NSBM Green University

Faculty of Computing

BSc (Hons) of Data Science

CS304.3 Advanced Database Management Systems

Module Lecturer: Mr. S.Naji

Responsive Restaurant Front-End System and Admin Dashboard For Hotel 'JAGABAY'

Group B

Student ID	Name
24572	MKIM Rohana
24614	GAAS Ganegoda
25180	PGJ Lakshani
24921	WKS Geshan
24490	MRK Karunathilaka
25561	TD Athulathmudali
24585	SSD Rosairo
25011	HHN Peiris

Table of Contents

Section 01

1. Introduction

Hotel 'Jagabay' is renowned for its exquisite hospitality and exceptional dining experiences. To further enhance its services, we have developed a comprehensive responsive restaurant front-end system and an administrative dashboard.

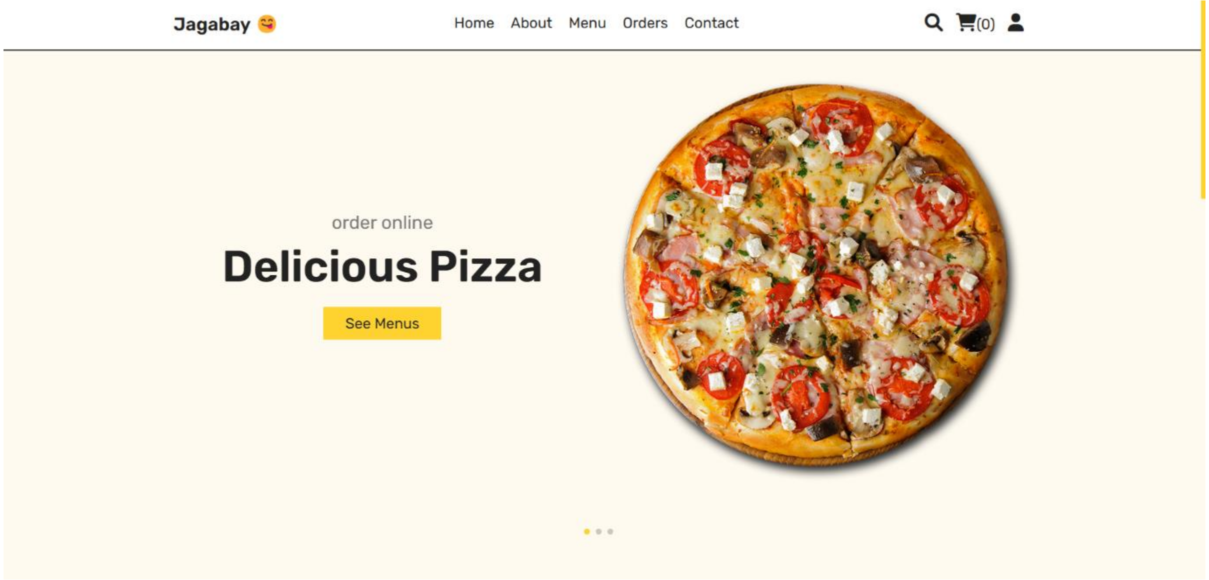
The restaurant front-end system is designed for customers to browse the menu, place orders, and make reservations, while the admin dashboard allows for efficient management of orders and customer interactions. This report details the development process, features, and implementation of these systems.

2. Project Overview

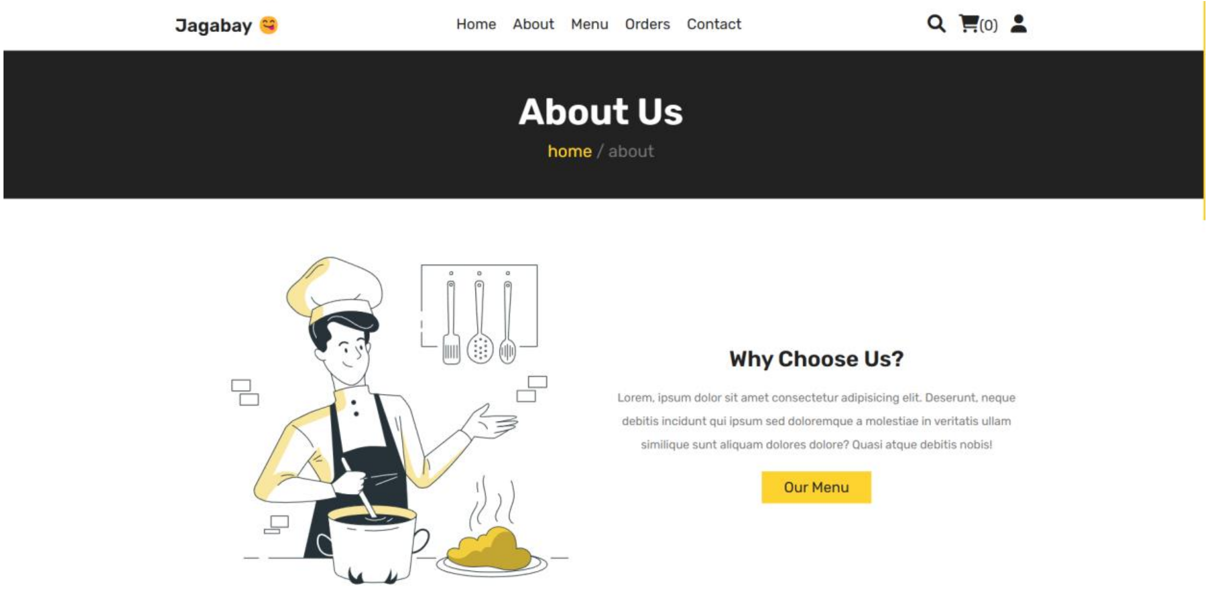
2.1. Hotel 'Jagabay' Restaurant Front-End System.

The restaurant's front-end system is crafted to offer a seamless user experience, allowing customers to interact with the restaurant's offerings effortlessly. The design is fully responsive, ensuring compatibility across various devices and screen sizes.

- **Home Page.**
    - Food categories and latest dishes
    - Quick access to ordering options



- **About Page.**
  - Detailed information about Hotel 'Jagabay' and its staff



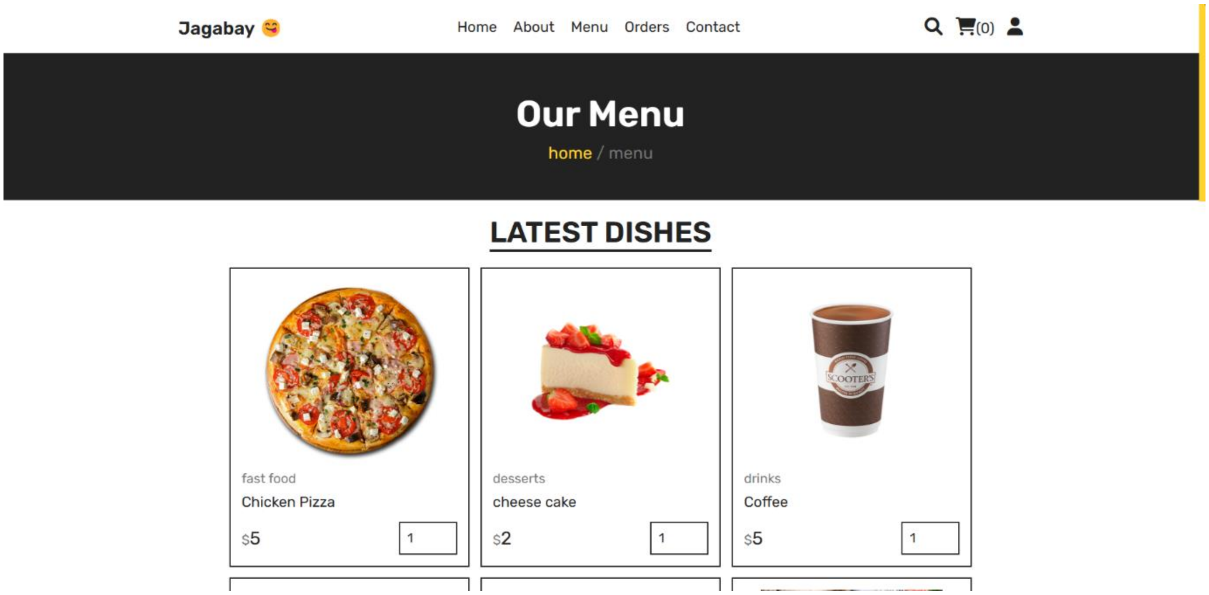
About Us

home / about

Why Choose Us?

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Deserunt, neque debitis incidunt qui ipsum sed doloremque a molestiae in veritatis ullam similique sunt aliquam dolores dolore? Quasi atque debitis nobis!

- **Menu Page:**
  - Comprehensive listing of restaurant menu items
  - Direct ordering functionality



Our Menu

home / menu

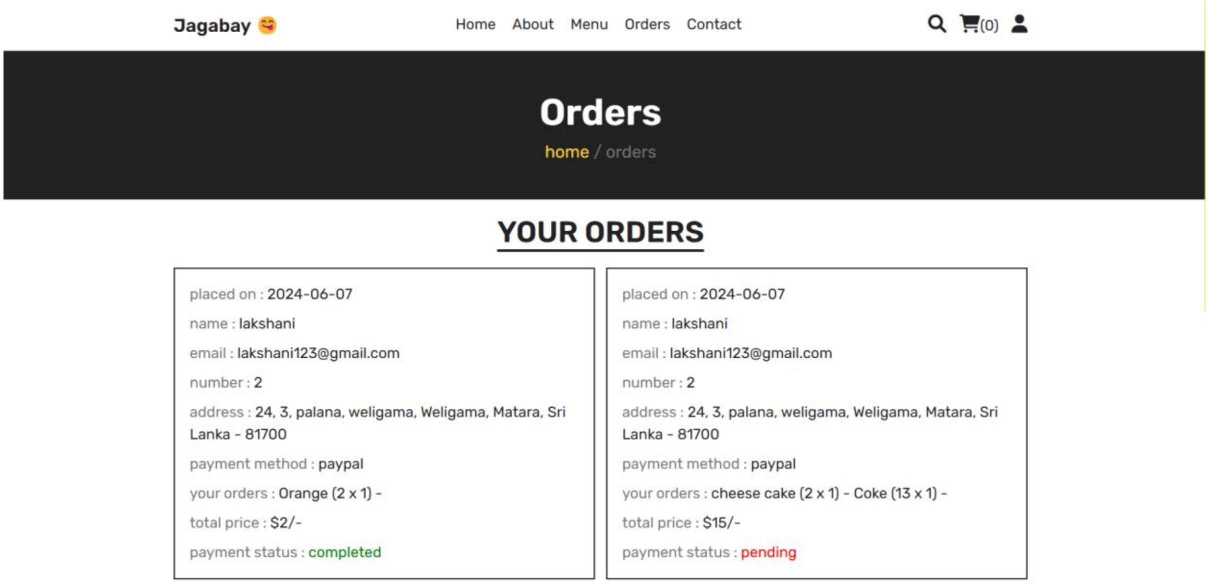
LATEST DISHES

fast food
Chicken Pizza
$5 1

desserts
cheese cake
$2 1

drinks
Coffee
$5 1

- **Orders Page.**
  - View all orders placed
  - Order details and status (pending/complete)



Orders

home / orders

YOUR ORDERS

placed on : 2024-06-07

name : lakshani

email : lakshani123@gmail.com

number : 2

address : 24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

payment method : paypal

your orders : Orange (2 x 1) -

total price : $2/-

payment status : completed

placed on : 2024-06-07

name : lakshani

email : lakshani123@gmail.com

number : 2

address : 24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

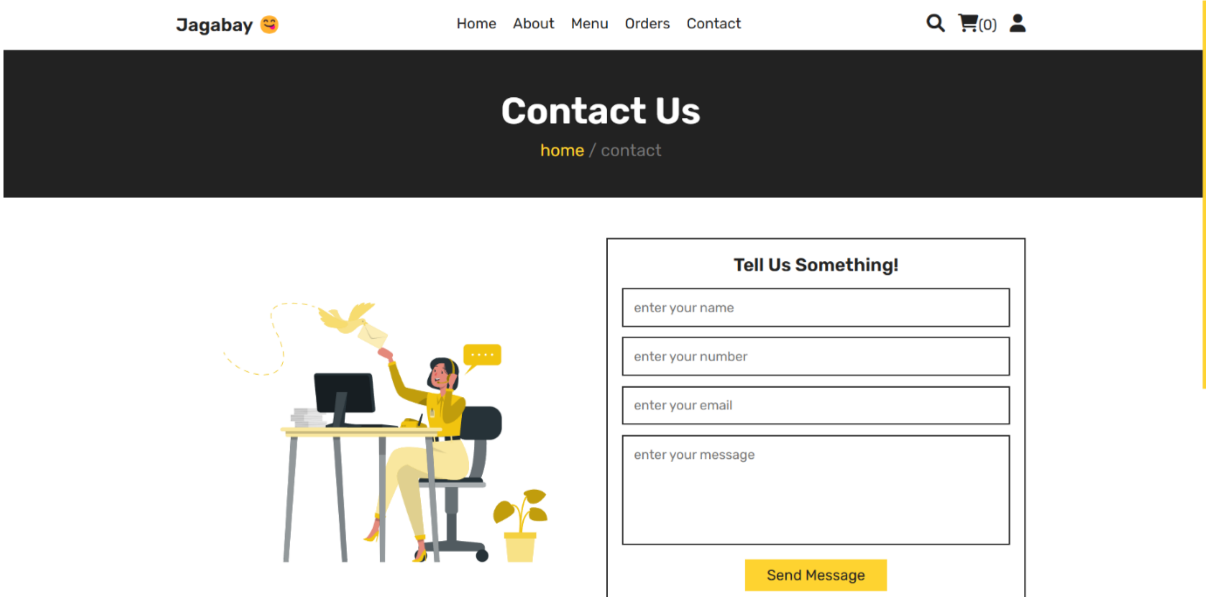
payment method : paypal

your orders : cheese cake (2 x 1) - Coke (13 x 1) -

total price : $15/-

payment status : pending

- **Contact Page.**
  - Contact details and messaging functionality for users

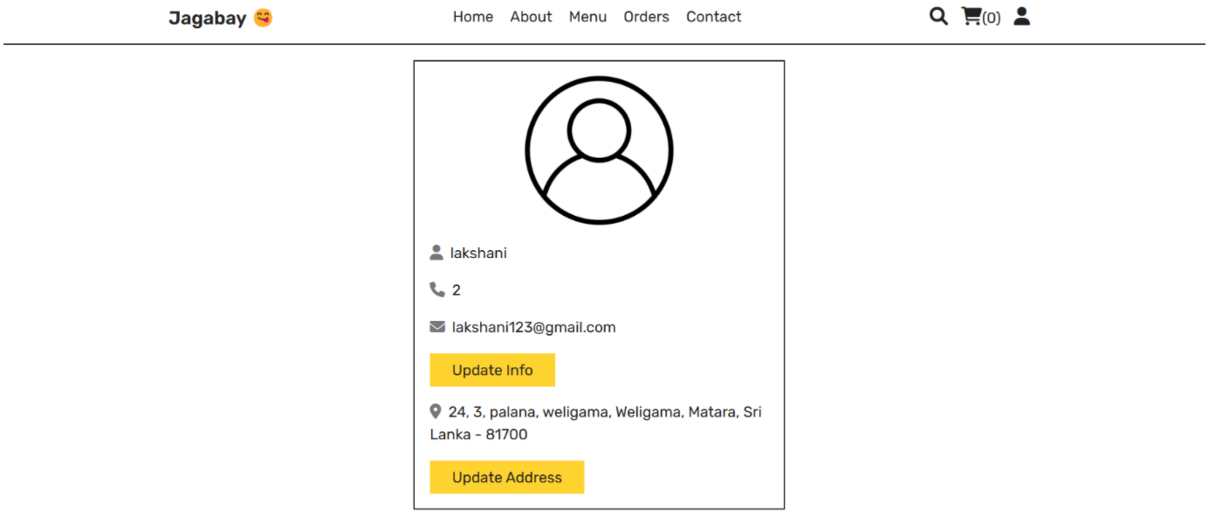


Contact Us

home / contact

Tell Us Something!

