\* Store a newly created resource in storage.

\*/

public function store(Request $req)

{

$req->validate([

"regnum" => "required",

"price" => "required|min:1|numeric",

]);

$uniqueString = hash('sha256', microtime(true) . rand(0, 1000000));

$regnum = $req->regnum;

$email = 'netecsystems@gmail.com'; // $req->email;

$packageid = $req->packageid;

$package = $req->package;

$amount = $req->price;

$myname = $req->myname;

$desc = $myname." (". $regnum.")";

$narration = $package. " package payment for ".$desc;

//presave to get a good result url

$payment\_saved = new PayNowPayment();

$payment\_saved->regnum = $req->regnum;

$payment\_saved->amount = $req->price;

$payment\_saved->reference = $uniqueString;

$payment\_saved->status = "created";

$payment\_saved->save();

$paynow = new Paynow(

$this->PAYNOW\_INTEGRATION\_ID,

$this->PAYNOW\_INTEGRATION\_KEY,

env('APP\_URL2') . '/payconfirmation/' . $payment\_saved->id,

env('APP\_URL2') . '/payment\_return'

);

$payment = $paynow->createPayment($narration, 'netecsystems@gmail.com');

$payment->add($narration, $amount);

$response = null;

try {

$response = $paynow->send($payment);

} catch (InvalidIntegrationException $error) {

return back()

->withErrors("Invalid Integration, contact system admin for support")

->withInput(request()->all());

}

if ($response->success()) {

// Or if you prefer more control, get the link to redirect the user to, then use it as you see fit

$browser\_url = $response->redirectUrl();

$poll\_url = $response->pollUrl();

// Check the status of the transaction

$status = $paynow->pollTransaction($poll\_url);

$hash = $response->data()['hash'];

//save the payment to DB

$payment\_saved->regnum = $req->regnum;

$payment\_saved->amount = $req->price;

$payment\_saved->status = $status->status();

$payment\_saved->browserurl = $browser\_url;

$payment\_saved->pollurl = $poll\_url;

$payment\_saved->hash = $hash;

$payment\_saved->reference = $status->paynowReference();

$payment\_saved->save();

if ($payment\_saved) {

DB::table('studentpackages')->updateOrInsert(

['reference' => $reference], // Search condition

['paid' => 0, 'regnum' => $regnum, 'packageid' => $packageid] // Update or insert values

);

return redirect($browser\_url);

}

return back()

->withErrors('Saving payment details failed')

->withInput(request()->all());

}

}

public function handlePaymentConfirmation(Request $request,$id)

{

$payment = PayNowPayment::findOrFail($id);

$reference= $status = '';

try {

$poll = $this->paynow->pollTransaction($payment->pollurl);

$payment->status =$status = $poll->status();

$reference=$payment->reference;

} catch (Exception $exception) {

return abort(500, $exception->getMessage());

}

// Check if the payment is successful

if ($status === 'Paid') {

// Update the application fee status if needed

DB::table('paynowpayments')

->where('id', $id)

->update(['status' => 'Paid']);

DB::table('studentpackages')

->where('reference', $reference)

->update(['paid' => 1]);

return response()->json(['message' => 'Payment confirmed and saved successfully'], 200);

} else {

// Handle unsuccessful payment

return response()->json(['error' => 'Payment was not successful'], 400);

}

}