E-COMMERCE WEBSTIE (BACKEND REPORT)

- **TECHNOLOGY USED :** - PHP + MYSQL(BACKEND)
- **TECHNOLOGY USED** :- HTML,CSS,JS/REACT JS FRAMEWORK

1.DATABASE DESIGN

- **User Table**

  Attributes :- userId,email,name,password,cart products,ordered products,wishlistproducts,address,dates,imagefile,number,reviews , rating.
- **Admin Table**

  Attributes :- AdminId,name,email,image,number,password,role.

2.Admin Dashboard

- **Add Products Table**

  Attributes:-productid,productName, category,Price,quantity,date,images,discount,rating ,size.
- **Category Table**

  Attribures ;- categoryID, categoryName, categoryImage.
- **Review & Rating Table**

  Attribures :- reviewId, productid, ReviewTitle, review, rating,images,timestamp.
- **Order Table**

  Attributes :- OrderID, orderItemID, date,quantity,amount,productid.
- **Add User Table**

  Attributs :- userId, username, email,password , number,address.

3.User Authentication

- **Register: POST /api/register**
- **Login : POST/api/login**
- **Logout : POST/api/logout**
- **Profile : GET/api/user/profile(need user authentication)**

4. Admin Functionality

- **Add Product**: **POST /api/admin/products**
- **Update Product**: **PUT /api/admin/products/{product_id}**
- **Delete Product: DELETE /api/admin/products/{product_id}**
- **Manage Orders: GET /api/admin/orders**
- **Update Order Status: PUT /api/admin/orders/{order_id}**

5. Shipping Orders API

For shipping orders, integrating with third-party APIs like Shiprocket,Shippo, EasyPost, or FedEx can be beneficial. Below is an example of integrating with Shippo.

1. Install Shippo SDK:

composer require goshippo/shippo-php

2. Create a Shipment

```
use Shippo;
Shippo::setApiKey('YOUR_API_KEY');
$addressFrom = [
    'name' => 'Sender Name',
    'street1' => '1234 Main Street',
    'city' => 'San Francisco',
    'state' => 'CA',
    'zip' => '94111',
    'country' => 'US'
];

$addressTo = [
    'name' => 'Recipient Name',
    'street1' => '1234 Market Street',
    'city' => 'San Francisco',
    'state' => 'CA',
    'zip' => '94105',
    'country' => 'US'
];

$parcel = [
    'length' => '5',
    'width' => '5',
    'height' => '5',
    'distance_unit' => 'in',
    'weight' => '2',
```

```
    'mass_unit' => 'lb'
];

$shipment = Shippo_Shipment::create([
    'address_from' => $addressFrom,
    'address_to' => $addressTo,
    'parcels' => [$parcel],
    'async' => false
]);

echo $shipment;
```

3.Tracking a Shipment

```
$tracking = Shippo_Track::get_status('shippo', 'YOUR_TRACKING_NUMBER');
echo $tracking;
```

6.OTP GENERATION CODE

OTP (One-Time Password) Generation for User Verification

OTPs are commonly used for user verification during registration, login, or sensitive transactions. Here's how you can implement OTP generation and verification in PHP:

Steps for OTP Generation and Verification:

1. **Generate OTP: Create a random numeric OTP.**
2. **Send OTP: Use an SMS or email service to send the OTP to the user.**
3. **Store OTP: Save the OTP in the database with an expiration time.**
4. **Verify OTP: Check the OTP provided by the user against the stored OTP.**

```
function generateOTP($length = 6) {
    $otp = '';
    for ($i = 0; $i < $length; $i++) {
        $otp .= mt_rand(0, 9);
    }
    return $otp;
}

function sendOTP($userContact, $otp) {
    // Use an SMS or email service to send the OTP
    // For SMS, you can use services like Twilio, Nexmo, etc.
    // For email, you can use PHPMailer or a similar library
    // Example with PHPMailer:

    // require 'PHPMailer/PHPMailerAutoload.php';
    // $mail = new PHPMailer;
    // $mail->isSMTP();
    // $mail->Host = 'smtp.example.com';
    // $mail->SMTPAuth = true;
    // $mail->Username = 'your_email@example.com';
    // $mail->Password = 'your_password';
    // $mail->SMTPSecure = 'tls';
    // $mail->Port = 587;
    // $mail->setFrom('your_email@example.com', 'Your Name');
    // $mail->addAddress($userContact);
    // $mail->Subject = 'Your OTP Code';
```

```
    // $mail->Body    = 'Your OTP code is ' . $otp;
    // $mail->send();
}

function storeOTP($userId, $otp) {
    // Store OTP in the database with an expiration time
    $expirationTime = date("Y-m-d H:i:s", strtotime('+5 minutes'));
    $query = "INSERT INTO otp_codes (user_id, otp, expiration_time) VALUES ('$userId', '$otp', '$expirationTime')";
    mysqli_query($connection, $query);
}

function verifyOTP($userId, $userOtp) {
    $query = "SELECT otp FROM otp_codes WHERE user_id = '$userId' AND expiration_time > NOW() ORDER BY expiration_time DESC LIMIT 1";
    $result = mysqli_query($connection, $query);
    $row = mysqli_fetch_assoc($result);

    if ($row && $row['otp'] == $userOtp) {
        // OTP is correct
        return true;
    } else {
        // OTP is incorrect or expired
        return false;
    }
}
```

7.Payment Integration

To handle payments, you can integrate with popular payment gateways like PayPal, Stripe, or Razorpay. Here's an example using Stripe:

1. Stripe Payment Integration:

```
composer require stripe/stripe-php
```

2. Create a Payment Intent:

```
require 'vendor/autoload.php';


\Stripe\Stripe::setApiKey('YOUR_STRIPE_SECRET_KEY');


function createPaymentIntent($amount) {
   try {
      $paymentIntent = \Stripe\PaymentIntent::create([
         'amount' => $amount * 100, // amount in cents
         'currency' => 'usd',
         'payment_method_types' => ['card'],
      ]);
      return $paymentIntent;
   } catch (\Exception $e) {
      return ['error' => $e->getMessage()];
   }
}
```

3. Handle Payment Confirmation on Client-Side:

On the client-side, use Stripe.js to handle the payment confirmation.

Putting It All Together

1. **OTP Generation:**

- Generate an OTP using generateOTP().
- Send the OTP using sendOTP().
- Store the OTP using storeOTP().
- Verify the OTP using verifyOTP().

2. Payment Handling:
   - Create a payment intent on the server-side using Stripe's API.
   - Handle the payment confirmation on the client-side using Stripe.js.

```
<script src="https://js.stripe.com/v3/"></script>

<script>

  var stripe = Stripe('YOUR_STRIPE_PUBLIC_KEY');



  async function handlePayment() {

    let response = await fetch('/create-payment-intent', {

      method: 'POST',

      headers: {

        'Content-Type': 'application/json',

      },

      body: JSON.stringify({

        amount: 5000 // amount in dollars

      }),

    });

    let result = await response.json();



    if (result.error) {

      console.error(result.error);
```

```
        } else {
            stripe.confirmCardPayment(result.clientSecret, {
                payment_method: {
                    card: cardElement,
                    billing_details: {
                        name: 'Customer Name',
                    },
                }
            }).then(function(result) {
                if (result.error) {
                    console.error(result.error.message);
                } else {
                    if (result.paymentIntent.status === 'succeeded') {
                        console.log('Payment successful!');
                    }
                }
            });
        }
    }
</script>
```