- **User Account Management.**
  - Login and registration functionality
  - Update user information
  - Update info and update address buttons for quick access



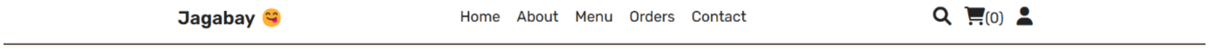
lakshani

2

lakshani123@gmail.com

24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

- **Navigation Bar.**
  - Search functionality to find dishes quickly
  - Cart icon to view and finalize orders



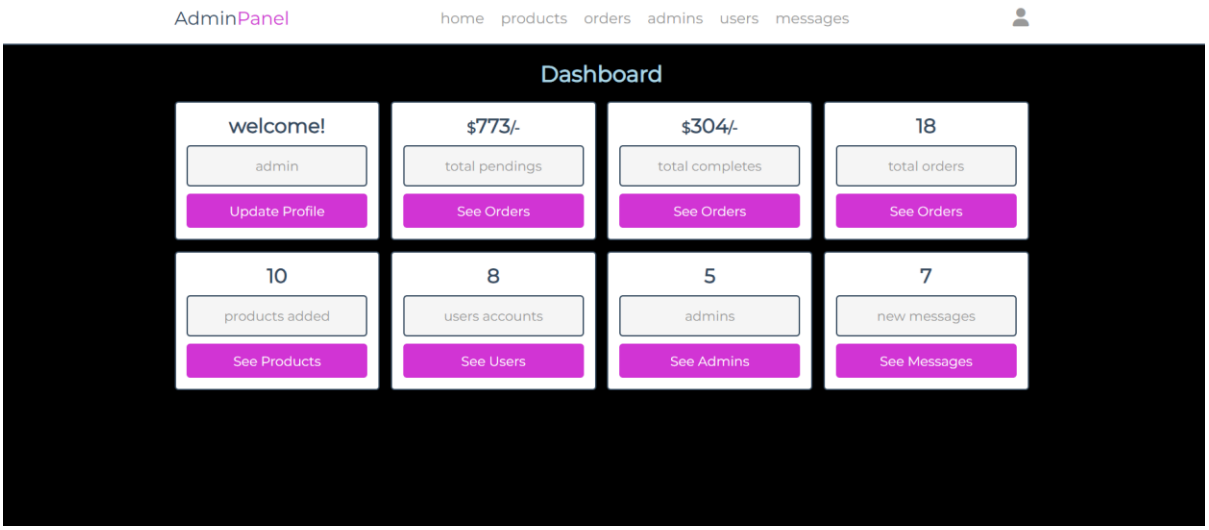
2.2. Admin Dashboard.

The admin dashboard provides a comprehensive overview of restaurant operations, including order management and customer interactions. It enables waiters to add customer orders and admins to view a summary of all activities, enhancing operational efficiency.

- **Dashboard Overview.**
  - ➢ Total Spending Orders
  - ➢ Total Completed Orders
  - ➢ Total Orders
  - ➢ Number of Products Added
  - ➢ Number of User Accounts
  - ➢ Number of Admins
  - ➢ Number of New Messages
- **Product Management.**
  - ➢ Product Page for adding new menu items
  - ➢ View orders placed and their details

- **Admin Management.**
  - ➢ Admin page for managing admin accounts
  - ➢ Add new admin functionality

- **User Management.**
  - ➢ User page for managing user accounts
  - ➢ View and respond to user messages



3. Design and Development

Technologies Used.

- **HTML, CSS, Vanilla JavaScript**

For building the front-end user interface and implementing interactive features.

- **PHP PDO**

For secure database connectivity and query execution.

- **MySQL Database**

For storing and managing data related to customers, orders, menu items, and reservations.

Responsive Design.

The website is designed to be fully responsive, utilizing modern CSS techniques such as Grid and Flexbox to ensure a consistent and user-friendly experience across all devices.

4. Implementation Details

4.1. Header Section.

- Implemented using Vanilla JavaScript to provide an intuitive navigation experience.

4.2. Category Section.

- Utilizes CSS Grid to display various food categories in a structured and responsive layout.

4.3. About Section.

- Designed using CSS Flexbox to provide a responsive layout describing the hotel and its restaurant.

4.4. Steps Section.

- Utilizes CSS Grid to provide a step-by-step guide on how to place orders online.

4.5. Reviews Section.

- Displays customer reviews and testimonials using a touch slider.

4.6. Menu Section.

- Uses CSS Grid to showcase the restaurant's menu items in an organized manner.

4.7. Orders Section.

- Designed using CSS Flexbox to display customer orders and their statuses.

4.8. Contact Section.

- Implemented using CSS Flexbox to display contact information and a contact form.

4.9. Search Form Section.

- Designed to allow users to quickly search for menu items.

4.10. Shopping Cart Section.

- Utilizes CSS Grid to display the items in the shopping cart and their total.

4.11. Checkout Section.

- Ensures a user-friendly checkout process with a responsive design.

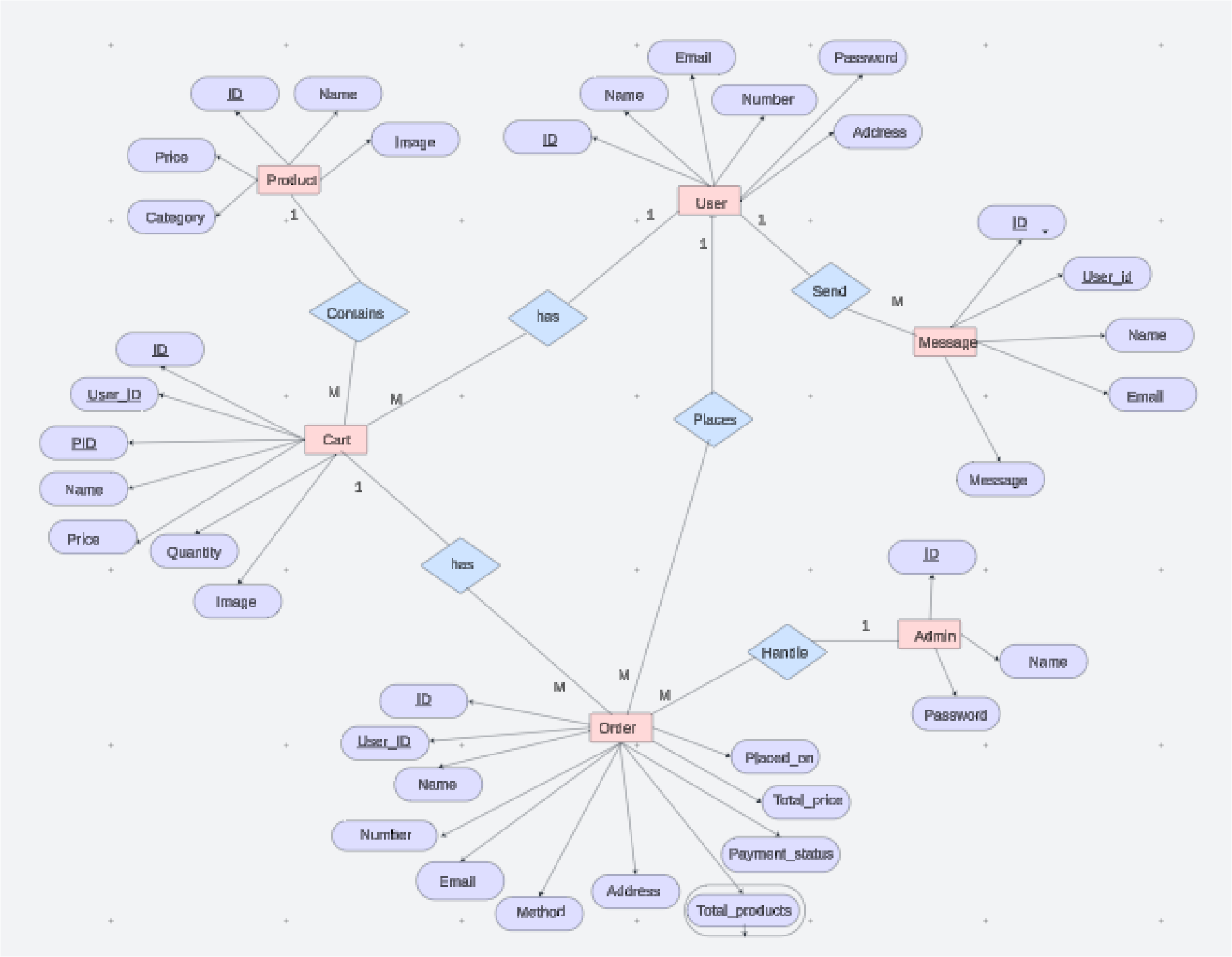
4.12. Profile Sections

- Allows users to view their profile information.
- Provides a form for users to update their profile information.
- Allows users to update their address information.

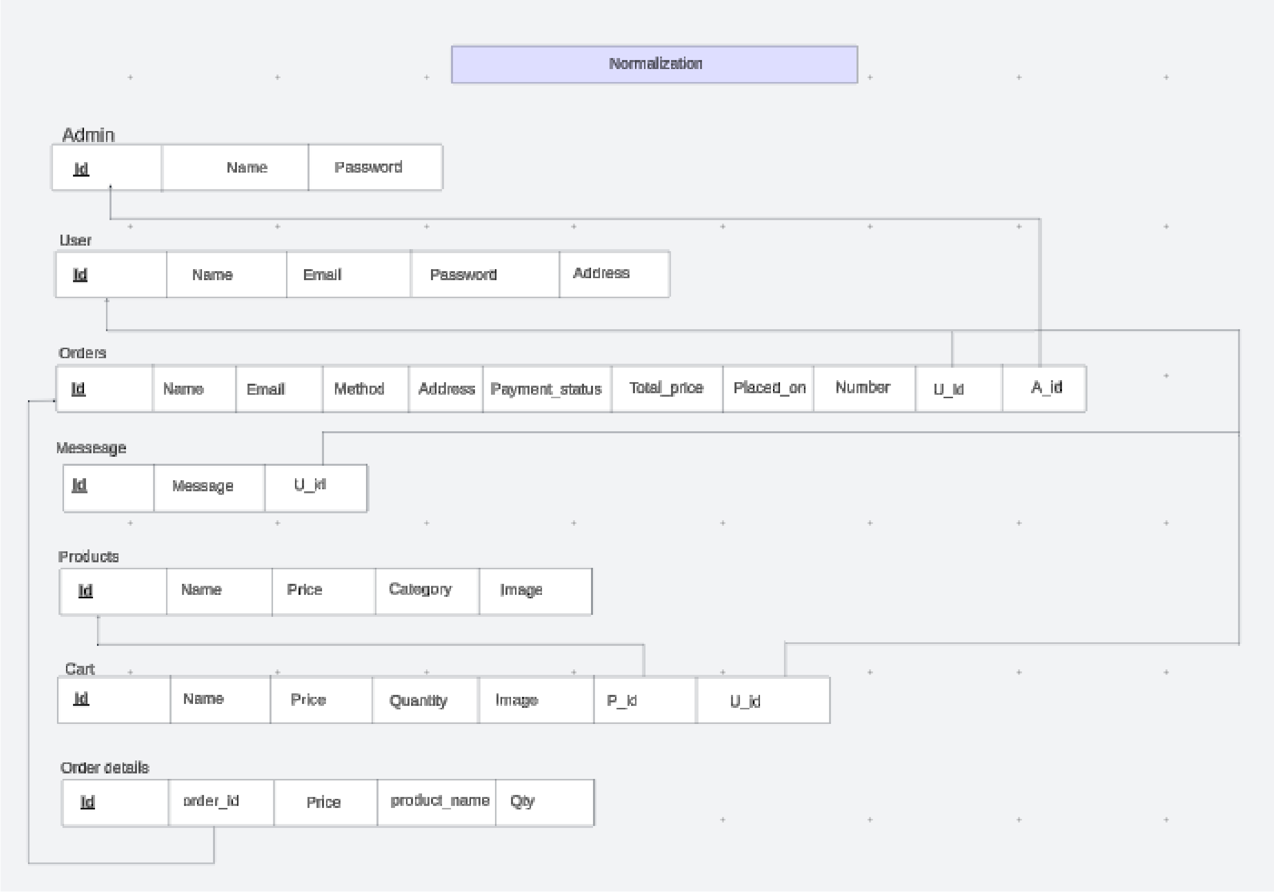
4.13. Footer Section.

- Uses CSS Grid to display links, contact information, and social media links in a structured manner.

