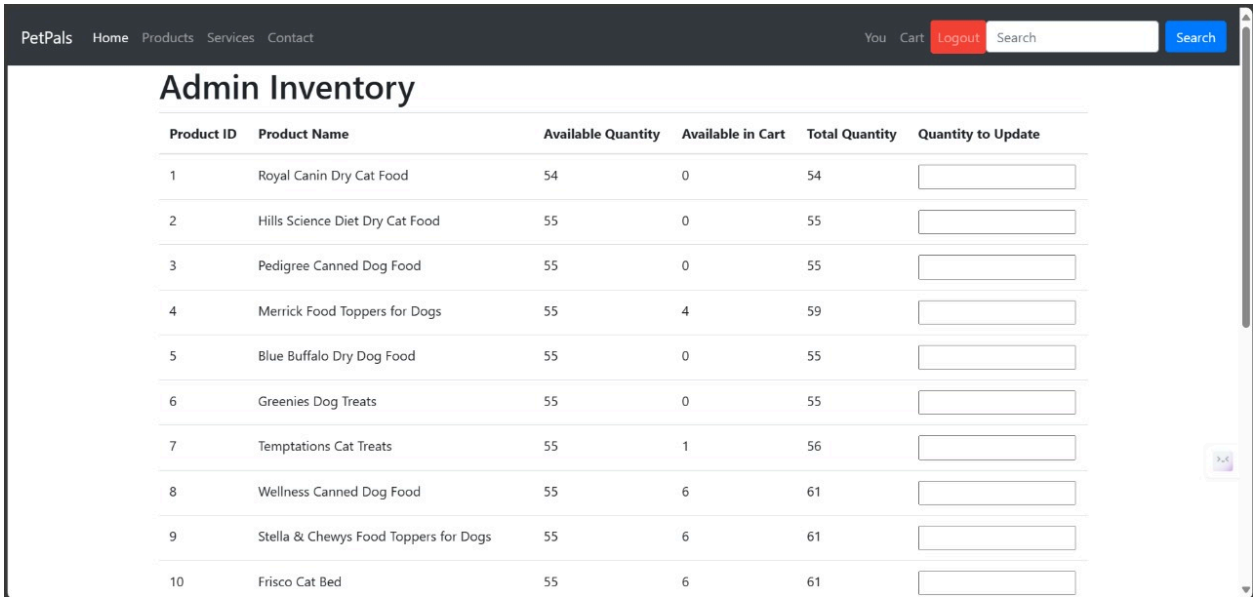

8. Previous Orders Page



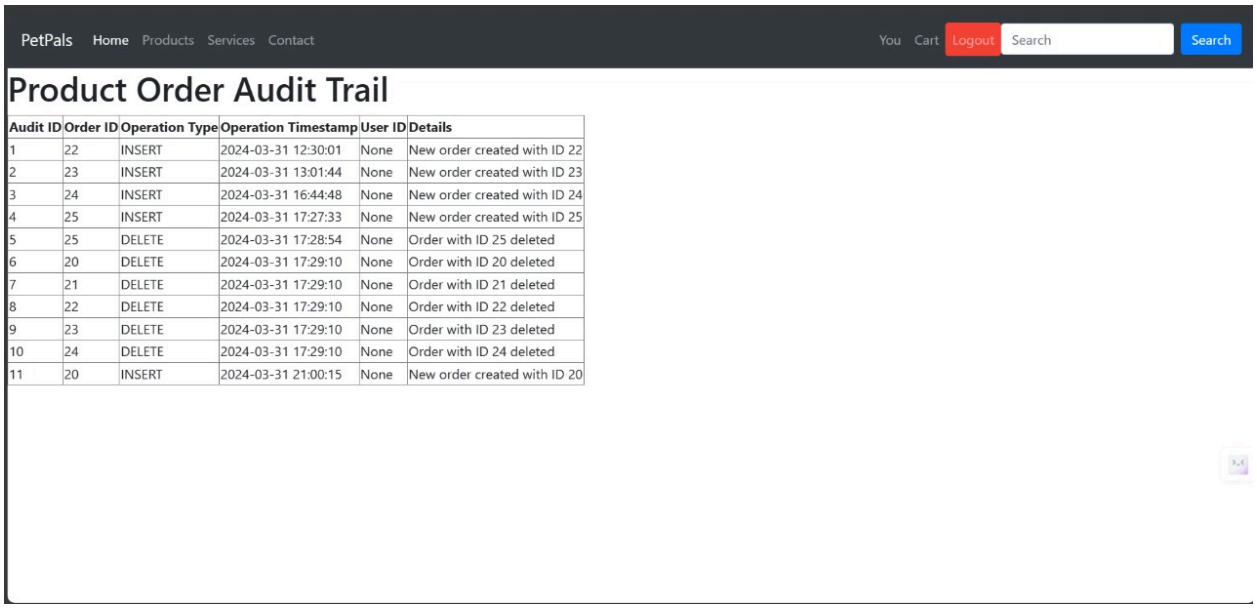
9. User Details Page




10. Admin Inventory Page



11. Product order audit trail record Page




| Audit ID | Order ID | Operation Type | Operation Timestamp | User ID | Details |
|---|---|---|---|---|---|
| 1 | 22 | INSERT | 2024-03-31 12:30:01 | None | New order created with ID 22 |
| 2 | 23 | INSERT | 2024-03-31 13:01:44 | None | New order created with ID 23 |
| 3 | 24 | INSERT | 2024-03-31 16:44:48 | None | New order created with ID 24 |
| 4 | 25 | INSERT | 2024-03-31 17:27:33 | None | New order created with ID 25 |
| 5 | 25 | DELETE | 2024-03-31 17:28:54 | None | Order with ID 25 deleted |
| 6 | 20 | DELETE | 2024-03-31 17:29:10 | None | Order with ID 20 deleted |
| 7 | 21 | DELETE | 2024-03-31 17:29:10 | None | Order with ID 21 deleted |
| 8 | 22 | DELETE | 2024-03-31 17:29:10 | None | Order with ID 22 deleted |
| 9 | 23 | DELETE | 2024-03-31 17:29:10 | None | Order with ID 23 deleted |
| 10 | 24 | DELETE | 2024-03-31 17:29:10 | None | Order with ID 24 deleted |
| 11 | 20 | INSERT | 2024-03-31 21:00:15 | None | New order created with ID 20 |