5. EER Diagram



6. Normalization



Normalization

Admin

Id	Name	Password

User

Id	Name	Email	Password	Address

Orders

Id	Name	Email	Method	Address	Payment_status	Total_price	Placed_on	Number	U_id	A_id

Messeage

Id	Message	U_id

Products

Id	Name	Price	Category	Image

Cart

Id	Name	Price	Quantity	Image	P_id	U_id

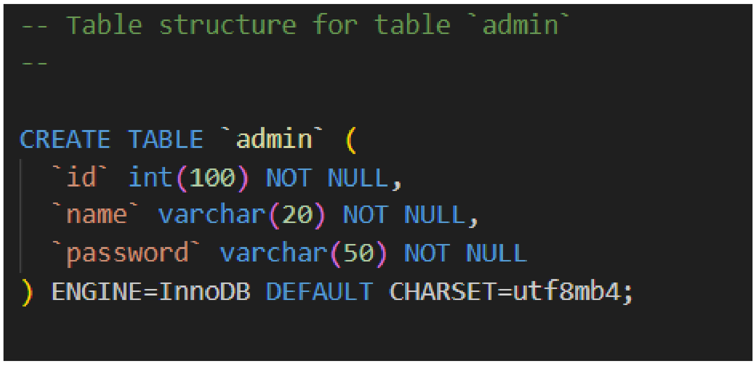
Order details

Id	order_id	Price	product_name	Qty

Section 02

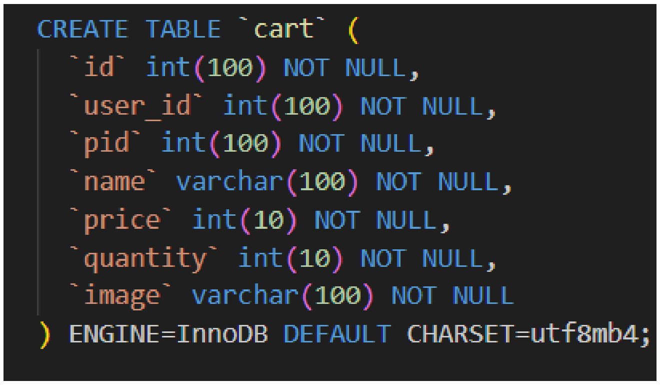
7. Tables

- Admin table



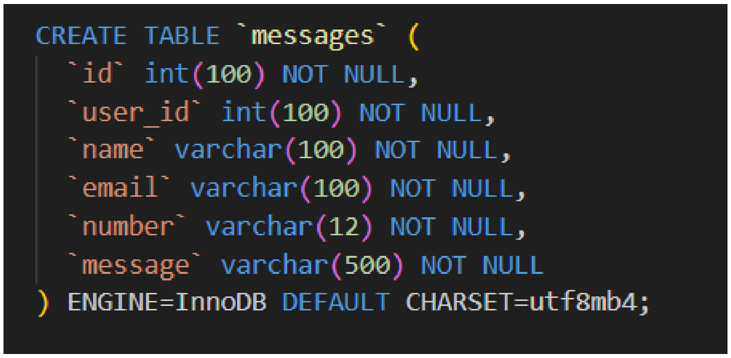
```sql
-- Table structure for table `admin`
--

CREATE TABLE `admin` (
  `id` int(100) NOT NULL,
  `name` varchar(20) NOT NULL,
  `password` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Cart table



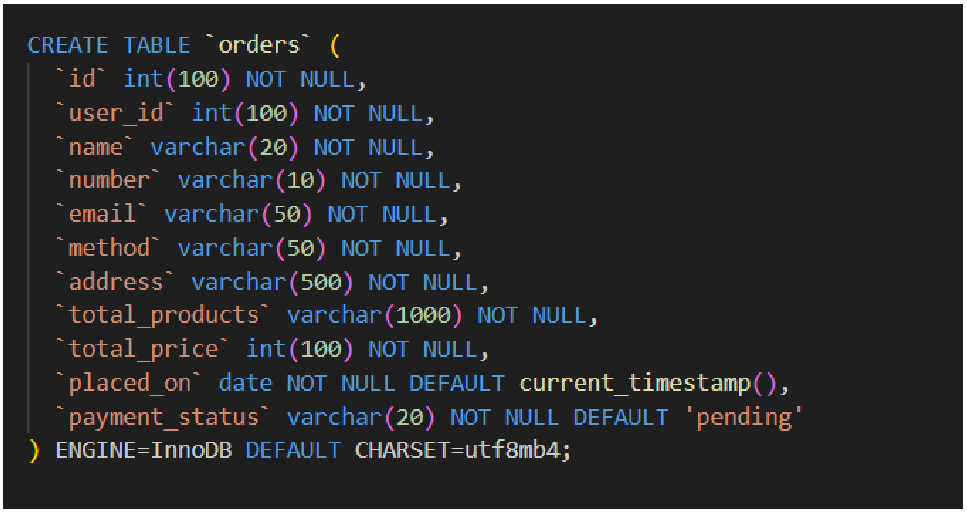
```sql
CREATE TABLE `cart` (
  `id` int(100) NOT NULL,
  `user_id` int(100) NOT NULL,
  `pid` int(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `price` int(10) NOT NULL,
  `quantity` int(10) NOT NULL,
  `image` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Message table



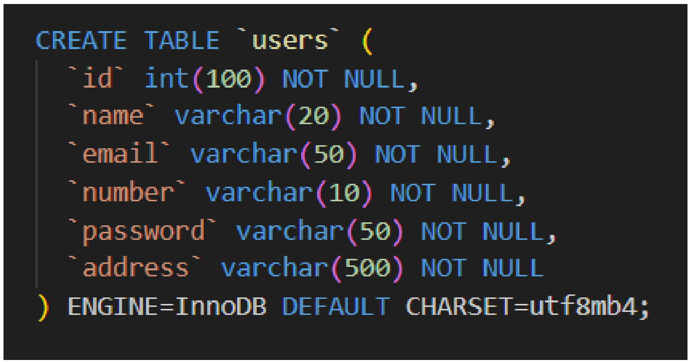
```sql
CREATE TABLE `messages` (
  `id` int(100) NOT NULL,
  `user_id` int(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `number` varchar(12) NOT NULL,
  `message` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Orders table



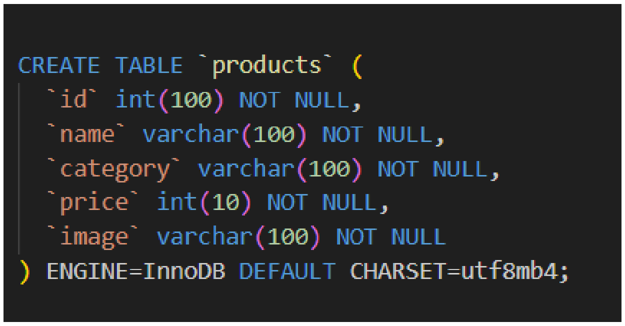
```
CREATE TABLE `orders` (
  `id` int(100) NOT NULL,
  `user_id` int(100) NOT NULL,
  `name` varchar(20) NOT NULL,
  `number` varchar(10) NOT NULL,
  `email` varchar(50) NOT NULL,
  `method` varchar(50) NOT NULL,
  `address` varchar(500) NOT NULL,
  `total_products` varchar(1000) NOT NULL,
  `total_price` int(100) NOT NULL,
  `placed_on` date NOT NULL DEFAULT current_timestamp(),
  `payment_status` varchar(20) NOT NULL DEFAULT 'pending'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- User table



```
CREATE TABLE `users` (
  `id` int(100) NOT NULL,
  `name` varchar(20) NOT NULL,
  `email` varchar(50) NOT NULL,
  `number` varchar(10) NOT NULL,
  `password` varchar(50) NOT NULL,
  `address` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Products table



```
CREATE TABLE `products` (
  `id` int(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `category` varchar(100) NOT NULL,
  `price` int(10) NOT NULL,
  `image` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

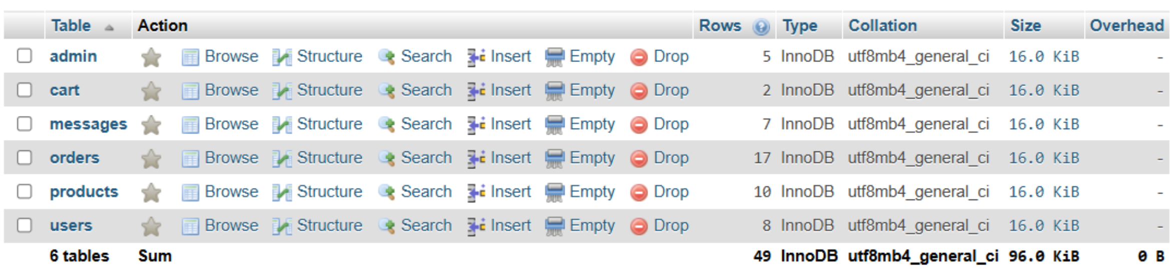
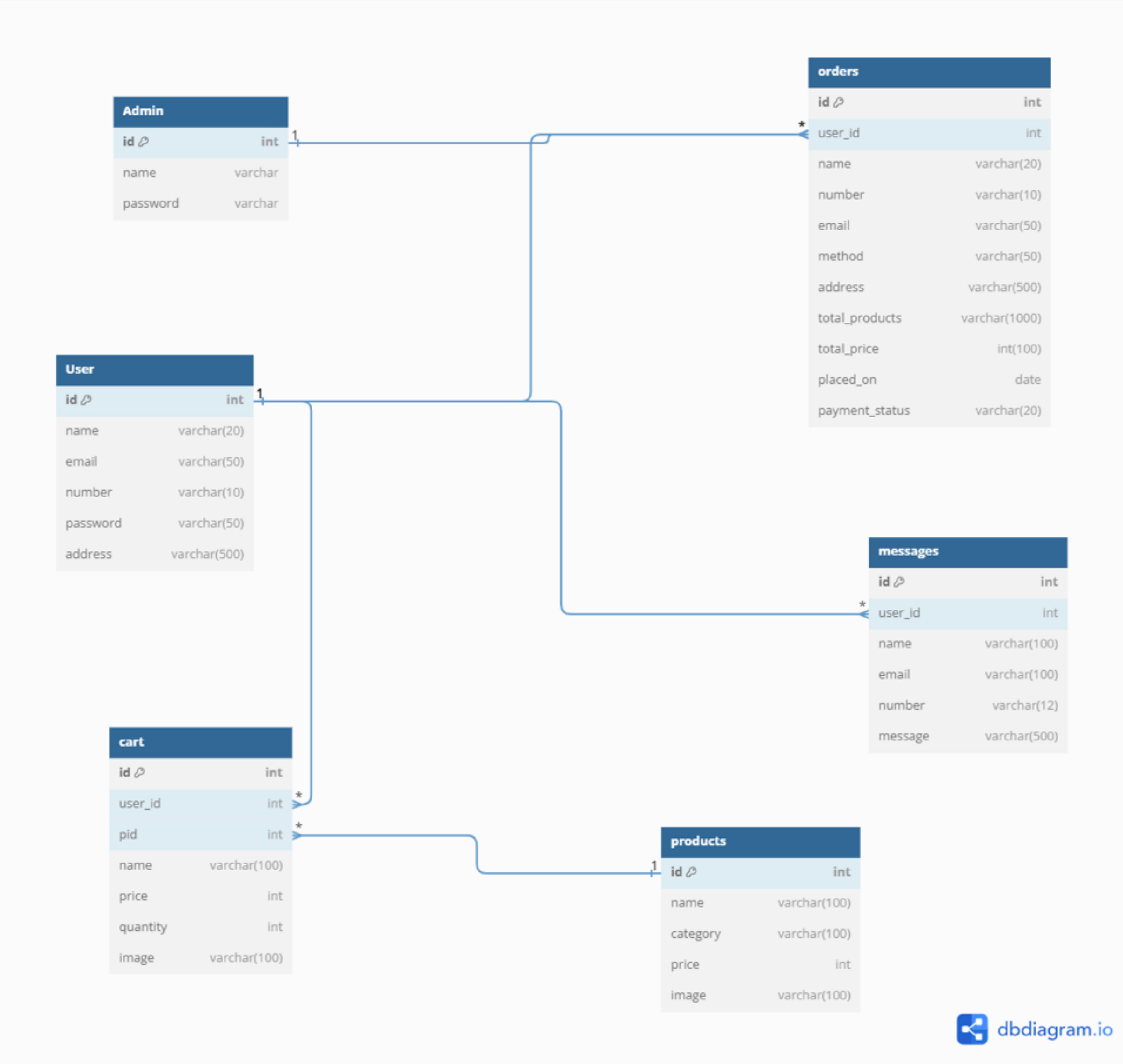


	Table ▲	Action	Rows	Type	Collation	Size	Overhead
□	admin	☆ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
□	cart	☆ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
□	messages	☆ Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
□	orders	☆ Browse Structure Search Insert Empty Drop	17	InnoDB	utf8mb4_general_ci	16.0 KiB	-
□	products	☆ Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
□	users	☆ Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	6 tables	Sum	49	InnoDB	utf8mb4_general_ci	96.0 KiB	0 B

8. Database Diagram



Admin	
id	int
name	varchar
password	varchar

orders	
id	int
user_id	int
name	varchar(20)
number	varchar(10)
email	varchar(50)
method	varchar(50)
address	varchar(500)
total_products	varchar(1000)
total_price	int(100)
placed_on	date
payment_status	varchar(20)

User	
id	int
name	varchar(20)
email	varchar(50)
number	varchar(10)
password	varchar(50)
address	varchar(500)

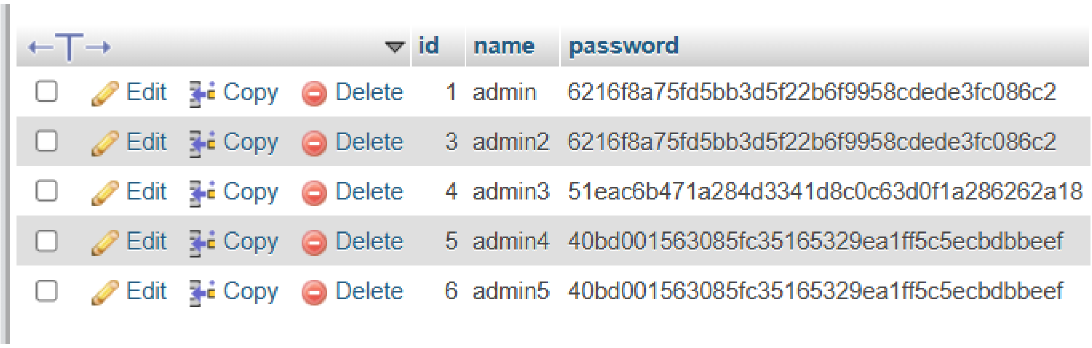
messages	
id	int
user_id	int
name	varchar(100)
email	varchar(100)
number	varchar(12)
message	varchar(500)

cart	
id	int
user_id	int
pid	int
name	varchar(100)
price	int
quantity	int
image	varchar(100)

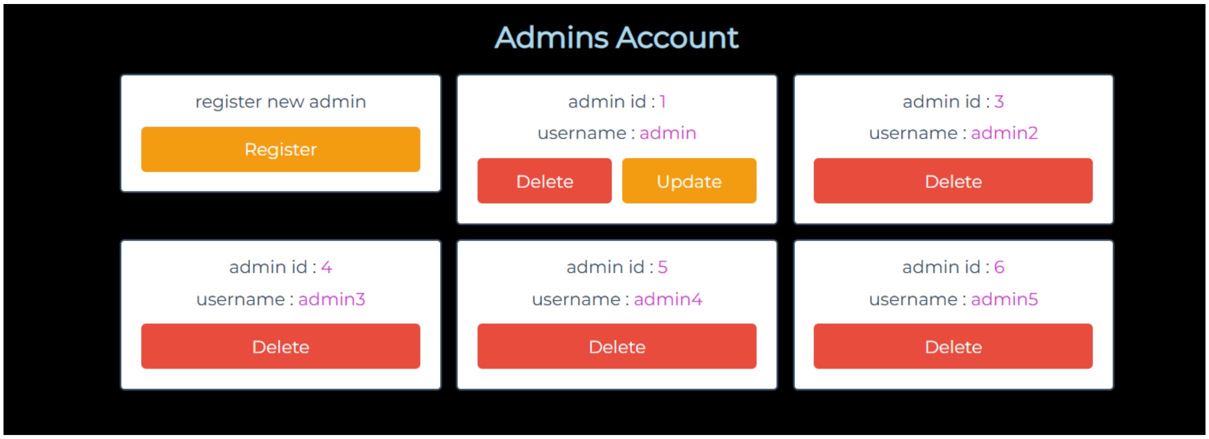
products	
id	int
name	varchar(100)
category	varchar(100)
price	int
image	varchar(100)

9. Tables with records

- Admin table



←T→				id	name	password
☐	Edit	Copy	Delete	1	admin	6216f8a75fd5bb3d5f22b6f9958cdede3fc086c2
☐	Edit	Copy	Delete	3	admin2	6216f8a75fd5bb3d5f22b6f9958cdede3fc086c2
☐	Edit	Copy	Delete	4	admin3	51eac6b471a284d3341d8c0c63d0f1a286262a18
☐	Edit	Copy	Delete	5	admin4	40bd001563085fc35165329ea1ff5c5ecbdbbeef
☐	Edit	Copy	Delete	6	admin5	40bd001563085fc35165329ea1ff5c5ecbdbbeef



Admins Account

register new admin

admin id : 1
username : admin

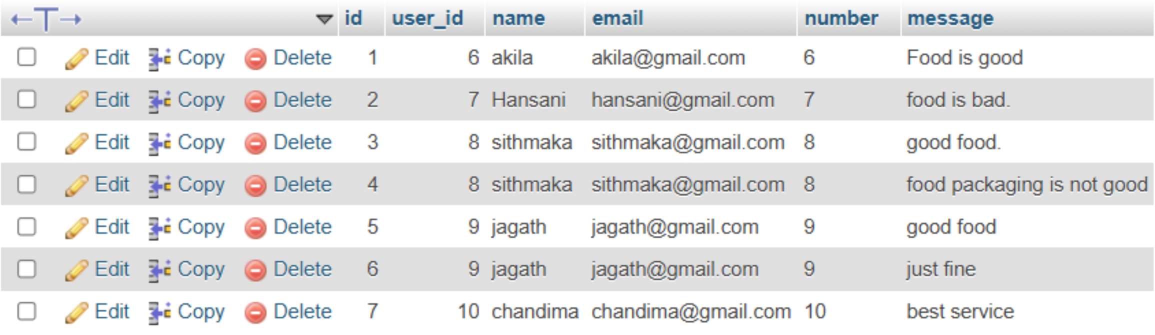
admin id : 3
username : admin2

admin id : 4
username : admin3

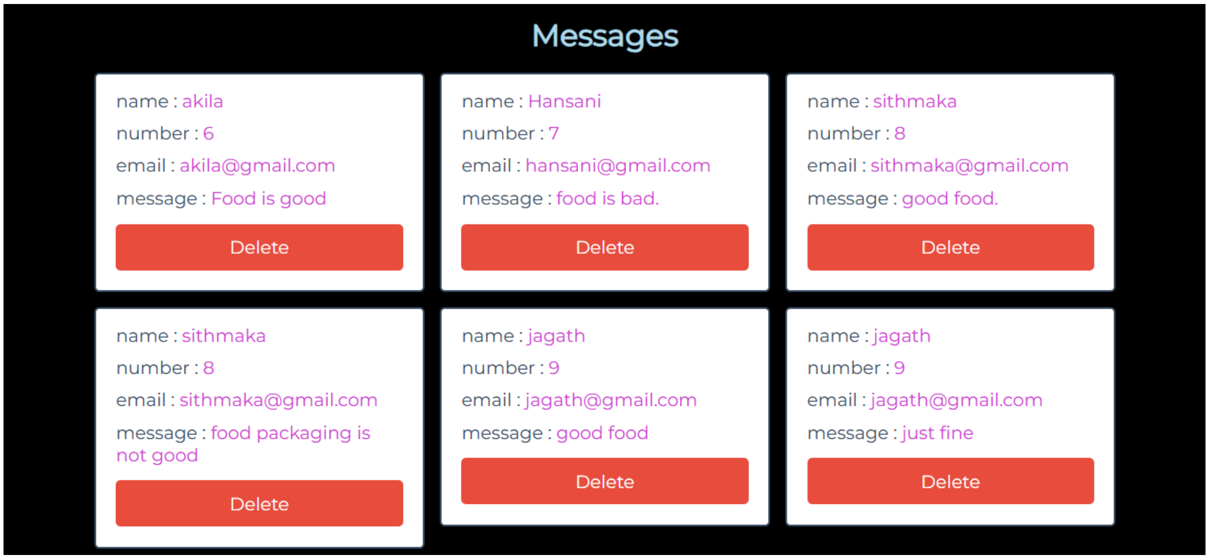
admin id : 5
username : admin4

admin id : 6
username : admin5

- Messages table



←T→				id	user_id	name	email	number	message
□	Edit	Copy	Delete	1	6	akila	akila@gmail.com	6	Food is good
□	Edit	Copy	Delete	2	7	Hansani	hansani@gmail.com	7	food is bad.
□	Edit	Copy	Delete	3	8	sithmaka	sithmaka@gmail.com	8	good food.
□	Edit	Copy	Delete	4	8	sithmaka	sithmaka@gmail.com	8	food packaging is not good
□	Edit	Copy	Delete	5	9	jagath	jagath@gmail.com	9	good food
□	Edit	Copy	Delete	6	9	jagath	jagath@gmail.com	9	just fine
□	Edit	Copy	Delete	7	10	chandima	chandima@gmail.com	10	best service



Messages

name : akila
number : 6
email : akila@gmail.com
message : Food is good

name : Hansani
number : 7
email : hansani@gmail.com
message : food is bad.

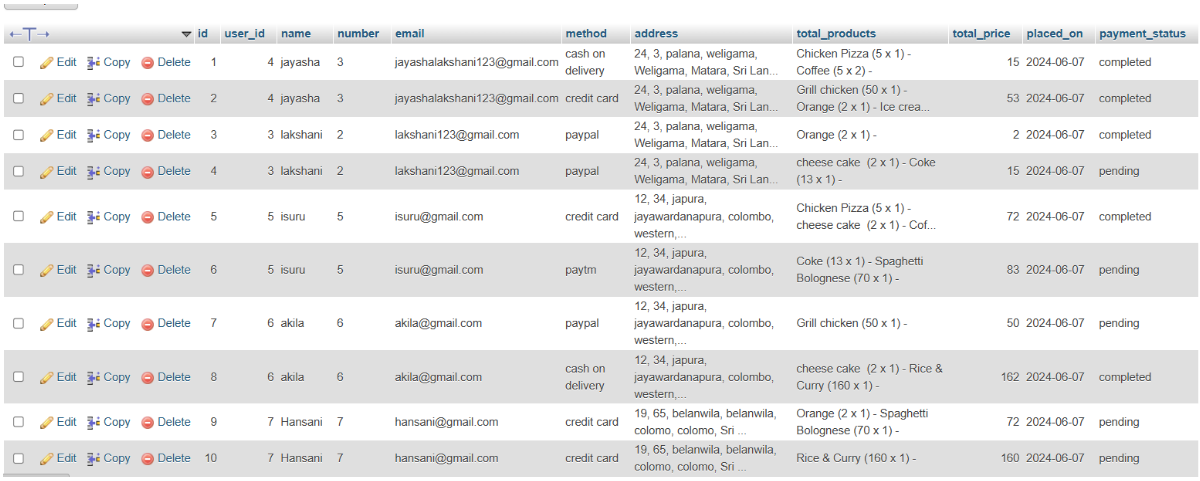
name : sithmaka
number : 8
email : sithmaka@gmail.com
message : good food.

name : sithmaka
number : 8
email : sithmaka@gmail.com
message : food packaging is not good

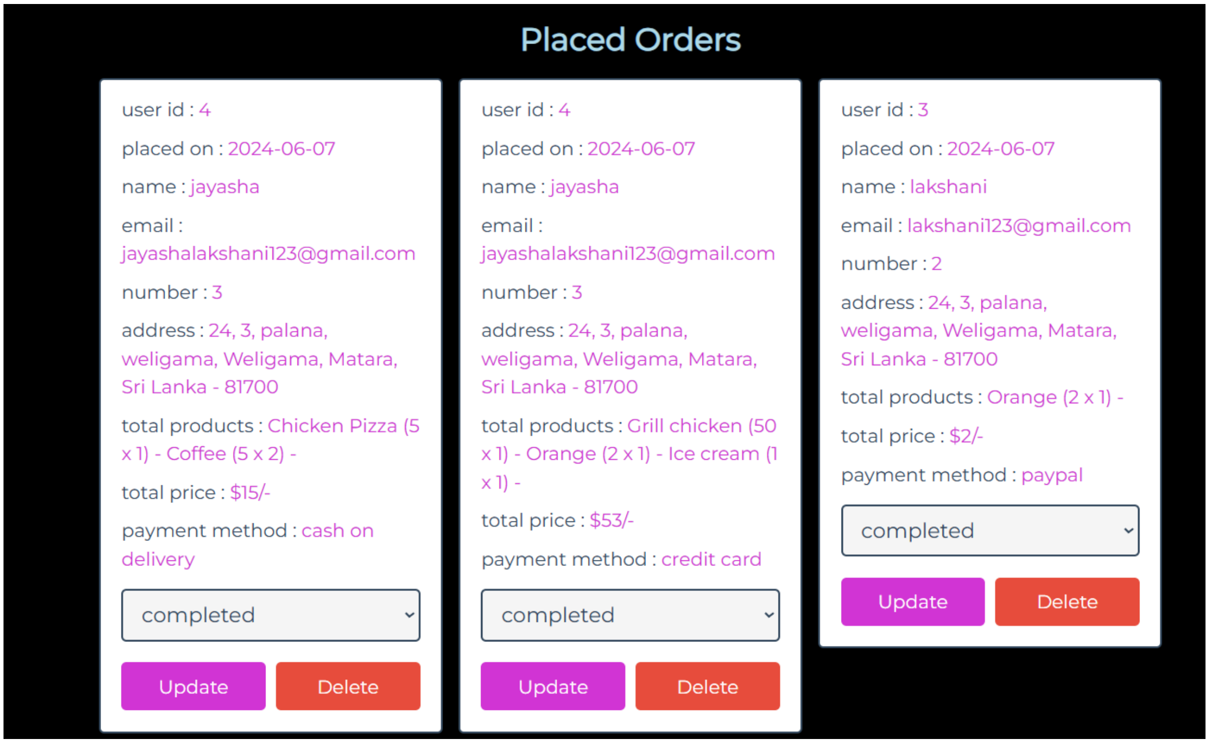
name : jagath
number : 9
email : jagath@gmail.com
message : good food

name : jagath
number : 9
email : jagath@gmail.com
message : just fine

- Order table



←T→ ▼	id	user_id	name	number	email	method	address	total_products	total_price	placed_on	payment_status
□ Edit Copy Delete	1	4	jayasha	3	jayashalakshani123@gmail.com	cash on delivery	24, 3, palana, weligama, Weligama, Matara, Sri Lan...	Chicken Pizza (5 x 1) - Coffee (5 x 2) -	15	2024-06-07	completed
□ Edit Copy Delete	2	4	jayasha	3	jayashalakshani123@gmail.com	credit card	24, 3, palana, weligama, Weligama, Matara, Sri Lan...	Grill chicken (50 x 1) - Orange (2 x 1) - Ice crea...	53	2024-06-07	completed
□ Edit Copy Delete	3	3	lakshani	2	lakshani123@gmail.com	paypal	24, 3, palana, weligama, Weligama, Matara, Sri Lan...	Orange (2 x 1) -	2	2024-06-07	completed
□ Edit Copy Delete	4	3	lakshani	2	lakshani123@gmail.com	paypal	24, 3, palana, weligama, Weligama, Matara, Sri Lan...	cheese cake (2 x 1) - Coke (13 x 1) -	15	2024-06-07	pending
□ Edit Copy Delete	5	5	isuru	5	isuru@gmail.com	credit card	12, 34, japura, jayawardanapura, colombo, western,...	Chicken Pizza (5 x 1) - cheese cake (2 x 1) - Cof...	72	2024-06-07	completed
□ Edit Copy Delete	6	5	isuru	5	isuru@gmail.com	paytm	12, 34, japura, jayawardanapura, colombo, western,...	Coke (13 x 1) - Spaghetti Bolognese (70 x 1) -	83	2024-06-07	pending
□ Edit Copy Delete	7	6	akila	6	akila@gmail.com	paypal	12, 34, japura, jayawardanapura, colombo, western,...	Grill chicken (50 x 1) -	50	2024-06-07	pending
□ Edit Copy Delete	8	6	akila	6	akila@gmail.com	cash on delivery	12, 34, japura, jayawardanapura, colombo, western,...	cheese cake (2 x 1) - Rice & Curry (160 x 1) -	162	2024-06-07	completed
□ Edit Copy Delete	9	7	Hansani	7	hansani@gmail.com	credit card	19, 65, belanwila, belanwila, colomo, colomo, Sri ...	Orange (2 x 1) - Spaghetti Bolognese (70 x 1) -	72	2024-06-07	pending
□ Edit Copy Delete	10	7	Hansani	7	hansani@gmail.com	credit card	19, 65, belanwila, belanwila, colomo, colomo, Sri ...	Rice & Curry (160 x 1) -	160	2024-06-07	pending



Placed Orders

user id : 4

placed on : 2024-06-07

name : jayasha

email : jayashalakshani123@gmail.com

number : 3

address : 24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

total products : Chicken Pizza (5 x 1) - Coffee (5 x 2) -

total price : $15/-

payment method : cash on delivery

user id : 4

placed on : 2024-06-07

name : jayasha

email : jayashalakshani123@gmail.com

number : 3

address : 24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

total products : Grill chicken (50 x 1) - Orange (2 x 1) - Ice cream (1 x 1) -

total price : $53/-

payment method : credit card

user id : 3

placed on : 2024-06-07

name : lakshani

email : lakshani123@gmail.com

number : 2

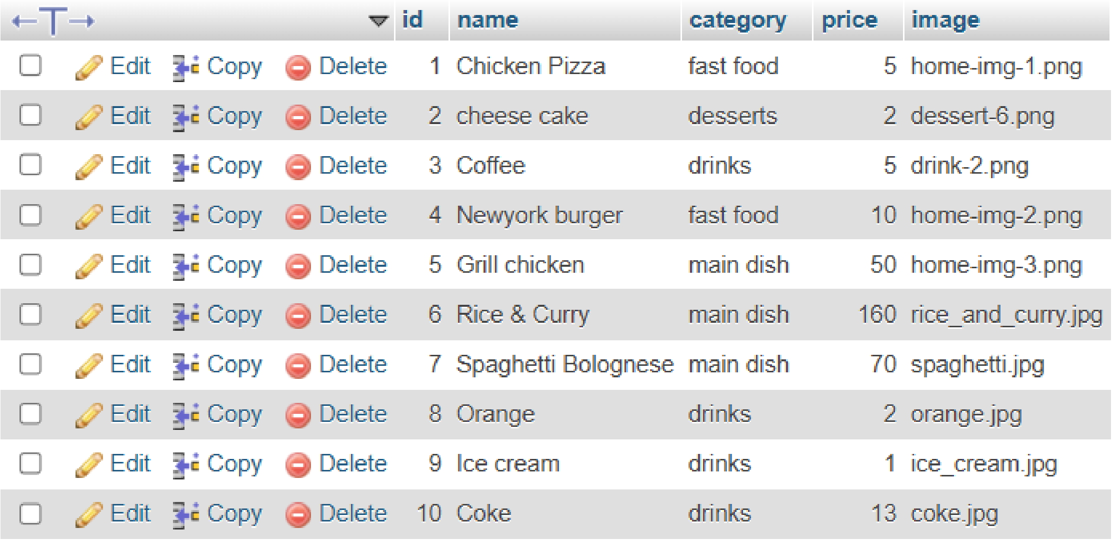
address : 24, 3, palana, weligama, Weligama, Matara, Sri Lanka - 81700

total products : Orange (2 x 1) -

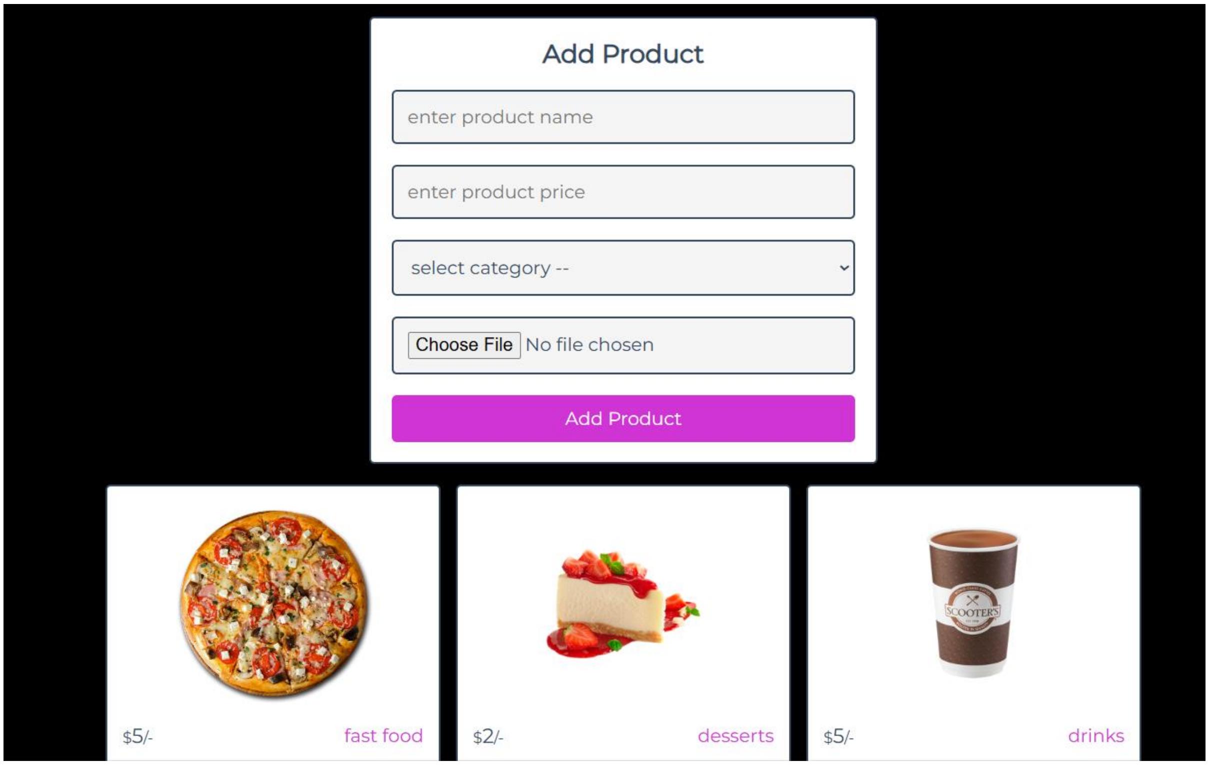
total price : $2/-

payment method : paypal

- Products table



←T→			▼	id	name	category	price	image
□	Edit	Copy	Delete	1	Chicken Pizza	fast food	5	home-img-1.png
□	Edit	Copy	Delete	2	cheese cake	desserts	2	dessert-6.png
□	Edit	Copy	Delete	3	Coffee	drinks	5	drink-2.png
□	Edit	Copy	Delete	4	Newyork burger	fast food	10	home-img-2.png
□	Edit	Copy	Delete	5	Grill chicken	main dish	50	home-img-3.png
□	Edit	Copy	Delete	6	Rice & Curry	main dish	160	rice_and_curry.jpg
□	Edit	Copy	Delete	7	Spaghetti Bolognese	main dish	70	spaghetti.jpg
□	Edit	Copy	Delete	8	Orange	drinks	2	orange.jpg
□	Edit	Copy	Delete	9	Ice cream	drinks	1	ice_cream.jpg
□	Edit	Copy	Delete	10	Coke	drinks	13	coke.jpg



Add Product

enter product name

enter product price

select category --

Choose File No file chosen

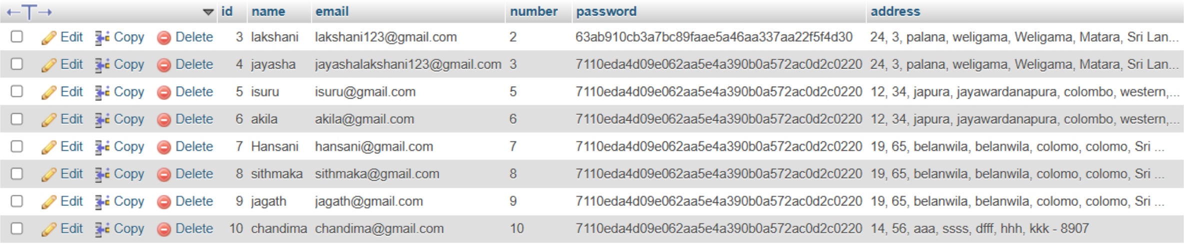
Add Product

$5/- fast food

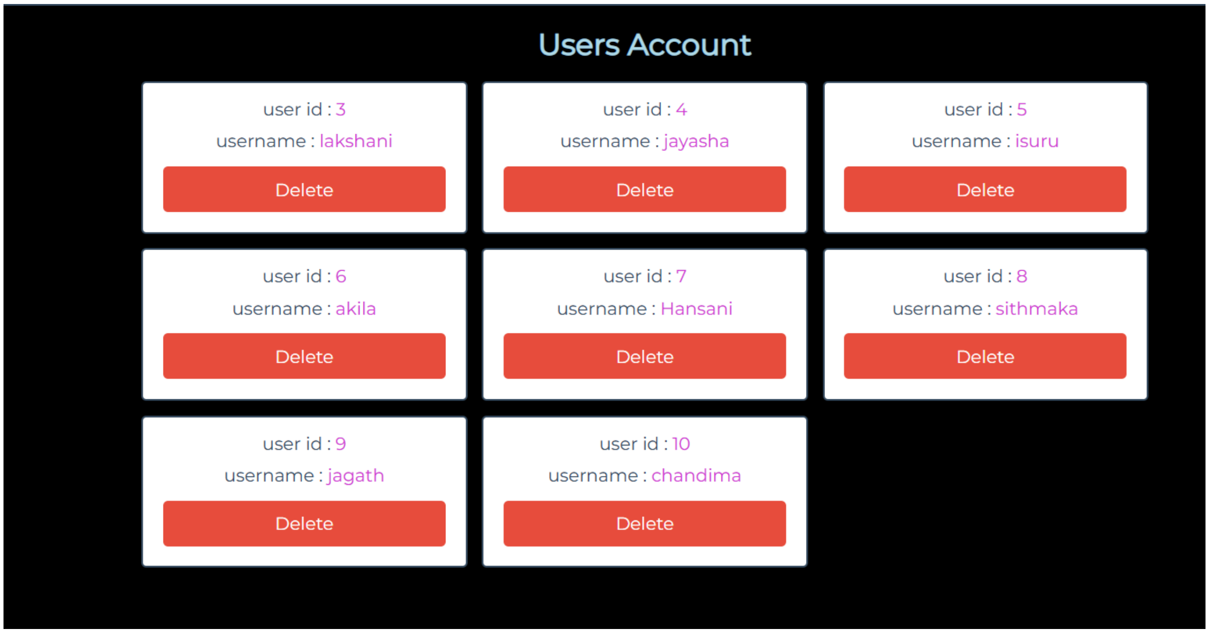
$2/- desserts

$5/- drinks

- Users table



←T→			▼	id	name	email	number	password	address
☐	Edit	Copy	Delete	3	lakshani	lakshani123@gmail.com	2	63ab910cb3a7bc89faae5a46aa337aa22f5f4d30	24, 3, palana, weligama, Weligama, Matara, Sri Lan...
☐	Edit	Copy	Delete	4	jayasha	jayashalakshani123@gmail.com	3	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	24, 3, palana, weligama, Weligama, Matara, Sri Lan...
☐	Edit	Copy	Delete	5	isuru	isuru@gmail.com	5	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	12, 34, japura, jayawardanapura, colombo, western,...
☐	Edit	Copy	Delete	6	akila	akila@gmail.com	6	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	12, 34, japura, jayawardanapura, colombo, western,...
☐	Edit	Copy	Delete	7	Hansani	hansani@gmail.com	7	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	19, 65, belanwila, belanwila, colomo, colomo, Sri ...
☐	Edit	Copy	Delete	8	sithmaka	sithmaka@gmail.com	8	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	19, 65, belanwila, belanwila, colomo, colomo, Sri ...
☐	Edit	Copy	Delete	9	jagath	jagath@gmail.com	9	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	19, 65, belanwila, belanwila, colomo, colomo, Sri ...
☐	Edit	Copy	Delete	10	chandima	chandima@gmail.com	10	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	14, 56, aaa, ssss, dfff, hhh, kkk - 8907



Users Account

user id : 3
username : lakshani

user id : 4
username : jayasha

user id : 5
username : isuru

user id : 6
username : akila

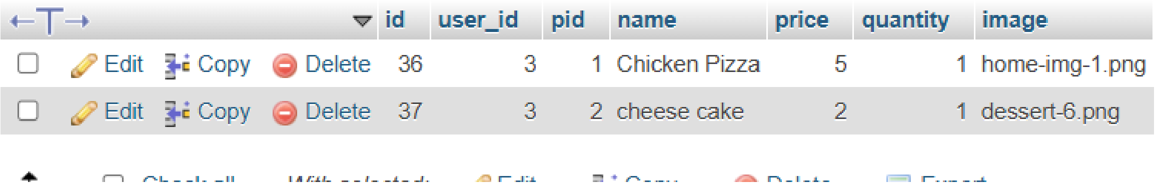
user id : 7
username : Hansani

user id : 8
username : sithmaka

user id : 9
username : jagath

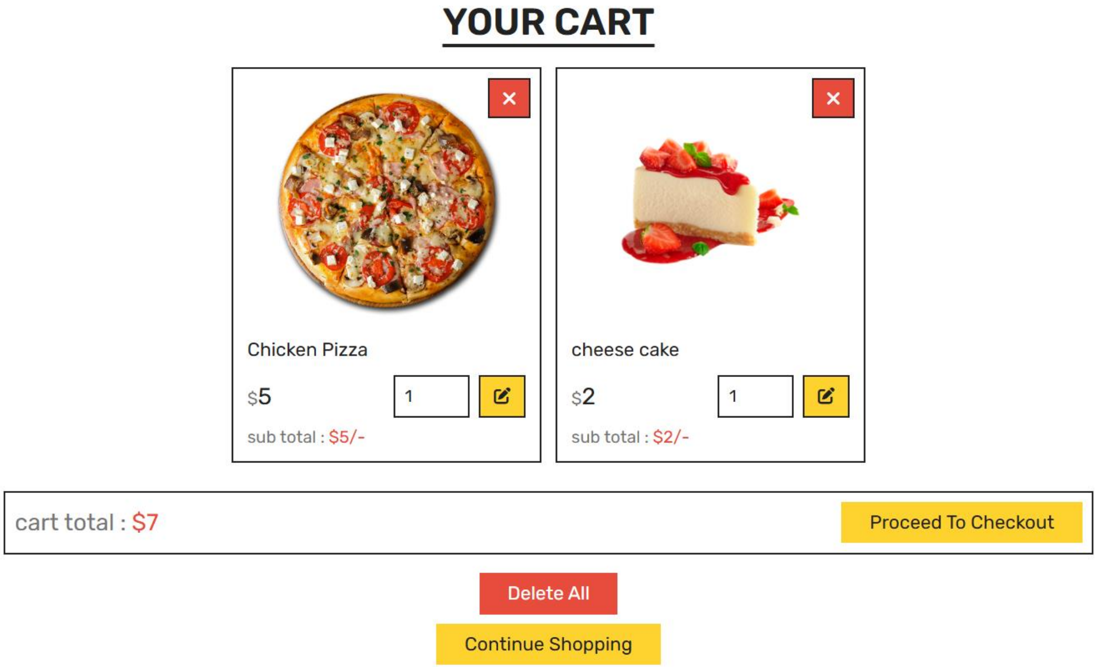
user id : 10
username : chandima

- Cart table



←T→			id	user_id	pid	name	price	quantity	image
□ Edit	Copy	Delete	36	3	1	Chicken Pizza	5	1	home-img-1.png
□ Edit	Copy	Delete	37	3	2	cheese cake	2	1	dessert-6.png

□ Check all With selected: Edit Copy Delete Export



YOUR CART

Chicken Pizza

$5 1

sub total : $5/-

cheese cake

$2 1

sub total : $2/-

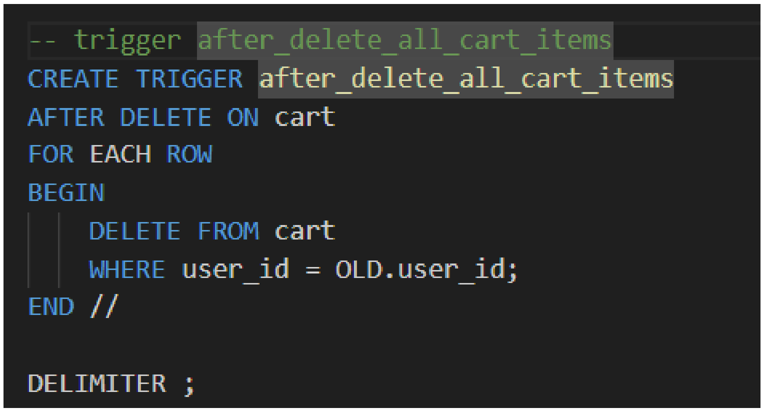
cart total : $7

Section 03

10. Triggers

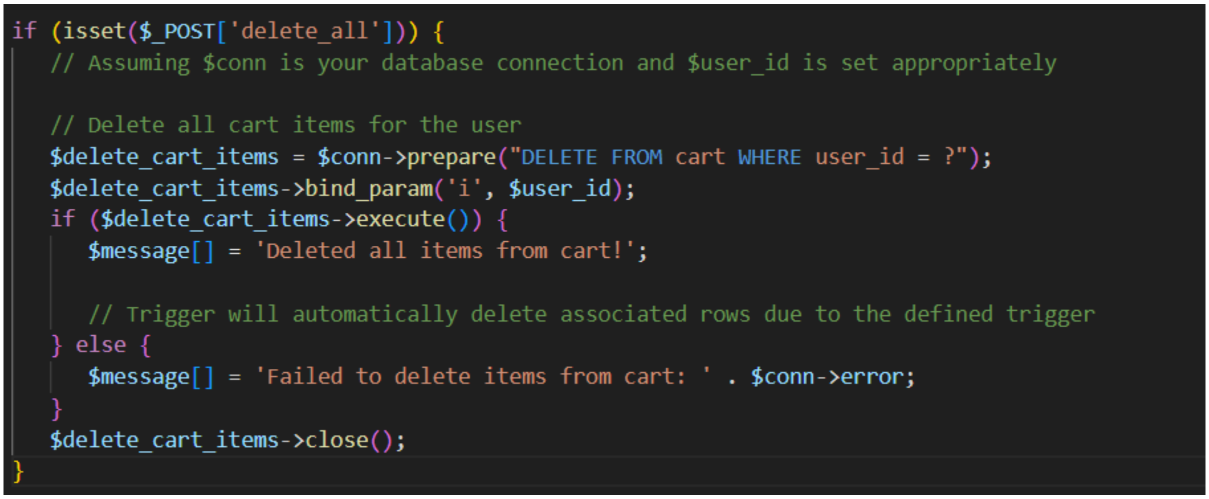
10.1. Delete all the items in the order (cart table)

1. Create Trigger



```
-- trigger after_delete_all_cart_items
CREATE TRIGGER after_delete_all_cart_items
AFTER DELETE ON cart
FOR EACH ROW
BEGIN
    DELETE FROM cart
    WHERE user_id = OLD.user_id;
END //

DELIMITER ;
```

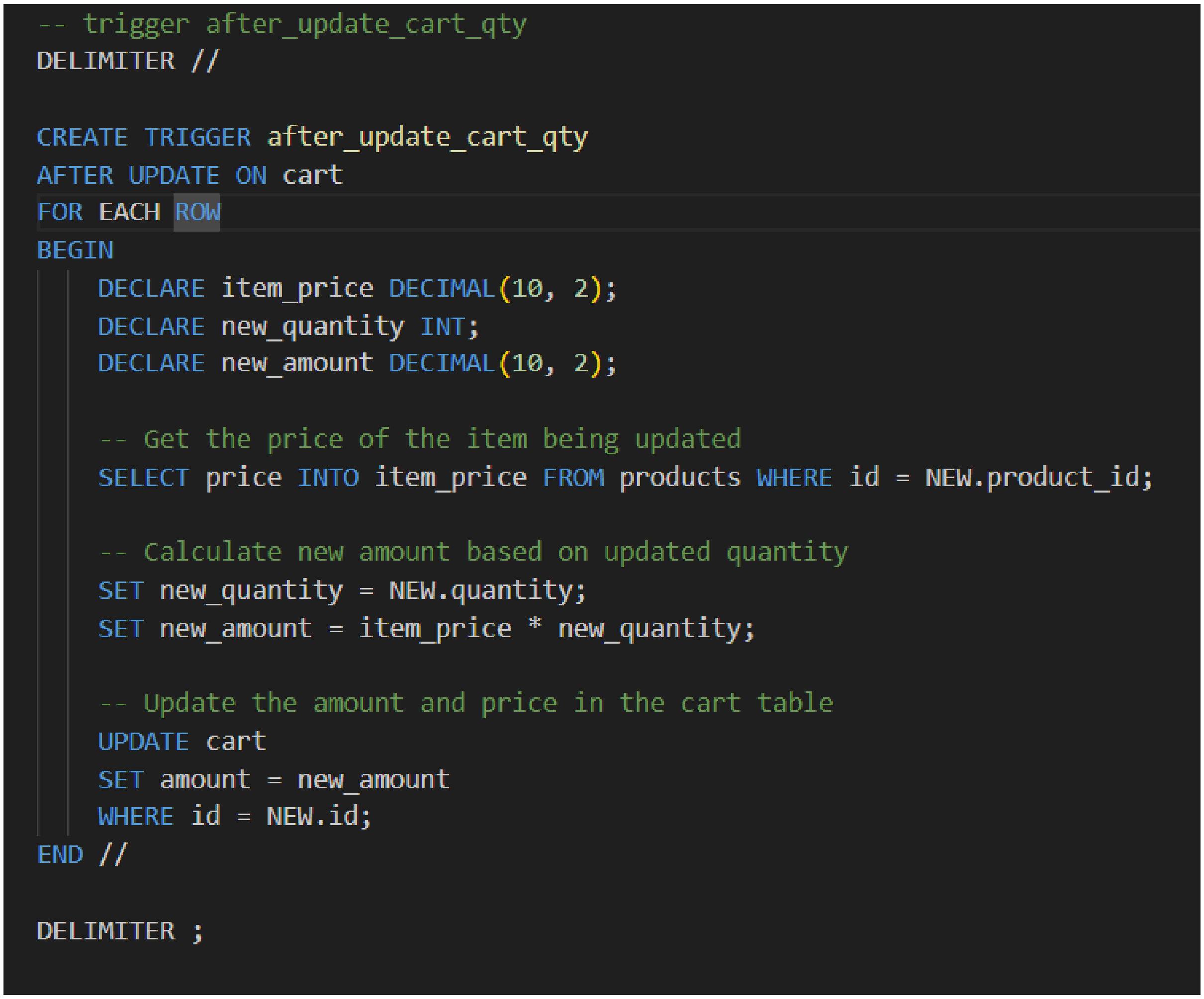
2. PHP code



```
if (isset($_POST['delete_all'])) {
    // Assuming $conn is your database connection and $user_id is set appropriately

    // Delete all cart items for the user
    $delete_cart_items = $conn->prepare("DELETE FROM cart WHERE user_id = ?");
    $delete_cart_items->bind_param('i', $user_id);
    if ($delete_cart_items->execute()) {
        $message[] = 'Deleted all items from cart!';

        // Trigger will automatically delete associated rows due to the defined trigger
    } else {
        $message[] = 'Failed to delete items from cart: ' . $conn->error;
    }
    $delete_cart_items->close();
}
```

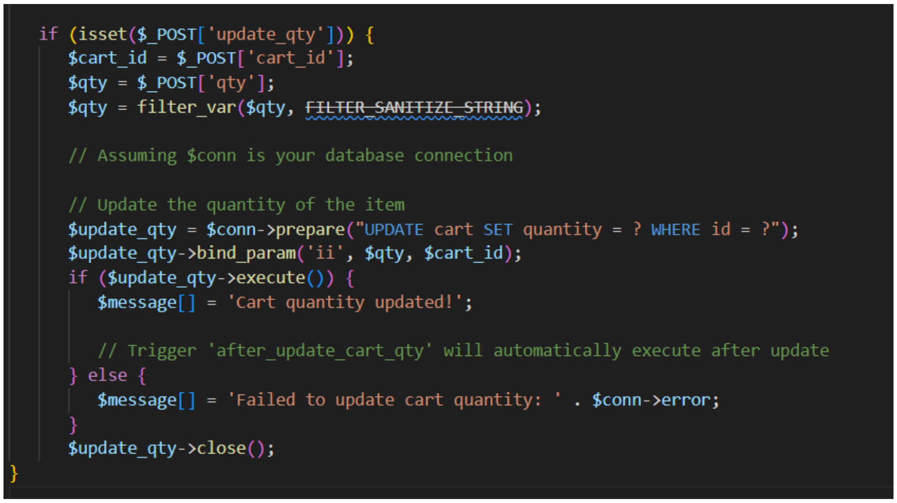
10.2. Update the item quantity (cart table)

1. Create Trigger



```sql
-- trigger after_update_cart_qty
DELIMITER //

CREATE TRIGGER after_update_cart_qty
AFTER UPDATE ON cart
FOR EACH ROW
BEGIN
    DECLARE item_price DECIMAL(10, 2);
    DECLARE new_quantity INT;
    DECLARE new_amount DECIMAL(10, 2);

    -- Get the price of the item being updated
    SELECT price INTO item_price FROM products WHERE id = NEW.product_id;

    -- Calculate new amount based on updated quantity
    SET new_quantity = NEW.quantity;
    SET new_amount = item_price * new_quantity;

    -- Update the amount and price in the cart table
    UPDATE cart
    SET amount = new_amount
    WHERE id = NEW.id;
END //

DELIMITER ;
```

2. PHP code

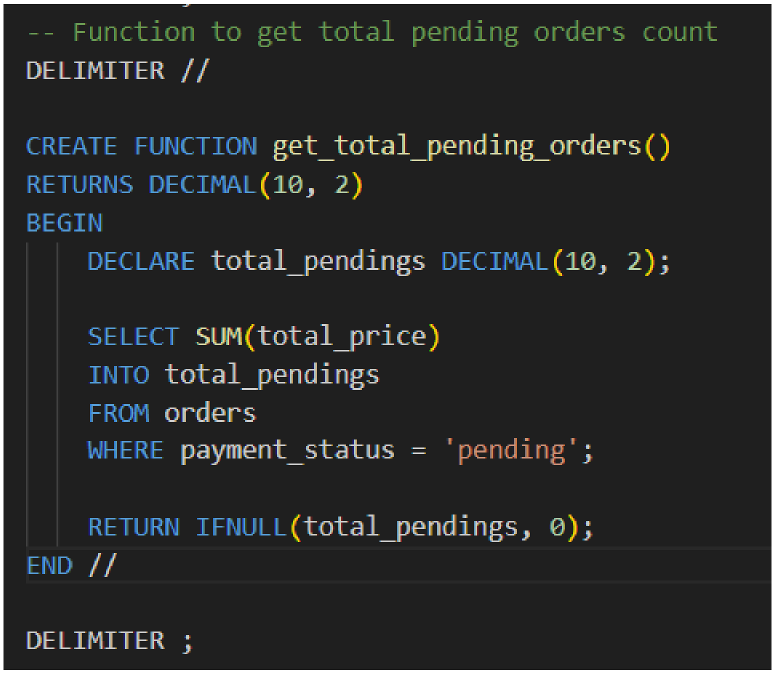


```php
if (isset($_POST['update_qty'])) {
    $cart_id = $_POST['cart_id'];
    $qty = $_POST['qty'];
    $qty = filter_var($qty, ~~FILTER_SANITIZE_STRING~~);

    // Assuming $conn is your database connection

    // Update the quantity of the item
    $update_qty = $conn->prepare("UPDATE cart SET quantity = ? WHERE id = ?");
    $update_qty->bind_param('ii', $qty, $cart_id);
    if ($update_qty->execute()) {
        $message[] = 'Cart quantity updated!';

        // Trigger 'after_update_cart_qty' will automatically execute after update
    } else {
        $message[] = 'Failed to update cart quantity: ' . $conn->error;
    }
    $update_qty->close();
}
```

11. Functions.

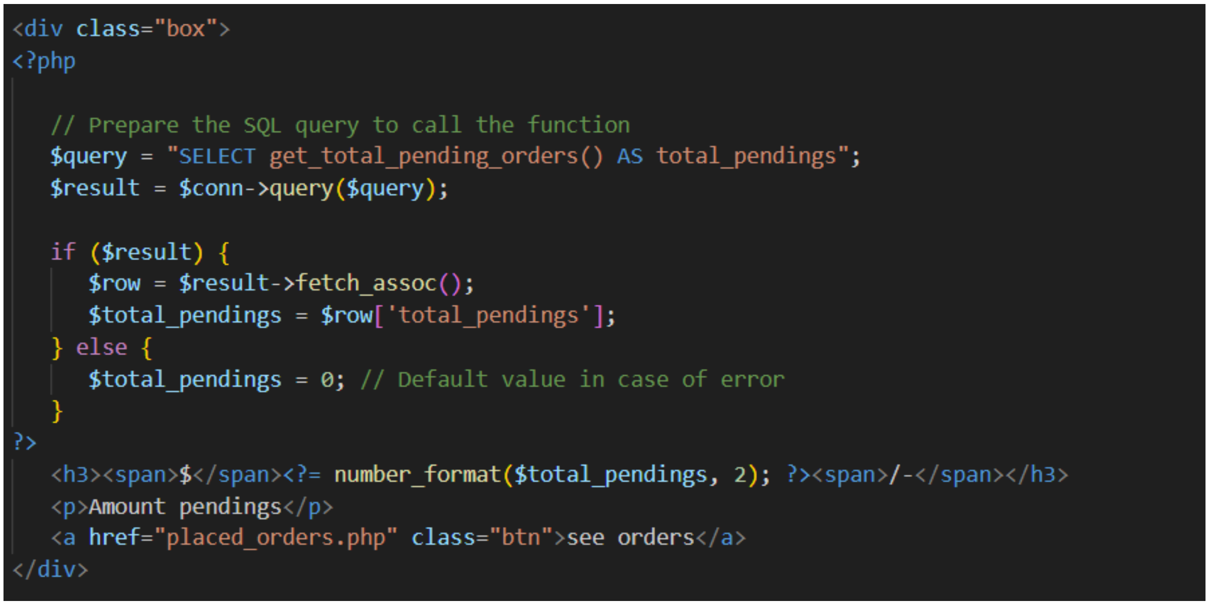
11.1. Create a function to get total pending amount

1. Create Function



```sql
-- Function to get total pending orders count
DELIMITER //

CREATE FUNCTION get_total_pending_orders()
RETURNS DECIMAL(10, 2)
BEGIN
    DECLARE total_pendings DECIMAL(10, 2);

    SELECT SUM(total_price)
    INTO total_pendings
    FROM orders
    WHERE payment_status = 'pending';

    RETURN IFNULL(total_pendings, 0);
END //

DELIMITER ;
```

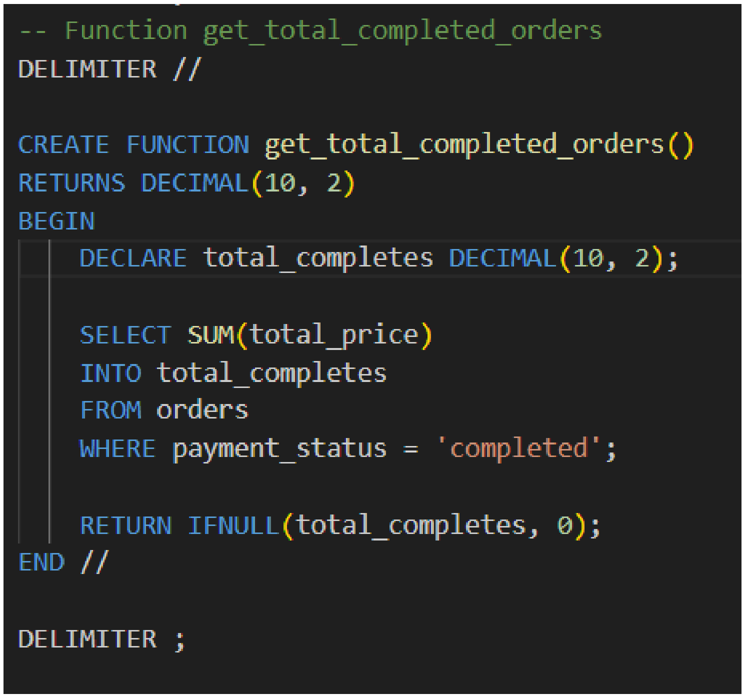
2. PHP Code



```php
<div class="box">
<?php

   // Prepare the SQL query to call the function
   $query = "SELECT get_total_pending_orders() AS total_pendings";
   $result = $conn->query($query);

   if ($result) {
      $row = $result->fetch_assoc();
      $total_pendings = $row['total_pendings'];
   } else {
      $total_pendings = 0; // Default value in case of error
   }
?>
   <h3><span>$</span><?= number_format($total_pendings, 2); ?><span>/-</span></h3>
   <p>Amount pendings</p>
   <a href="placed_orders.php" class="btn">see orders</a>
</div>
```

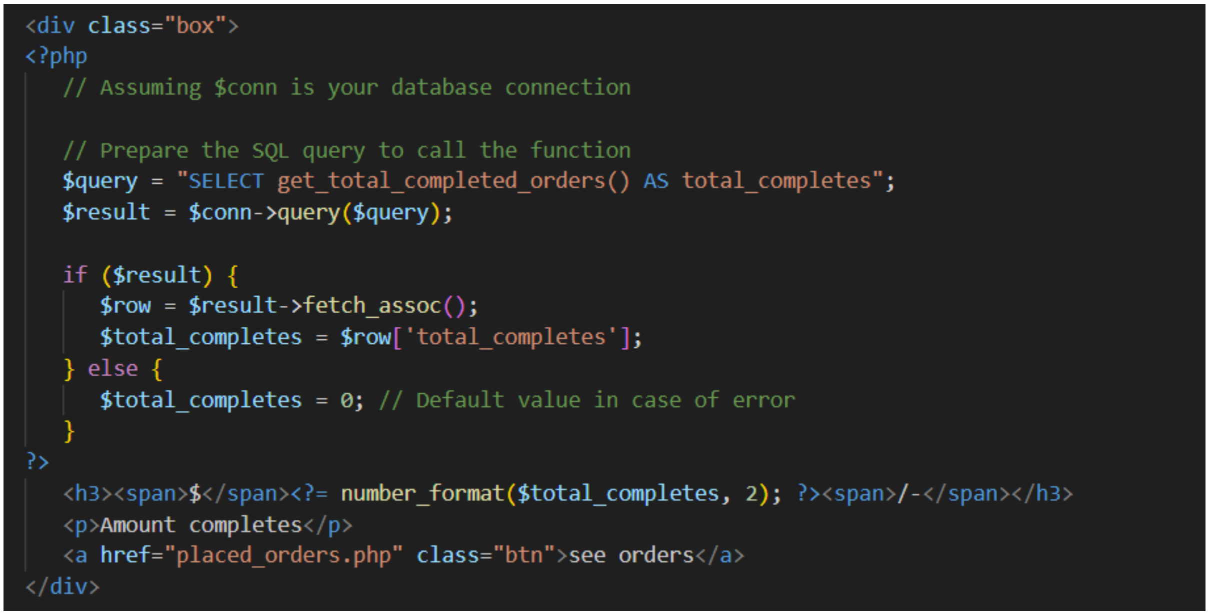
11.2. Create a function to get total complete amount

1. Create Function



```
-- Function get_total_completed_orders
DELIMITER //

CREATE FUNCTION get_total_completed_orders()
RETURNS DECIMAL(10, 2)
BEGIN
    DECLARE total_completes DECIMAL(10, 2);

    SELECT SUM(total_price)
    INTO total_completes
    FROM orders
    WHERE payment_status = 'completed';

    RETURN IFNULL(total_completes, 0);
END //

DELIMITER ;
```

2. PHP Code

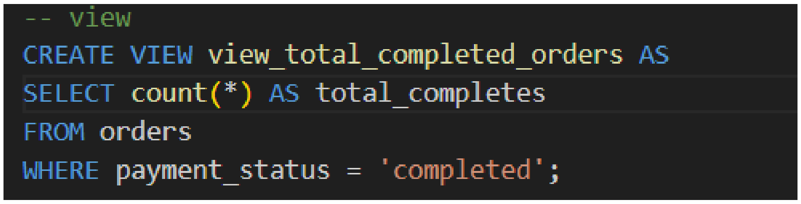


```
<div class="box">
<?php
    // Assuming $conn is your database connection

    // Prepare the SQL query to call the function
    $query = "SELECT get_total_completed_orders() AS total_completes";
    $result = $conn->query($query);

    if ($result) {
        $row = $result->fetch_assoc();
        $total_completes = $row['total_completes'];
    } else {
        $total_completes = 0; // Default value in case of error
    }
?>
    <h3><span>$</span><?= number_format($total_completes, 2); ?><span>/-</span></h3>
    <p>Amount completes</p>
    <a href="placed_orders.php" class="btn">see orders</a>
</div>
```

12. Views.

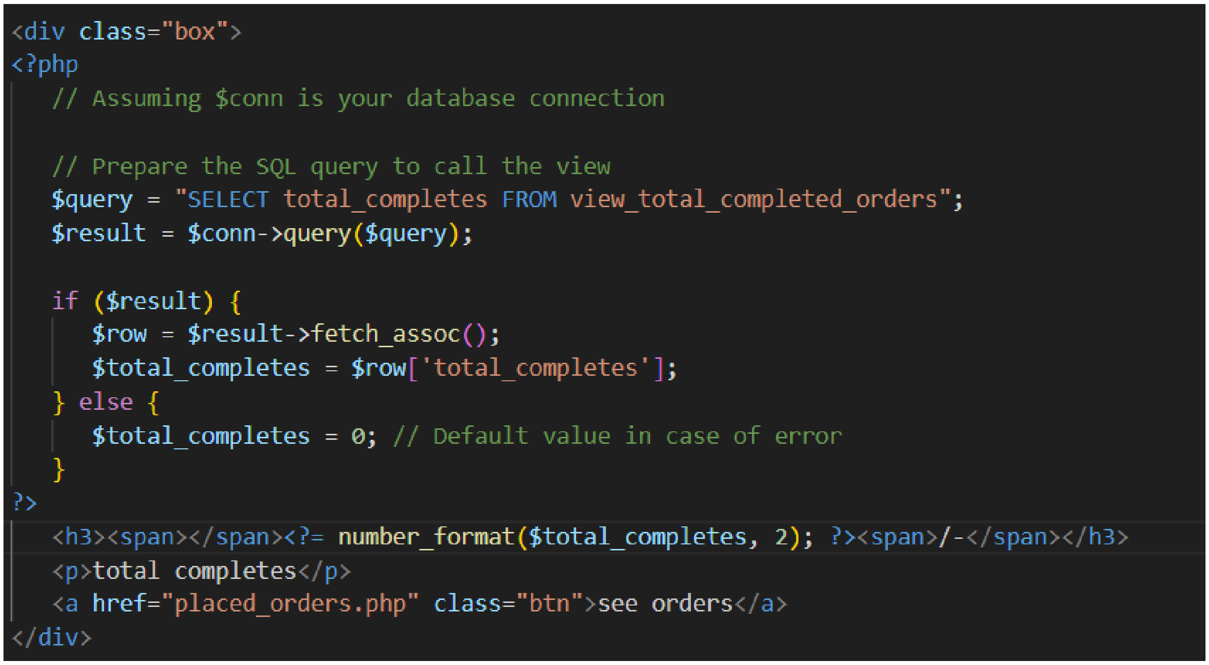
12.1. Number of orders completes

1. Create view



```
-- view
CREATE VIEW view_total_completed_orders AS
SELECT count(*) AS total_completes
FROM orders
WHERE payment_status = 'completed';
```

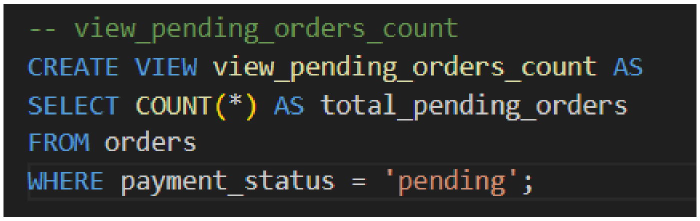
2. PHP Code



```
<div class="box">
<?php
   // Assuming $conn is your database connection

   // Prepare the SQL query to call the view
   $query = "SELECT total_completes FROM view_total_completed_orders";
   $result = $conn->query($query);

   if ($result) {
      $row = $result->fetch_assoc();
      $total_completes = $row['total_completes'];
   } else {
      $total_completes = 0; // Default value in case of error
   }
?>
   <h3><span></span><?= number_format($total_completes, 2); ?><span>/-</span></h3>
   <p>total completes</p>
   <a href="placed_orders.php" class="btn">see orders</a>
</div>
```

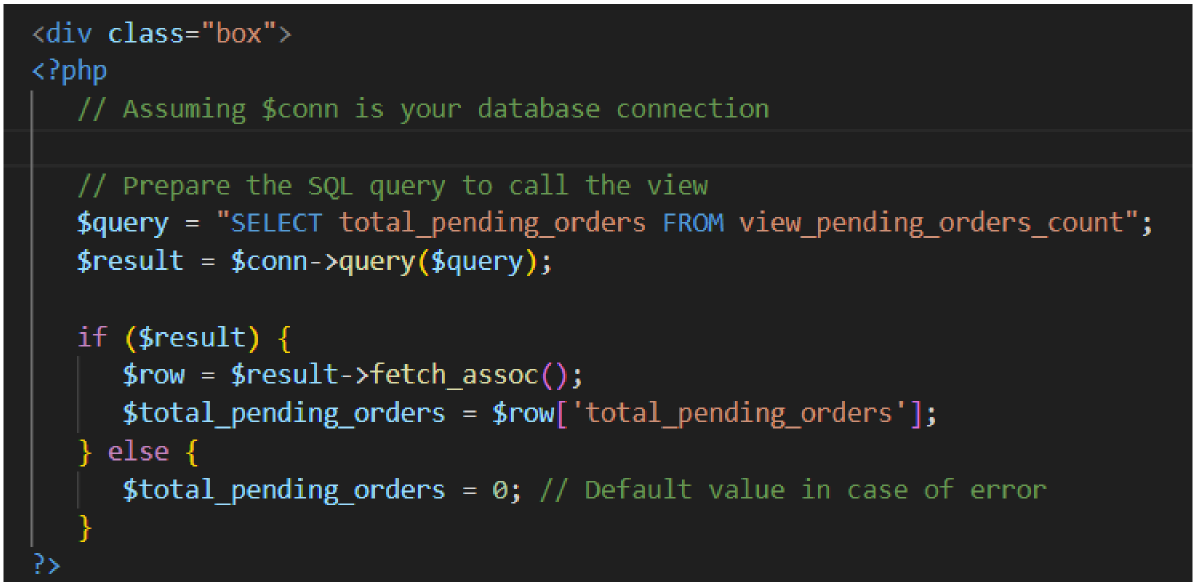
12.2. Number of orders pending

1. Create View



```
-- view_pending_orders_count
CREATE VIEW view_pending_orders_count AS
SELECT COUNT(*) AS total_pending_orders
FROM orders
WHERE payment_status = 'pending';
```

2. PHP Code

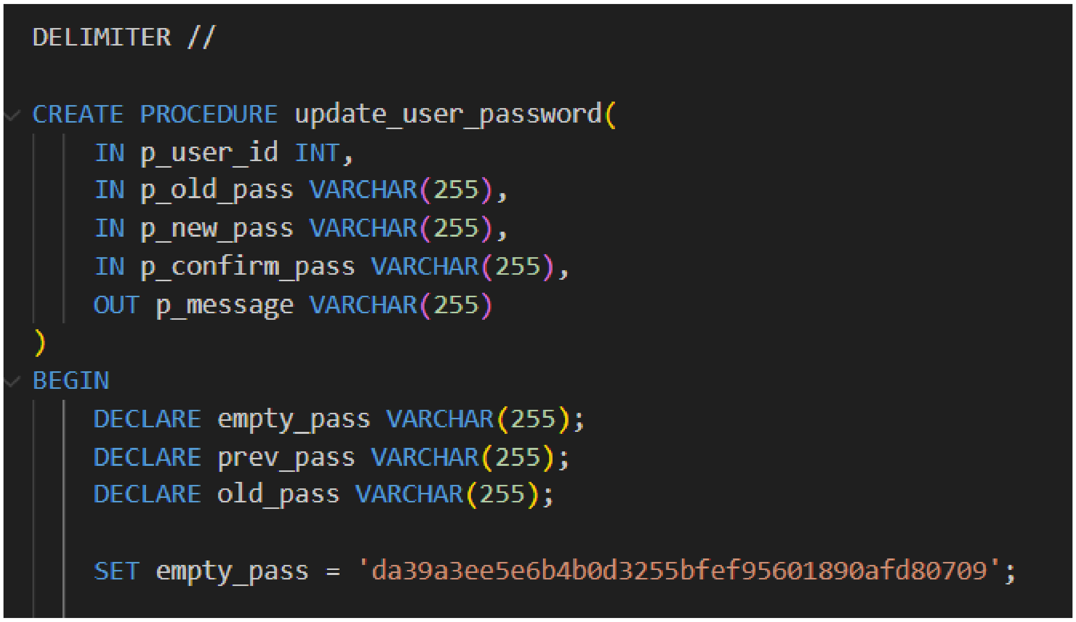


```
<div class="box">
<?php
   // Assuming $conn is your database connection

   // Prepare the SQL query to call the view
   $query = "SELECT total_pending_orders FROM view_pending_orders_count";
   $result = $conn->query($query);

   if ($result) {
      $row = $result->fetch_assoc();
      $total_pending_orders = $row['total_pending_orders'];
   } else {
      $total_pending_orders = 0; // Default value in case of error
   }
?>
```

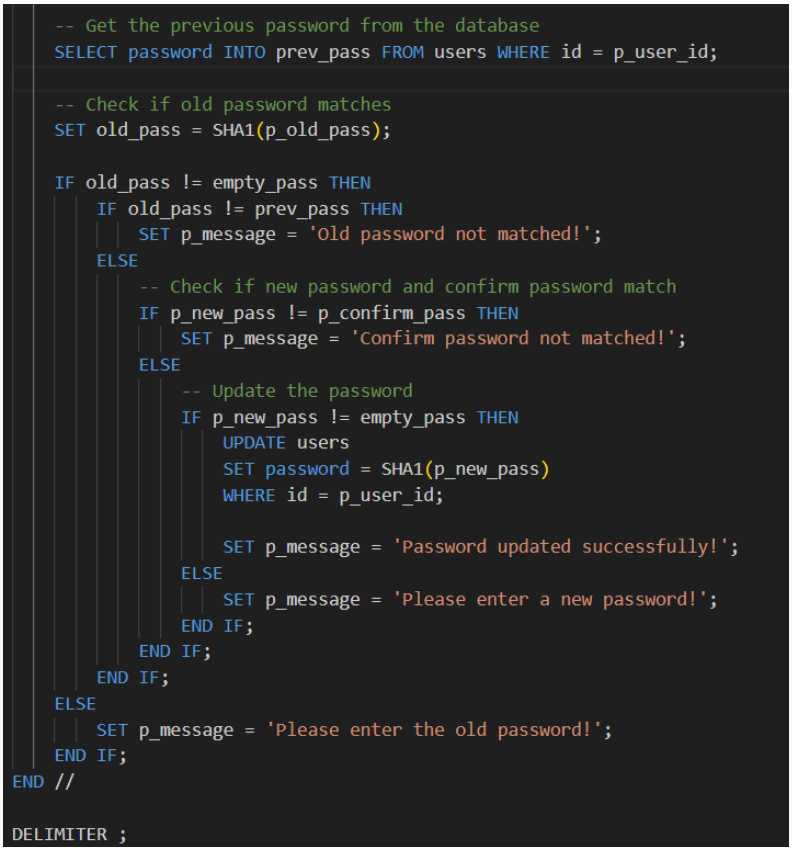
13. Procedures.

13.1. Update password.

1. Create a Stored Procedure

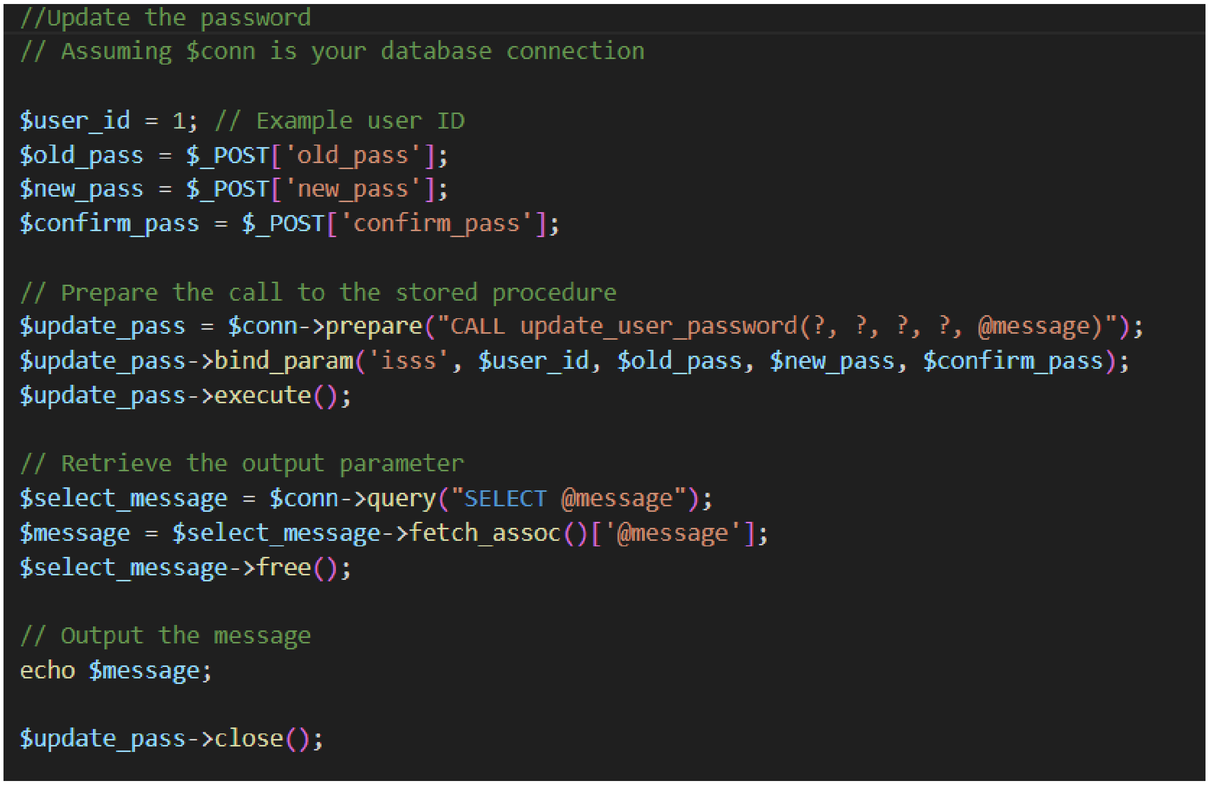


```sql
DELIMITER //

CREATE PROCEDURE update_user_password(
    IN p_user_id INT,
    IN p_old_pass VARCHAR(255),
    IN p_new_pass VARCHAR(255),
    IN p_confirm_pass VARCHAR(255),
    OUT p_message VARCHAR(255)
)
BEGIN
    DECLARE empty_pass VARCHAR(255);
    DECLARE prev_pass VARCHAR(255);
    DECLARE old_pass VARCHAR(255);

    SET empty_pass = 'da39a3ee5e6b4b0d3255bfef95601890afd80709';
```



```sql
    -- Get the previous password from the database
    SELECT password INTO prev_pass FROM users WHERE id = p_user_id;

    -- Check if old password matches
    SET old_pass = SHA1(p_old_pass);

    IF old_pass != empty_pass THEN
        IF old_pass != prev_pass THEN
            SET p_message = 'Old password not matched!';
        ELSE
            -- Check if new password and confirm password match
            IF p_new_pass != p_confirm_pass THEN
                SET p_message = 'Confirm password not matched!';
            ELSE
                -- Update the password
                IF p_new_pass != empty_pass THEN
                    UPDATE users
                    SET password = SHA1(p_new_pass)
                    WHERE id = p_user_id;

                    SET p_message = 'Password updated successfully!';
                ELSE
                    SET p_message = 'Please enter a new password!';
                END IF;
            END IF;
        END IF;
    ELSE
        SET p_message = 'Please enter the old password!';
    END IF;
END //

DELIMITER ;
```

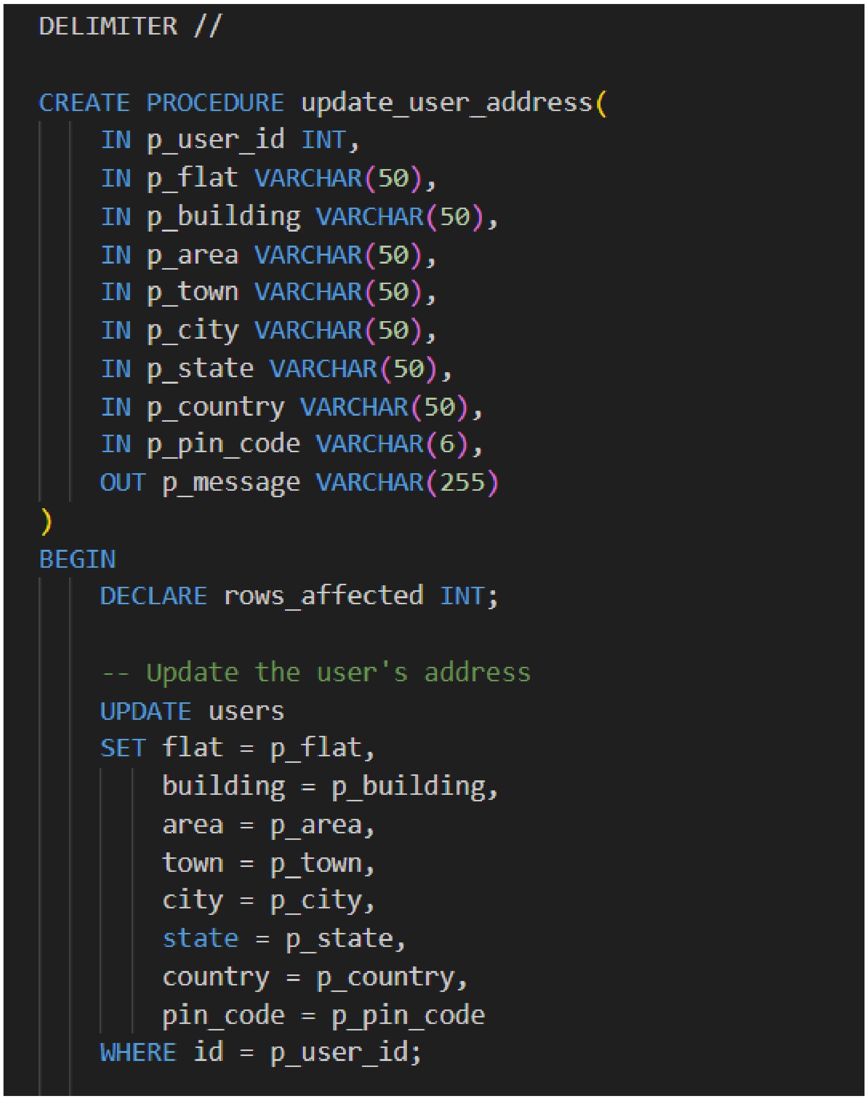
2. PHP Code to Call the Stored Procedure



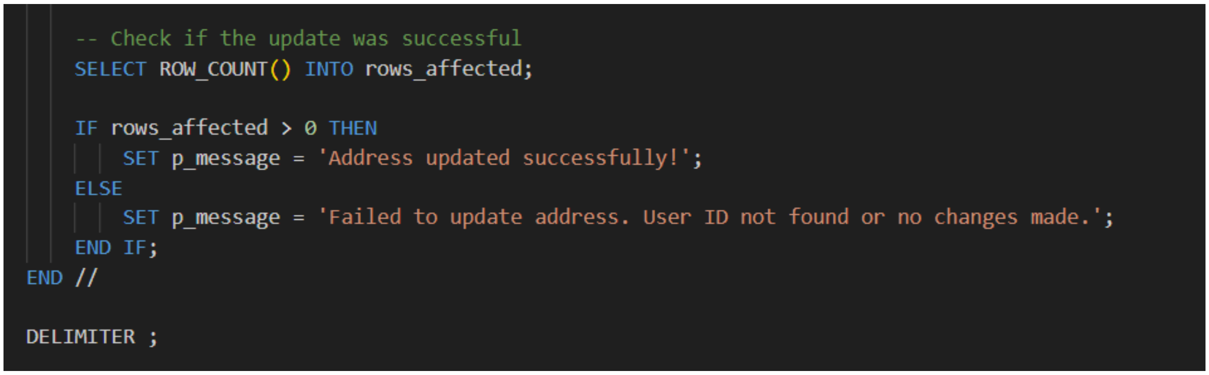
```php
//Update the password
// Assuming $conn is your database connection

$user_id = 1; // Example user ID
$old_pass = $_POST['old_pass'];
$new_pass = $_POST['new_pass'];
$confirm_pass = $_POST['confirm_pass'];

// Prepare the call to the stored procedure
$update_pass = $conn->prepare("CALL update_user_password(?, ?, ?, ?, @message)");
$update_pass->bind_param('isss', $user_id, $old_pass, $new_pass, $confirm_pass);
$update_pass->execute();

// Retrieve the output parameter
$select_message = $conn->query("SELECT @message");
$message = $select_message->fetch_assoc()['@message'];
$select_message->free();

// Output the message
echo $message;

$update_pass->close();
```

13.2. Update address.

1. Create a Stored Procedure

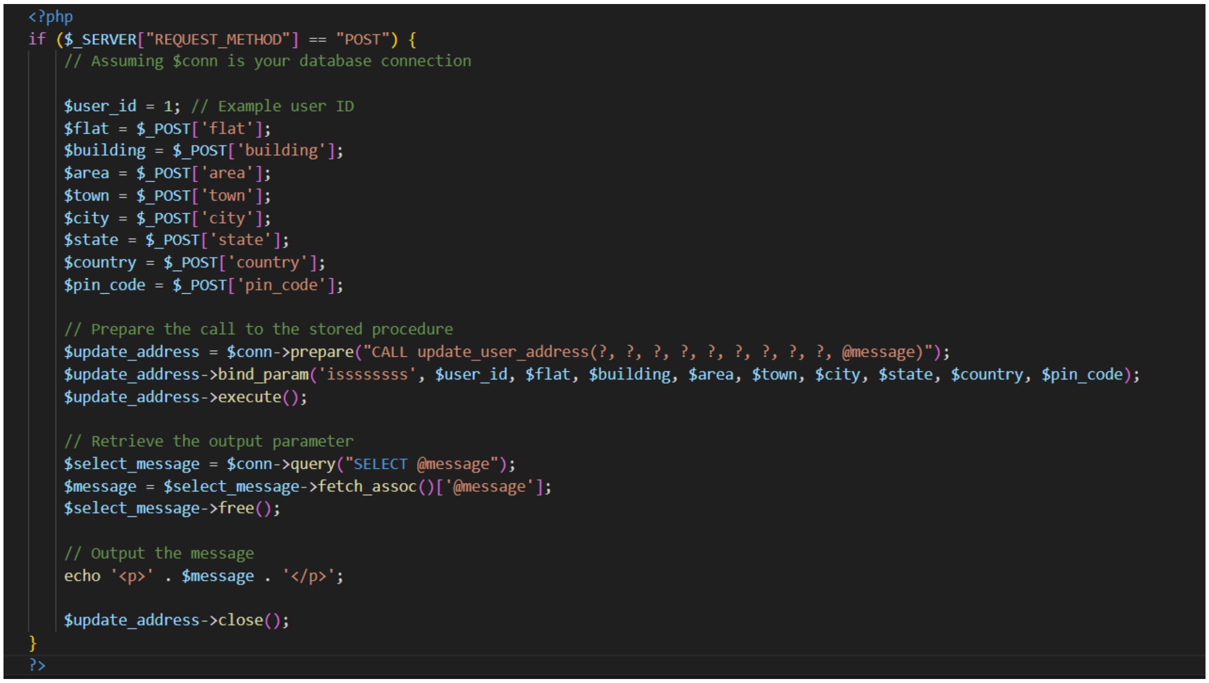


```
DELIMITER //

CREATE PROCEDURE update_user_address(
    IN p_user_id INT,
    IN p_flat VARCHAR(50),
    IN p_building VARCHAR(50),
    IN p_area VARCHAR(50),
    IN p_town VARCHAR(50),
    IN p_city VARCHAR(50),
    IN p_state VARCHAR(50),
    IN p_country VARCHAR(50),
    IN p_pin_code VARCHAR(6),
    OUT p_message VARCHAR(255)
)
BEGIN
    DECLARE rows_affected INT;

    -- Update the user's address
    UPDATE users
    SET flat = p_flat,
        building = p_building,
        area = p_area,
        town = p_town,
        city = p_city,
        state = p_state,
        country = p_country,
        pin_code = p_pin_code
    WHERE id = p_user_id;
```



```
    -- Check if the update was successful
    SELECT ROW_COUNT() INTO rows_affected;

    IF rows_affected > 0 THEN
        SET p_message = 'Address updated successfully!';
    ELSE
        SET p_message = 'Failed to update address. User ID not found or no changes made.';
    END IF;
END //

DELIMITER ;
```

2. PHP Code to Call the Stored Procedure



```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Assuming $conn is your database connection

    $user_id = 1; // Example user ID
    $flat = $_POST['flat'];
    $building = $_POST['building'];
    $area = $_POST['area'];
    $town = $_POST['town'];
    $city = $_POST['city'];
    $state = $_POST['state'];
    $country = $_POST['country'];
    $pin_code = $_POST['pin_code'];

    // Prepare the call to the stored procedure
    $update_address = $conn->prepare("CALL update_user_address(?, ?, ?, ?, ?, ?, ?, ?, ?, @message)");
    $update_address->bind_param('issssssss', $user_id, $flat, $building, $area, $town, $city, $state, $country, $pin_code);
    $update_address->execute();

    // Retrieve the output parameter
    $select_message = $conn->query("SELECT @message");
    $message = $select_message->fetch_assoc()['@message'];
    $select_message->free();

    // Output the message
    echo '<p>' . $message . '</p>';

    $update_address->close();
}
?>
```

14. Conclusion

In conclusion, the development of the responsive restaurant front-end system and admin dashboard for Hotel 'Jagabay' has significantly improved operational efficiency and customer satisfaction. The systems are designed to be scalable, with room for future enhancements based on user feedback and technological advancements.